Technical Report

UCAM-CL-TR-677
ISSN 1476-2986

Number 677

UNIVERSITY OF CAMBRIDGE

Computer Laboratory

ECCO: Data centric asynchronous communication

Eiko Yoneki

December 2006

15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom
phone +44 1223 763500

http://www.cl.cam.ac.uk/

This technical report is based on a dissertation submitted September 2006 by the author for the degree of Doctor of Philosophy to the University of Cambridge, Lucy Cavendish College.

Technical reports published by the University of Cambridge Computer Laboratory are freely available via the Internet:

http://www.cl.cam.ac.uk/techreports/

ISSN 1476-2986

Abstract

The thesis of this dissertation deals with ubiquitous and wireless pervasive computing. The main focus is on data centric networking in distributed systems, which relies on content addressing instead of host addressing for participating nodes, thus providing network independence for applications. Publish/subscribe asynchronous group communication realises the vision of data centric networking that is particularly important for networks supporting mobile clients over heterogeneous wireless networks. In such networks, client applications prefer to receive specific data, requiring selective data dissemination. Underlying mechanisms such as asynchronous message passing, distributed message filtering and query/subscription management are essential requirements. Furthermore, recent progress in wireless sensor networks brought a new dimension of data processing in ubiquitous computing, where the sensors are used to gather high volumes of different data types and to feed them as contexts to a wide range of applications. This data processing requires advanced mechanisms that make intelligent use of simple data to create meaningful information. The following specific subjects in ubiquitous computing environments are addressed.

Particular emphasis has been placed on fundamental design of event representation. Besides the existing event attributes, event order, and continuous context information such as time or geographic location could be incorporated within an event description. Data representation of event and query will be even more important in future ubiquitous computing, where events flow over heterogeneous networks. This dissertation presents a multidimensional event representation (i.e., Hypercube structure in RTree) for efficient indexing, filtering, matching, and scalability in publish/subscribe systems. The hypercube event with a typed content-based publish/subscribe system for wide-area networks is demonstrated for improving the event filtering process.

As a primary focus, this dissertation investigates a structureless, asynchronous group communication over wireless ad hoc networks named *ECCO* Pervasive Publish/Subscribe* (ECCO-PPS). ECCO-PPS uses a context adaptive controlled flooding, which takes a cross layer approach between middleware and network layer and provides a content-based publish/subscribe paradigm. Traditionally events have been payload data within network layer components; the network layer never touches the data contents. However, application data have more influences on data dissemination in ubiquitous computing scenarios. The state information of the local node may be the event forwarding trigger. Thus, the model of publish/subscribe must become more symmetric, with events being disseminated based on rules and conditions defined by the events themselves. The event can thus choose the destinations instead of relying on the potential receivers' decision. The publish/subscribe system offers a data centric approach, where the destination address is not described with any explicit network address. The symmetric publish/subscribe paradigm brings another level to the data centric paradigm, and this new paradigm leads to a fundamental change of the functionality at the network level of asynchronous group communication and membership maintenance. Dynamic channelisation by clustering the subscriptions is also attempted.

To add an additional dimension of event processing in global computing, understanding event aggregation, filtering and correlation is an important issue. Event correlation is discussed as part

of publish/subscribe functionality. Temporal ordering of events is essential for event correlation over distributed systems. This dissertation introduces generic composite event semantics with interval-based semantics for event detection. This precisely defines complex timing constraints among correlated event instances. The sensed data must be aggregated and combined into higher-level information or knowledge at appropriate points, while flowing over heterogeneous networks. This issue should be integrated within the context of communication mechanisms in ubiquitous computing.

The major goal of this dissertation is to provide advanced data centric asynchronous communication, which provides efficiency, reliability, and robustness, while adapting to the underlying network environments.

* *ECCO* means *here I am, here it is,...* in Italian. Italian telephone conversations start with *Pronto* meaning *hello*, often followed by *Ecco*. My research project names (*Pronto, Ecco*) are inspired by an Italian tele-conversation protocol.

List of Acronyms

ADC:	Analogue to Digital Converter
ADT:	Abstract Data Type
ALM:	Application Level Multicast
ALMRP:	Application-Level Multicast Routing Protocol
AMRoute:	Ad hoc Multicast Routing
AODV:	Ad Hoc On-Demand Distance Vector
ARBM:	Associativity Based Multicast
ARMP:	Adaptive Reliable Multicast Protocol
BMW:	Broadcast Medium Window
BT:	Bluetooth (IEEE 802.15.1)
CAN:	Content Addressable Networks
CBM:	Content Based Multicast
CEA:	Cambridge Event Architecture
CGM:	Group Multicast Algorithm
COBEA:	CORBA Based Event Architecture
CORBA:	Common Object Request Broker Architecture
CQL:	Continuous Query Language
DAML:	DARPA Agent Markup Language
DBMS:	Database Management System
DHT:	Distributed Hash Table
DOLR:	Decentralised Object Location and Routing
DSDV:	Destination-Sequenced Distance Vector
DSR:	Dynamic Source Route
DTN:	Delay Tolerant Network
DVMRP:	Distance Vector Multicast Routing Protocol
DYMO:	Dynamic MANET On-demand routing protocol
ECA:	Event-Condition-Action
ESB:	Enterprise Service Bus
ESP:	Event Space Partitioning
FG:	Forwarding Group
FGMP:	Forwarding Group Multicast Protocol
FSA:	Finite State Automata
FSM:	Finite State Machine
FSP:	Filter Set Partitioning
GCG:	Global Clock Generator
GHT:	Geographic Hash Table
GPRS:	General Packet Radio Service
GPS:	Global Positioning System
GPSR:	Greedy Perimeter Stateless Routing
GSM:	Global System for Mobile Communications

IETF:	Internet Engineering Task Force
JMS:	Java Message Service
JNS:	Java Network Simulator
JXTA:	Juxtapose - Java P2P framework
LANMAR:	Landmark Ad Hoc Routing
LBM:	Location Based Multicast
LER:	Last Encounter Routing
LLCP:	Lightweight Local Clock Propagation
MANET:	Mobile Ad Hoc Network
MAODV:	Multicast Ad hoc On-Demand Distance Vector
MBR:	Minimum Boundary Rectangle
MEMS:	Micro Electro Mechanical System
MF:	Message Ferry
MP2P:	Mobile Peer-to-Peer Network
NAM:	Network Animator
NTP:	Network Time Protocol
ODMRP:	On-Demand Multicast Routing Protocol
OGSA:	Open Grid Services Architecture
OLSR:	Optimised Link State Routing
OOM:	Object Oriented Middleware
OSGi:	Open Services Gateway Initiative
OWL:	Web Ontology Language
P2P:	Peer-to-Peer Network
PAN:	Personal Area Network
PDA:	Personal Digital Assistant
PPR:	Parametric Probabilistic Routing
RB:	Reliable Broadcast
RDF:	Resource Description Framework
RDFS:	Resource Description Framework Schema
RDG:	Route Driven Gossip
RMA:	Reliable Multicast Algorithm
RMI:	Java Remote Method Invocation
RMTP:	Reliable Multicast Transport Protocol
RON:	Resilient Overlay Network
RST:	Range Search Tree
SLP:	Service Location Protocol
SMS:	Short Message Service
SOA:	Service Oriented Architecture
SQL:	Structured Query Language
UMTS:	Universal Mobile Telecommunications System
UPnP:	Universal Plug and Play
URI:	Uniform Resource Identifier
UTC:	Universal Coordinated Time
UWB:	Ultra Wide Band
WLAN:	Wireless Local Area Network
WMN:	Wireless Mesh Network
WSDL:	Web Services Description Language

WSN:	Wireless Sensor Network
WWW:	World Wide Web
XML:	Extensible Markup Language
XPath:	XML Path Language
ZRP:	Zone Routing Protocol

Contents

1 Introduction

The thesis of this dissertation focuses on data centric networking in distributed systems, which relies on content addressing and a symmetric communication paradigm between senders and receivers. Integration of complex data processing with networking is a key vision for future ubiquitous computing.

A rapid increase of event monitoring capability by wireless sensors is driving the next evolutionary stage in ubiquitous computing, where numerous and increasingly more pervasive computers are embedded in everyday-life scenarios. We will soon be drowning in an ocean of events captured by pervasive devices in daily life. Events flow from pervasive devices to the Internet, and network environments will be heterogeneous, ranging from remote sensor networks to Internet scale wide-area networks. Traditional systems for collecting, aggregating, and delivering these events are unable to cope with the dynamism of event sources and the volume of event messages. This data processing requires an advanced mechanism that makes intelligent use of simple data to create meaningful information. Events reflect the real world and provide contexts for computing, such as spatial information or specific event correlation.

There will be a wide range of applications in this new generation of ubiquitous computing scenarios, including environmental monitoring (e.g., water quality, air pollution), weather forecast distribution, social and community network building, health care, disaster recovery coordination, financial monitoring, e-science, and e-commerce.

1.1 Global Computing

The integration of smart sensors with the Internet or wireless infrastructure networks increases the coverage area, where sensor networks, Peer-to-Peer (P2P) systems, grid systems and many other applications collaborate. Thus, the global scale of applications needs to provide query processing and deliver real-time data from many distributed sources and use shared Internet resources to aggregate, filter or disseminate the sensed data. These systems have a wide range of problems, from routing strategy and dynamic query optimisation to reliability. Heterogeneous network environments can be classified into the following categories:

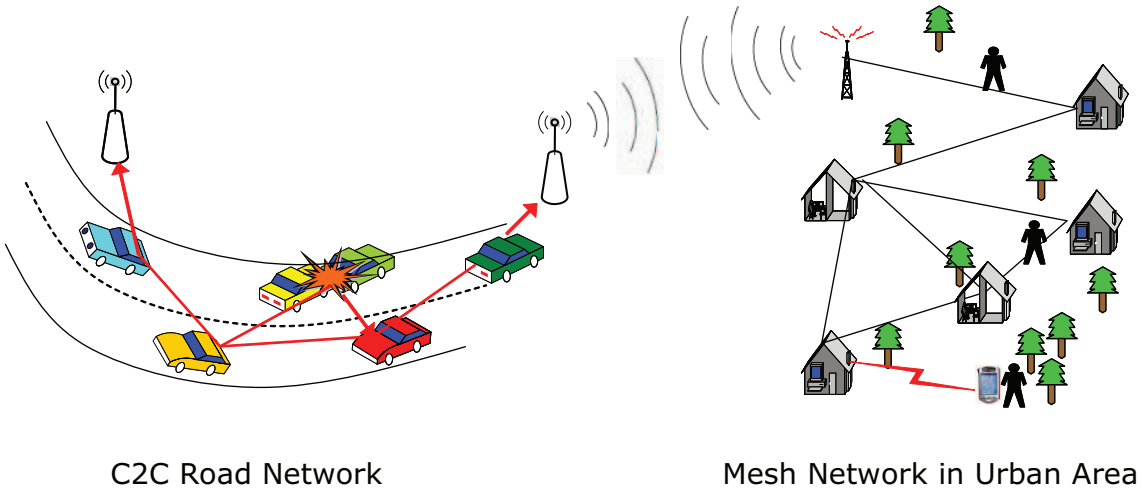


Figure 1.1: Inter-Car Vehicular communication

- Large-scale wired networks (e.g., P2P networks over the Internet)
- Pervasive dynamic mobile networks (e.g., Mobile Ad Hoc Networks (MANETs))
- Static or semi dynamic embedded networks (e.g., Wireless Sensor Networks (WSNs))

Vehicular networks are an example of the second category, which consists of the following communication types:

- On-board devices are connected to each other in real-time to detect alert conditions.
- Various inter-car communications (e.g., infrastructure-less Universal Mobile Telecommunications System (UMTS), direct communication by radio, indirect communication by store-and-forward routing).
- Car-to-fixed infrastructure based communications (e.g., General Packet Radio Service (GPRS), UMTS, Global System for Mobile Communications (GSM), Short Message Service (SMS), and Wireless Local Area Networks (WLANs)).

Fig. 1.1 depicts a vehicular network and an urban mesh network [AWW05]. A car accident is observed by surrounding cars, and they communicate in ad hoc mode to carry the information towards a car within communication distance of a road side Internet access point. This information is used to provide an alert of traffic congestion near the accident area via wireless mesh. The wireless mesh covers an urban area to provide Internet access through predefined connections among participants of the mesh.

In the urban area, people walking with mobile devices can collect information on the air quality such as carbon monoxide concentration and transmit this information via the nearby mesh endpoint to the monitoring applications. Fig. 1.2 depicts the propagation of networks in the vertical direction for network types and in the horizontal direction for spatial aspects.

The pervasive ad hoc networks play important roles not only for creating ad hoc communication but also for collecting sensor data and conveying it to Internet backbone nodes, where no network infrastructure support exists in remote locations. Moving groups are useful candidates for forming a communication bridge from remote locations to the infrastructure based networks. The group can be formed in trains, airplanes, ships and even among strangers walking on the same road.

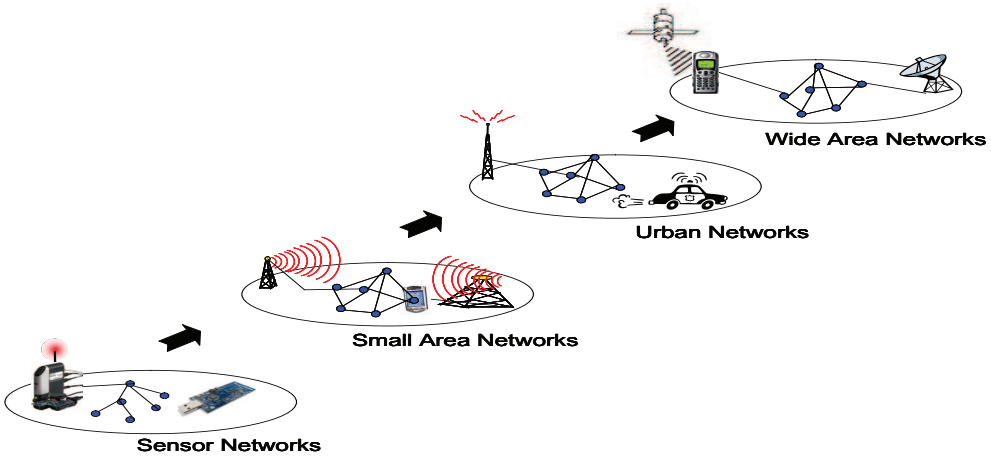


Figure 1.2: Propagation of Networks

A new type of communication is required, where networks will be structureless and rely on ad hoc connections between nearby nodes to establish multi-hop dynamic routes to propagate data. Opportunistic networks and pre-constructed wireless meshes have great potential for conveying time-critical sensor data. It is desirable to create a communication paradigm by applying an abstraction layer over hybrid wireless network environments to maximise existing wireless connectivity or data dissemination reliability by store-and-forward mechanisms, where the abstraction layer can be any type of overlay such as a moving group consisting of humans or logical mobile agents. Each network environment consists of another level of heterogeneous networks, which can be overlay networks or networks with various media.

Management of communication among combinations of mobile devices and ad hoc networks is best achieved through the creation of highly dynamic, self-organising, mobile peer-to-peer (MP2P) systems. A key aspect here is to achieve global computing, where a large user population accesses information, and information sources vary widely in size and complexity.

1.2 Event Driven Reactive Systems

The term event indicates that something happens that we can observe and respond to. For example, when it starts raining, we will open an umbrella. Then we might decide to change the plan of the day, which leads to phone calls to friends to rearrange the plan for the evening. We observe events and react to them not only locally but also propagate a chain of reactions. Applying this to distributed applications, an event is a state change in the real world, which is propagated as a message among applications over heterogeneous network environments. Thus, event notification is set to become a crucial building block in future ubiquitous systems.

Network environments will be highly decentralised and distributed over a multitude of different devices that are dynamically networked and interact in an event-driven mode. System entities are often loosely coupled and highly dynamic, with entities coming up and going down periodically and the system evolving constantly. Decentralised management is essential, and effective abstraction for building adaptive distributed applications over heterogeneous network environments is desirable. This requires a new generation of middleware and components that can function and exchange data in these highly dynamic environments over hybrid wireless networks. The middleware will support a reactive programming paradigm for sensing, decision making and actuating. The middleware platform will provide an open, highly configurable and self-adaptive platform,

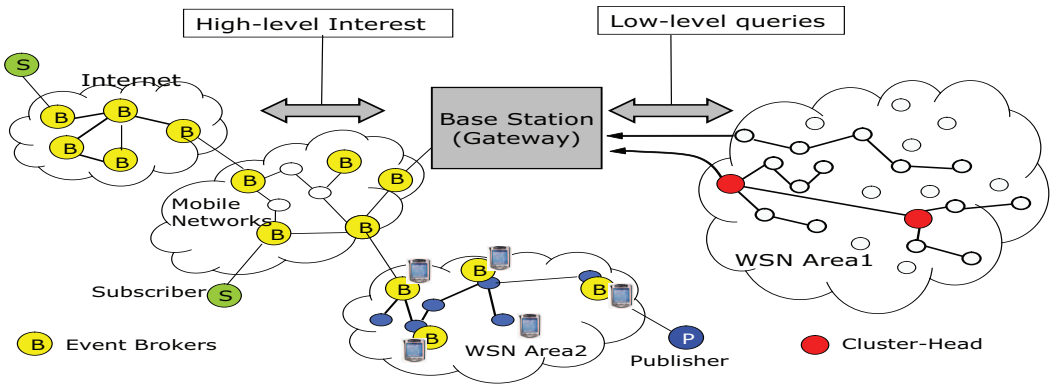


Figure 1.3: Bridging WSNs to the Internet

where sensed data can be shared among different applications over large-scale environments.

The publish/subscribe paradigm is powerful as an underlying communication mechanism for such middleware systems. With the recent evolution of distributed event-based middleware over a P2P overlay network, the construction of event broker grids will extend a seamless asynchronous communication capability over heterogeneous network environments. Broker nodes within publish/subscribe systems that offer data management services (e.g., aggregation, filtering, and correlation) can efficiently coordinate data flow. This supports the construction of a functional overlay and ultimately supports an active framework. P2P offers a promising paradigm for developing efficient distributed systems and applications, while aligning grid technologies with Web Services (see Chapter 3). Fig. 1.3 depicts the view of a P2P based event broker system, where two wireless sensor networks (WSNs) are deployed and a mobile network bridges between the base station and the Internet backbone. *B* indicates the broker nodes in the event-based system. The *WSN Area 1* contains two cluster heads, which can directly communicate with the base station. The subscriptions for publish/subscribe systems are considered *High-level Interest*, which is propagated to the WSN through the base station after being broken down to the *Low-level queries* (see Section 7.4, [YB05b] for details).

1.3 Data Centric Asynchronous Communication

The data centric approach relies on content addressing instead of host addressing for participating nodes, thus providing network independence for applications. Publish/subscribe asynchronous group communication realises the vision of data centric networking that is particularly important for networks supporting mobile clients over heterogeneous wireless networks. In such networks, client applications prefer to receive data specific to them, thus requiring selective data dissemination. Underlying mechanisms such as asynchronous message passing, distributed message filtering and query/subscription management are essential aspects.

Traditionally, events have been payload data within network layer components; the network layer never touches the data contents. Application data are more influential to data dissemination in ubiquitous computing. For example, it is important to decide whether to forward the data based on spatial information of subscriber nodes, when data is meaningful at the certain location. The state information of the local node may be the event forwarding trigger.

Thus, the publish/subscribe model must become more symmetric, where an event is disseminated based on the rules and conditions defined by the event itself. The event can select the destinations instead of relying on the potential receivers' decisions. The first data centric paradigm was

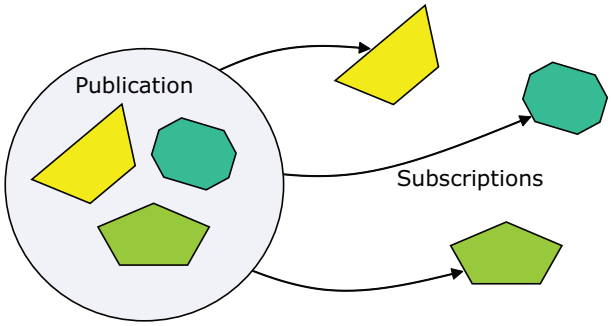


Figure 1.4: Asymmetric Publish/Subscribe Model

realised by decoupling of destination and explicit network addresses. The symmetric publish/ subscribe paradigm brings another level to the data centric paradigm, and this new paradigm leads to a fundamental change in the functionality at the network level of asynchronous group communication and membership maintenance. Fig. 1.4 shows the traditional publish/subscribe model, in which the subscriptions are the complete subset of publications. On the other hand, Fig. 1.5 depicts that the publications are disseminated based on the rules and conditions defined by the publication itself. The publisher rather than the subscriber can choose the destinations. The publishing conditions can be geographical information or any local information of potential receivers. Gossip-based (epidemic) dissemination determines forwarding decisions based on the given parameter of probability. The symmetric dissemination mechanism can define this parameter for each publication individually.

This new type of publish/subscribe model must be well abstracted. A key is the event model itself including data structures with flexible indexing capability, which includes additional information or criteria for network behaviour so that symmetric publish/subscribe is possible. Data fragments are dispersed among devices, services and agents, and the level of this data can differ in ubiquitous computing. These data appear as unstructured text that could be written in multiple languages using characters with different encodings. Yet knowledge must be represented for computations, whether the focus is the data itself or their use. Heterogeneity of information over global distributed systems must be considered, and the information sensed by the devices must be aggregated and combined into higher-level information or knowledge that will ultimately be

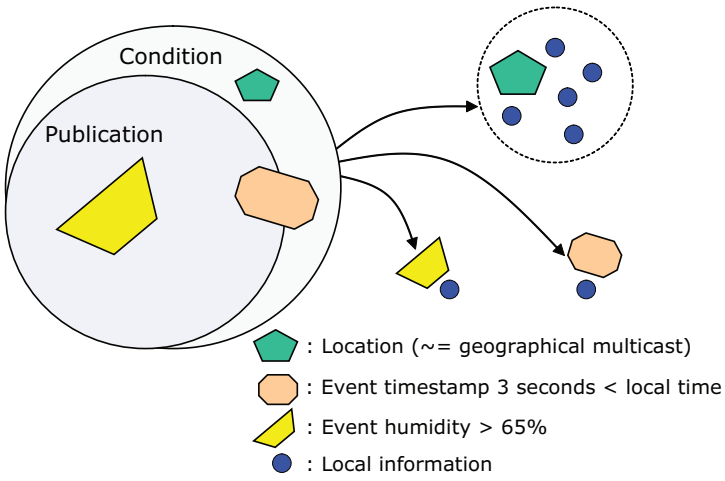


Figure 1.5: Symmetric Publish/Subscribe Model

delivered to subscribers.

Thus, we need to consider the general event model that applies throughout the network. More descriptive queries will be required to accomplish a selective event dissemination mechanism and reduce network traffic over wireless networks. Each persistent query has to be properly placed distributed for effective event detection. This happens over Internet scale wide-area networks as well as isolated embedded sensor network systems. Furthermore, an important challenge here is providing advanced data processing including event aggregation, filtering, and correlation to give efficiency, reliability, and robustness, while adapting to underlying network environments.

1.4 Thesis Contribution

The described research for ubiquitous computing covers diverse areas, including distributed system design, wireless communication, information theory, P2P networking, embedded systems, data mining, language technology, and intelligent agents. The goal of the thesis of this dissertation is focused on several key issues for data centric communication aspects to construct an event-based distributed system. These issues are described in the following four sections.

1.4.1 Multidimensional Event Model

In the publish/subscribe communication paradigm, defining events without unambiguous semantics is important, because events flow in diverse networks. This requires a fundamental design of event representation. Besides the existing event attributes, event order and continuous context information such as time or geographic location must be incorporated within an event description. Thus, to provide more accurate temporal correlation, *duration* is defined. Another aspect is that sensor data is high-volume, multidimensional and dynamic. I present a multidimensional event representation (i.e., *the Hypercube* structure in RTree [Gut84]) for efficient indexing, filtering, matching, and selective dissemination in publish/subscribe systems.

1.4.2 Content-Based Publish/Subscribe with Hypercube Filter

A range query is a difficult subscription to implement in content-based publish/subscribe. I apply *Hypercube* events to a content-based publish/subscribe system and experiment with the effect of multidimensional filtering. Hermes [PB02] is a typed content-based publish/subscribe system for wide-area networks from our group, which uses the algorithm of Scribe [CDK+02] as a base over Pastry. It also combines attribute filtering using a similar algorithm to SIENA [CRW01] to support content-based publish/subscribe. Current filtering is a set of predicates, and the matching mechanism is basic. I integrate *Hypercube* events with Hermes to add efficient event filtering.

1.4.3 Structureless Asynchronous Group Communication

Mobile/wireless networks are dynamic and resource constrained. Existing network protocols mainly focus on end-to-end communication. To increase performance, these protocols use contexts (location, topology etc.). The most influential context to impact event dissemination is the event itself, which comes from applications. Thus, an integrated approach is necessary to cross from application to network layers for more efficient event dissemination (see Section 6.3.1). Events must therefore be treated as part of a communication token.

I investigated a structureless asynchronous group communication system over wireless ad hoc networks named *ECCO Pervasive Publish/Subscribe* (ECCO-PPS). ECCO-PPS uses a context

adaptive controlled flooding, which takes a cross layer approach between middleware and the network layer and provides a content-based publish/subscribe paradigm. It realises the symmetric publish/subscribe paradigm.

Future wireless networks will be hybrids, and pure ad hoc networks will play relatively minor roles for deployment in the real world. Data centric communication abstractions such as ECCO-PPS will help in constructing reactive distributed applications.

1.4.4 Unified Event Correlation Semantics

In event-based middleware systems, an event correlation service allows consumers to subscribe to patterns of events (composite events). This provides an additional dimension of data management, better selective event dissemination, scalability and performance in distributed systems. The information sensed by the pervasive devices must be aggregated and combined into higher-level information or knowledge at the appropriate point, while flowing over heterogeneous networks. Thus, there is a strong need for an event correlation service in ubiquitous computing environments. Temporal ordering of events is a crucial aspect for event correlation in such systems. I introduce novel generic composite event semantics with interval-based semantics for event detection. This precisely defines complex timing constraints among correlated event instances.

Thus, a defined event model, combined with symmetric publish/subscribe, can unify communication and content in heterogeneous ubiquitous computing environments.

1.5 Dissertation Structure

The background and the vision of ubiquitous computing are described in Chapters 2 and 3 followed by four chapters focusing on specific subjects, and Chapter 8 concludes the dissertation.

Chapter 2 overviews wireless networks, publish/subscribe systems, and event correlation.

Chapter 3 outlines the system architecture for ubiquitous computing, and highlights the major subjects in this dissertation.

Chapter 4 defines the event query model and presents a hypercube-based event query representation for symmetric publish/subscribe.

Chapter 5 illustrates the extension of Hermes using hypercube for content-based filtering and discusses future P2P adaptation.

Chapter 6 presents ECCO-PPS, structureless content-based publish/subscribe for wireless mobile ad hoc networks.

Chapter 7 discusses event correlation and presents unified semantics for filtering, aggregation, and correlation.

Chapter 8 concludes this dissertation and discusses potential extensions.

2

Background

Recent progress has led mobile devices to become ubiquitous. For example, Wireless Sensor Networks (WSNs) are composed of wireless sensor nodes distributed in the environment and include various sensors (e.g., cameras, microphones, or temperature sensors). Each node is equipped with a wireless communication transceiver, sensor, power supply unit, machine controllers, and microcontrollers on Micro Electro Mechanical System (MEMS) chips that are only a few millimetres square. Automatic, self-organising and self-managing systems will be essential for such ubiquitous environments, where billions of computers are embedded in everyday life. This heterogeneous collection of devices will interact with sensors and actuators embedded in our homes, offices and transportation systems, all of which will form an intelligent pervasive environment and will be integrated in the Internet. This new dimension of ubiquitous computing requires more complex communication mechanisms and, most importantly, intelligent data processing throughout the networks.

Network resources will be ubiquitously distributed from tiny wireless sensor networks to the wide area Internet. For example, cars, laptops and PDAs, will usually be connected, regardless of their locations. They may collect fragmented data from isolated ad hoc networks to carry to Internet nodes. Meanwhile, many applications using physical information such as geographical locations will appear.

In daily life, the synchronous polling mode dominates the search for information on the World Wide Web (WWW). A complementary model, asynchronous publish/subscribe event notification is becoming popular, where a user subscribes to specific events and receives notifications when any of these events are published. Amazon shopping alerts, the auction notification of *eBay*, and stock quotes and news alerts are all examples of this model. In ubiquitous computing scenarios, applications that communicate with WSNs to perform automation tasks will rely on the event notification model. Non-human subscribers increase the scalability and impact the design of publish/subscribe systems, because of the high number of subscribers/publishers, more complex subscriptions, and high rate of event processing.

Research in ubiquitous computing is multi-disciplinary, including information theory, databases, networking, embedded systems and signal processing to investigate applications that require

acquisition, processing and control of information over a distributed network infrastructure. Network automation has to be integrated with physical and network control layers to manage the network dynamically. This requires new approaches to design network elements that provide reconfigurability for continuous performance monitoring to support consistent performance, reliability and recovery from network failure.

This chapter outlines the technologies surrounding ubiquitous computing from the event-based system's perspectives. Insight into specific technologies will be provided in the corresponding chapters.

2.1 Emergence of Loosely Coupled Communication

Synchronous communication, as provided in object oriented middleware (OOM) such as Java RMI and CORBA, is not appropriate for loosely coupled senders and receivers. OOM-providers added asynchronous message passing extensions, but static dependencies remained and they did not support wide-area, or large-scale inter-networks, where participants might be anonymous, detach, or move. Message Oriented Middleware (MOM), underlying event-based systems, enables inter-operation of heterogeneous, distributed, and dynamically changing components of large information systems. This is acknowledged in the business domain through the provision of Sun's JMS API [Mic01] as a common interface for Java applications to IBM's MQSeries [IBM00], Microsoft Message Queue (MSMQ), TIBCO's TIB/Rendezvous [TIB98], Softwired's iBus [Sof98], and BEA's WebLogic [Web]. They evolved into event-based middleware, based on the publish/subscribe communication paradigm and have become popular because asynchronous, many-to-many communication is well suited for constructing reactive distributed systems. Some early research projects defined content-based notification systems, such as the Cambridge Event Architecture (CEA) using a programming language for typed events [BBH+95].

While the majority of commercially developed systems are centralised, efforts in research are focusing on better scalability by exploiting distributed event-based middleware architectures over peer-to-peer (P2P) networks. The P2P style interaction model facilitates sophisticated peer interactions involving advertising resources, search and subsequent discovery of resources, request for access to these resources, responses to these requests, and message exchange between peers. As a result, a series of application-level multicast systems has emerged over P2P. Scribe [CDK+02] is a topic-centric publish/subscribe messaging system based on the P2P platform Pastry [RD01]. Pastry utilises routing mechanisms to achieve scalability. SIENA [CW03] [CRW01], Gryphon [Res01], Hermes [PB02] and Narada [Nar02] are also built over P2P networks.

P2P based messaging, with the publish/subscribe paradigm, has emerged as the most promising approach to communication over wide area networks, but it lacks integration with mobile computing. In the enterprise domain, extensions that support mobile computing in event-based middleware are provided by adding an edge server to manage mobile devices, as in Softwired iBus/Mobile [Sof98]. iBus/Mobile is designed as an extension of J2EE application servers. It includes a messaging middleware client library compatible with the JMS standard as well as a middleware gateway used to connect mobile applications to J2EE application servers. It supports mobile communication-specific protocols such as GPRS, UMTS, and CDPD. This approach is not fully scalable to more dynamically extended businesses over the Internet via web services or mobile computing, because it requires static deployments of servers and gateways.

Sun Microsystems' JXTA [Mic00] is a library specification for P2P computing, defining three layers: a core layer, a service layer, and an application layer. The application layer wraps all the applications that are developed by JXTA programmers. The service layer contains services simplifying the development task for the programmer. The JXTA community currently implements

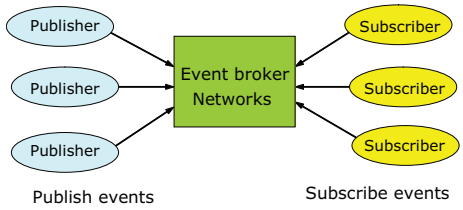


Figure 2.1: Publish/Subscribe Model

services such as protocols for service discovery and many-to-many communication. The core JXTA layer consists of protocols ensuring basic communication between peers, message routing and peer group creation.

The term event-based middleware differs from event-based communication, because it requires not only a basic event dissemination mechanism but also a set of services such as discovery of services, reliability, fault-tolerance, security, transactions, mobility and an abstraction of the underlying communication, which makes a middleware more complete by providing a distributed platform to applications.

2.2 Publish/Subscribe Paradigm

Publish/Subscribe is a powerful abstraction for building distributed applications. Communication is message-based and can be anonymous, where participants are decoupled giving the advantage of removal of static dependencies in a distributed environment. It is an especially good solution to support highly dynamic, decentralised systems (e.g., wired environments with huge numbers of clients, MANETs, and P2P).

Most distributed event-based middleware contains three main elements: a publisher who publishes events (messages), a subscriber who subscribes his interests to the system, and an event broker network to match and deliver the events to the corresponding subscribers (see Fig. 2.1). Event brokers are usually connected in an arbitrary topology. The characteristics of publish/subscribe are:

- in space: no direct connection or knowledge between clients
- in flow: no synchronised operation is required on event publishing and subscribing
- in time: no need to be running at the same time

In a distributed event-based middleware, the event brokers form an agent network providing routing, matching events, and filtering services (see Fig. 2.2):

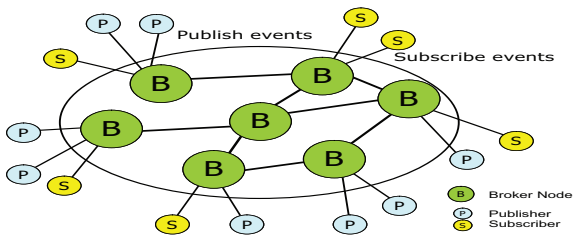


Figure 2.2: Distributed Publish/Subscribe System

2.2.1 Underlying Network Environments

The majority of enterprise commercial products implement event-based middleware in a centralised form with TCP-based transport. Alternatively, IP Multicast is deployed within event-based middleware in limited environments (see [Sof98] [TIB98]). In distributed network environments such as P2P environments, it is becoming common to construct event-based middleware over overlay networks [CNF98b] [CW03] [PB02]. This approach is essentially the application level of multicast deployment. Event-based middleware creates a dissemination tree to distribute messages published by the event source and to maintain the tree. Thus, event-based middleware can no longer take advantage of network layer multicast directly to forward events to event brokers, because complex partial matching needs to be performed at each step, unless there is some interface established to pass this context from the middleware to the network layer. Because context for the reconfiguration comes from all applications (e.g., business logic), middleware (e.g., filtering), and network layer (e.g., location), it is necessary to define an interface for it.

2.2.2 Subscription Model

Most early event-based middleware systems are based on the concepts of group (channel) or topic communication (i.e., topic-based publish/subscribe). These systems categorise events into pre-defined groups. Topic-based publish/subscribe is an abstraction of numeric network addressing schemes. In Fig. 2.3, three topics are defined (i.e., *Global Warming, Hurricanes*, and *Air Pollution*) for news forms, along with news subjects, $S1$ to $S6$, belonging to matching topics. Subject $S3$ (*Hurricane Katharina*) belongs to both topics $T1$ and $T2$.

On the other hand, content-based subscription used in SIENA [CW03], Gryphon [Res01], and Elvin [SA98] delivers the messages depending on their content; applications can therefore select different combinations of messages without changing the addressing structure. Content-based subscription extends the capability of event notification with more expressive subscription filters compared to topic-based subscription. The most advanced and expressive form of subscription language is content-based with pattern matching, which is important for event notification. Common topic-based systems arrange topics in hierarchies, but a topic cannot have several super topics. Research is also ongoing to structure complex content-based data models [MFB02] and reflection-based filters [EFGK03]. XRoute [SCG01] proposes an approach for content-based routing of XML data in mesh-based overlay networks. Fig. 2.4 depicts a content-based subscription with two attributes, where ranges values are specified in both attributes.

Type-based subscription (introduced in [BBH+95] [EFGS00] [EFGK03]) provides a natural way for multiple sub-typing of events. The event type model integrates with the type model of an object-oriented programming language, thus avoiding any explicit message classification through topics.

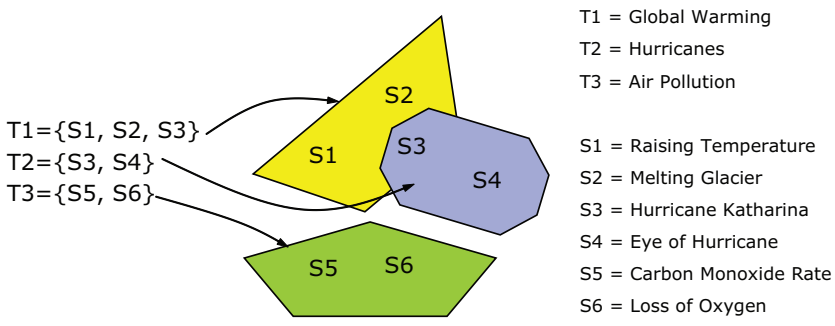


Figure 2.3: Topic-Based Model

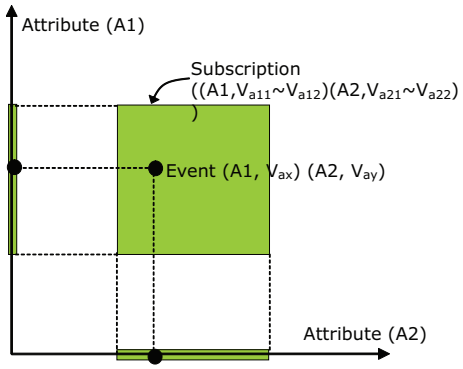


Figure 2.4: Content-Based Model

Events are treated as first class objects, and subscribers specify the class of objects they are willing to receive. In the *Cambridge Event Architecture* (CEA) [BBH+95] [Hay96], complex subscriptions based on composite event patterns are supported with attribute filtering. However, most type-based subscription models provide no attribute-based filtering, because it breaks encapsulation principles. Instead, an arbitrary method can be called on the event object to provide a filtering condition. It combines publish/subscribe with object-orientation, but an efficient implementation would be difficult since a filter expression can be arbitrarily complex and is therefore difficult to optimise or distribute. Large scale distributed systems benefit from expressing subscriptions based on attribute values providing fine-grained selective information dissemination. Thus, the combination of hierarchical topics and efficient content filtering could provide a more flexible approach for mobile applications. There are efforts to build content-based subscription with distributed hash tables by automatically organising the content into several topics [TAJ03].

The combination of hierarchical topics and high speed content filtering provides the flexibility necessary to allow applications to evolve beyond their initial design. Content-based publish/ subscribe is essential for better (filtered) data flow in mobile computing.

2.2.3 Routing Scheme

The dynamic construction of event dissemination trees to route events from publishers to all interested subscribers is the biggest challenge for supporting content-based subscription in distributed environments.

A naive way of performing distributed publish/subscribe operations is to propagate all messages to every broker, and to perform content-based matching operations at each local node (e.g., JEDI [CNF98a]). This approach may maintain a large routing table at the tree root. A simple optimisation can be done at every server to keep the summary of all subscriptions. Each local node can propagate its local summary of subscriptions to the brokers using a reverse path of event dissemination. Examples are [CDW01], [GKP99], [BCM+99], and [YEG99]. Most of these examples block unnecessary event traffic at the earliest point by comparing subscription covering. SIENA [CRW01] formalises this coverage notion by managing subscription filters as a partially ordered set. Events in SIENA are name value pairs, and values are instances of predefined primitive types such as integers. Filters are conjunctions of attribute filters, and constraints include comparisons of the predefined simple values or strings. Routing strategies in SIENA use two classes of algorithms: advertisement forwarding and subscription forwarding. The advertisements inform the event notification service. Every advertisement floods the whole network, and a tree is formed covering all servers in the system. Upon receiving a subscription, the server propagates the subscription using a reverse path of all advertisers that are relevant to the subscription. Events are forwarded through active paths. In subscription forwarding, on the other hand, a tree is formed based on

routing paths sent by subscriptions. SIENA exploits routing advantages in combinations of both subscriptions and advertisements. They prune the tree by propagating only paths that are not covered by the previous operation. Gryphon uses a content-based matching algorithm [ASS+99]. It creates a hierarchical tree from publishers to subscribers and maps content-based publish/subscribe to network level multicast.

The P2P communication paradigm is becoming popular for sharing and exchanging information among distributed peers. Advanced P2P systems form a structured overlay by peers and provides more efficient lookup services. They use Distributed Hash Table (DHT) functionalities. CAN [RFH+01], Chord [SMLN+04] and Pastry [RD01] are key examples of such overlays. Such systems provide locating information from exact id (e.g., the resource name). Various systems extend this simple functionality to more complex queries [GAA03] [SGAA04]. Many subject-based publish/subscribe systems are implemented using an application layer multicast through DHT overlays.

2.2.4 Application Level Multicast

DVMRP (Distance Vector Multicast Routing Protocol) [R+99] emerged as the first multicast routing protocol in the 1980s. Today, many multicast applications use IP multicast on MBONE, the multicast-capable virtual network layered over the Internet. Multicast is an important transport for messaging systems, because it minimises link bandwidth consumption, router processing, and delivery delay. However, today's multicast schemes are not scalable to support large numbers of distinct multicast groups. Thus, many Application-Level Multicast Routing Protocols (ALMRPs) have been developed, and the majority use tree routing to get logarithmic scaling behaviour with respect to the number of receivers. These protocols put an unbalanced load on the nodes in networks, because most hosts only receive messages, while routers forward messages to peers. These routers have to look at the content of messages to decide on what links to forward. For good performance, robustness, and scalability, it is important to assign dependable hosts for routers. Fig. 2.5 depicts the architectural difference between IP multicast and ALPRPs.

Narada [Nar02] [ALJ02] and many subsequent designs can best be understood as two layered protocols: a protocol that maintains a mesh of hosts and a multicast routing protocol that builds a tree on top of this mesh. Every node has full knowledge of every other node, so in Narada robustness is an advantage, while only small groups are targeted.

Other examples are Bayeux/Tapestry [Z+01], Scribe/Pastry [CDK+02], SelectCast [CRS01], and CAN [RHK+01]. The multicast service model is less powerful than that of a content-based network, and there is currently no optimal way of using or adapting the multicast routing infrastructure to provide a content-based service.

Hermes [PB02] [PB03] is a type-based publish/subscribe system using a similar rendezvous mech-

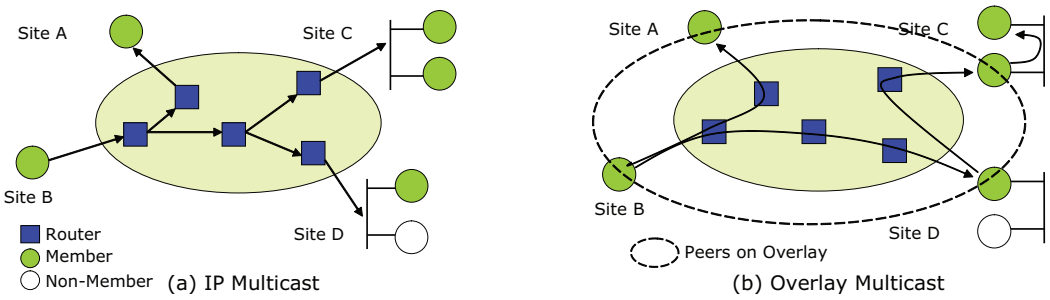


Figure 2.5: IP Multicast and Overlay Multicast

anism to Scribe. Content-based functionalities are added using a similar mechanism to SIENA.

[TAJ03] presents an approach to build a distributed content-based publish/subscribe system on top of a topic based publish/subscribe relying on a DHT infrastructure. It maps subscriptions based on the database schema to topics. [RLW+03], on the other hand, determines multicast groups from clustered subscribers, and matching events are performed against multicast groups. This approach considers a static scenario in which the subscribers join the multicast group, and each multicast group already has the pattern of subscriptions. [PWR04] proposes a solution with specific content being described using attributes and events and subscriptions being described using predefined attributes. These solutions are fundamentally inflexible for content-based publish/subscribe; they are based on a DHT-based routing mechanism. Meghdoot [GSAA04] is a scalable content-based publish-subscribe system over DHT-based on CAN [RFH+01]. Subscription and event routing exploit CAN routing algorithms. However, the service models of multicast and topic-based publish/subscribe is less powerful than that of a content-based network, and there is no optimal way for using or adapting the multicast routing infrastructure to provide a content-based service.

2.2.5 Event Structure

In SIENA, an event is described with simple name and value pairs, while Hermes defines an event type and attribute in an XML schema. In type-based publish/subscribe[EFGS00], the event type model integrates with the type model of an object-oriented programming language.

In a system with large numbers of brokers, events and subscribers, it is necessary to develop efficient data structure and algorithms for subscription propagation and event matching/filtering. Semantically summarising subscriptions would be one approach, which saves network bandwidth and processing cycles for matching.

The subscription can be summarised at the attribute level by a *subsumption* [BPS94] mechanism among attribute values. The frequency of subsumption therefore makes data structure more compact. A distributed scheme for propagating, merging, and updating the subscription summaries in a network of brokers is useful. Each node summarises the received summary from its neighbours, adds its own subscription summary, and propagates the result to other neighbours. It can create a summary in a hierarchical structure among nodes.

Another approach is translating a content-based subscription to a topic-based subscription and constructing a topic hierarchy. After the dissemination tree is created, it can simply use multicast mechanisms. To automate building such topic trees, the content provided by the application follow some constraints (i.e., *template*). This enables the building of a content-based distributed P2P DHT-based publish/subscribe system over a topic-based system. The optimal set of indices for an application domain depends on how well the *template* is defined for query trends.

2.3 Publish/Subscribe in Wireless Networks

Most research focuses on integrating ad hoc networks into P2P networks, but this will not support mobile computing in its generality. The decentralised event-based middleware systems are still immature and fragmented; integrating a mobile environment requires an architecture of event-based middleware on both wired and mobile networks from a unified viewpoint [CN01] [CNF98a] [MCE02]. Computing devices are becoming increasingly mobile. Mobile computing environments need to deal with more dynamic environments and more resource constraints. This diversity of clients creates even more complex environments in distributed systems, and the middleware communication service is important for integrating hybrid environments into coherent distributed

systems. It is especially difficult for event-based middleware to support dynamic reconfiguration of the topology of distributed dissemination infrastructure.

2.3.1 Wireless Mobile Computing

Mobile computing brought different aspects to traditional middleware support. Systems supporting mobility have to react to frequent changes in the environment due to the movement of mobile devices and users. Moreover, the infrastructure offered by the environment must detect the appearance and disappearance of devices and services adding and removing functionality. Middleware for mobile computing needs to consider the following issues:

- Light Computational Load: Mobile applications and middleware run on resource constrained devices.
- Adaptivity: Mobile systems operate in an extremely dynamic context and need to adapt to the given environments. For example, bandwidth may not be stable or services that are available at a particular moment may not be there a second later.
- Disconnected operation: Because of low bandwidth, high latency, and frequent disconnections, a middleware should provide an interface to applications that allows the maintenance of communication during disconnected operation. Device locations continually change, which causes logical disconnection from the service and context change.
- Data model: A data source can be interpreted in different formats and semantics depending on the specifications of mobile devices and wireless networks.
- Communication abstraction: There are various carriers such as 2G, 2.5G, 3G, 4G, Bluetooth, and IEEE 802.11, and many devices are non-programmable. A middleware needs to offer an interface that provides a communication abstraction.
- Ad hoc networks: a feature of some mobile/wireless networks is a dynamically reconfigurable network without a fixed infrastructure that does not require intervention of a centralised access point. Ad hoc networks hold different levels of security from fixed networks. Such networks can operate in either a stand-alone fashion or can be connected to the Internet.

Asynchronous communication is essential for supporting MANET environments. The most popular messaging model, publish/subscribe, maps well onto a decentralised group structure in MANETs. The messaging system in a MANET should be self organised, because the topology of a mobile P2P system has to constantly adjust itself by discovering new communication links and also needs to be fully decentralised due to the lack of a central access point.

Mobile computing is a dynamic distributed system where links between network nodes change dynamically. Based on lower-layer discovery protocols, these devices automatically detect others and spontaneously form ad hoc communities. The majority of research projects consider packet forwarding in an ad hoc routed infrastructure. The sources and destination of messages will never be directly connected with each other. A store-and-forward broadcasting technique will play an important role.

A mobile phone is a good example of a wireless communication tool with future potential in ubiquitous computing. Mobility and coverage superior to other technologies made mobile phones become popular.

Building a simple and efficient data communication network by integrating IP technology is becoming popular. The bandwidth is 100 $Kbps$ for upload and 1 $Mbps$ for download. There will be many wireless hot-spots on streets, offices, trains, resorts, etc. These hot-spots support

IP networking. Therefore, distributed mobile hosts can potentially obtain Internet connectivity anytime and anywhere.

Traditional nomadic distributed systems, based on core fixed routers, switches and hosts is changing. They support mobile hosts through base stations with wireless communication capabilities and are evolving into more flexible forms.

In the Internet Engineering Task Force (IETF), the working group for Mobile Ad Hoc Networking (MANET) has recently made steps towards standardising new routing protocols called DYMO [CBRP05] (for Dynamic Mobile Networks). However, the majority of MANET research focuses on pure ad hoc environments that exist in the real world. On the other hand, Wireless grids [McK03] form new types of networks using mobile, nomadic, and fixed wireless devices that connect sensors, mobile phones, and other edge devices with each other. Wireless grids can be fixed or dynamic, forming ad hoc networks, and offer a wide variety of applications. Recently wireless mesh networks (WMNs) [AWW05] have emerged. In WMNs, nodes comprise mesh routers and mesh clients forming dynamically self-organised and self-configured ad hoc networks. Mesh networks may involve either fixed or mobile devices. DTN [JFP04] [CHC+05] routing accounts for opportunistic connectivity. Network storage allows DTN nodes to buffer data bundles until connections are available. Thus, the integrated network architecture allows automated transfer of pending bundles once connections are re-established. DTN is asynchronous store-and-forward message delivery and supports multi-hop delivery of data. Routing accounts for buffer management, allowing selection of the best next hop based on buffer availability and proximity to the packet path. Furthermore, sensors will be attached to the human body creating a Personal Area Network (PAN). This recent emergence of new hybrid wireless networks leads us to design an abstraction of semantic data dissemination mechanisms integrating an ontology-based event model and event correlation.

2.3.2 Multicast in Ad Hoc Networks

For wireless networks, the most natural communication type is broadcasting. However, the problems in ad hoc wireless networks' multicasting are mobility of sources, destinations and intermediate nodes in the distribution tree. The dynamic topology of the network raises challenges for the maintenance of a multicast group; such as providing better bandwidth utilisation, reduced host/router processing, and resolving unknown receiver addresses.

MANET routing protocols can be classified into the following categories (see also [RT99]):

- **Proactive (table driven):** each forwarding address is kept in a table to maintain consistent up-to-date routing information. Nodes respond to network topology changes by propagating router updates through the network. Maintaining routes at all times may cause high overhead. An example is Ad Hoc On-Demand Distance Vector (AODV) [P+02].
- **Reactive (on-demand):** when a source node requires a route to a destination, it initiates a route discovery process within the network. This approach may keep traffic low without maintaining routes, but it causes delay in route determination. An example is Optimised Link State Routing (OLSR) [CJ03].
- **Hybrid:** combination of the above. For example, Zone Routing Protocol (ZRP) [HPS02] uses proactive methods for nodes within n-hops and is reactive for the other cases.

Examples of existing multicast routing protocols for ad hoc wireless networks are:

- Source-Based Tree: maintains a per-source multicast tree from each source to every multicast member. (e.g., Distance Vector Multicast Routing Protocol (DVMRP) [M+00], Mul-

ticast Routing Extensions for OSPF (MOSPF) [M+00]).

- Core-Based Tree: spans a single multicast tree covering every multicast member (e.g., Multicast Ad hoc On-Demand Distance Vector (MAODV) [R+99], Ad hoc Multicast Routing (AMRoute) [BLMT99]).
- Multicast Mesh: creates a multicast mesh instead of a tree. A mesh reduces the congestion problem in the core-based tree, since in a mesh there are multiple routes available. (e.g., Core Assisted Mesh Protocol (CAMP)[Gar99], On-Demand Associativity-Based Multicast (ARBM) [TGB00]).
- Location-Based Forwarding: uses the concept of physical/spatial regions, namely the forwarding region and the multicast region (e.g., Location Based Multicast (LBM)[KV99]).

2.3.3 Publish/Subscribe in Wireless Networks

Most publish/subscribe systems have focused on systems where nodes do not move and broker networks remain fixed. Fixed event dissemination structures are not suitable for applications in mobile environments, where physical network topology and node locations change continuously.

Most event-based middleware for wireless networks consider that mobile nodes connect to applications in wired networks through the wireless networks. Examples are [CNP00] [SAS01] [FGKZ03] [CIP02] and [CSZ03]. JEDI offers *moveOut* and *moveIn* operations that enable subscribers to disconnect and reconnect to a different network, requiring applications to explicitly call those operations [CNP00]. Similarly, the mobility extension of SIENA requires an explicit request from applications [CIP02]. Elvin supports disconnection and reconnection using central caching proxies [SAS01]. [FGKZ03] extends Rebeca to support mobile and location-dependent applications by transparently rebinding a client to different brokers and offering a fine-grained control over notification delivery in the form of location-dependent filters, but it does not target ad hoc wireless networks. For more details on publish/subscribe in mobile ad hoc networks, see Chapter 6.

Several middleware systems have been developed to support wireless ad hoc network environments (e.g., STEAM [Mei02] and IBM WebSphere MQ [IBM03]). STEAM provides proximity-based group communication. These systems construct publish/subscribe on top of existing transport protocols.

2.4 Distributed Stream Data Processing

Stream Data Processing shares many problems with publish/subscribe systems. In many applications such as network management, stock analysis, and Internet-scale news filtering and dissemination, data arrives in a stream. Examples are news-feed data and continuously arriving measurements from sensors. Stream processing systems typically support numbers of continuous queries. Stock analysts continuously monitor incoming stock quotes to discover matching patterns. Classic database systems are optimised for one-shot queries over persistent datasets, and solutions based on database triggers do not scale for stream processing.

In the database community, TelegraphCQ [CCD+03], Aurora [ACC+03], Borealis [AAB+05], and STREAM [ABB+03] have progressed in providing support for stream data manipulation from a database-centric perspective. Aurora manages continuous data stream for monitoring applications; it focuses on DBMS-like support for applications. Borealis extends Aurora by adding dynamic revision of query results and dynamic query modification. Similarly, the STREAM project [ABB+03] views stream processing as the running of continuous queries expressed in a query language (CQL) that includes sliding windows and sampling over the data stream. Queries

are converted into an execution plan that includes stream-related operators, data queues, and data stores that manage the sliding windows and samples over data that recently passed through the stream.

Cayuga [DGR04] is a knowledge broker system supporting continuous queries over persistent datasets and data streams. Core functions are a subscription matching engine and data mining modules. The subscription matching engine deals with continuous queries and extends publish/ subscribe functionality and multi-query techniques to support detection of temporal patterns in data streams. Stream tuples, partial results of queries from the subscription matching engine and data mining modules are stored in the archive database. The data mining modules provide a query for the archive data. The search engine, database, and mining modules are well integrated to enable automatic subscription setting for monitoring incoming data streams.

2.5 Multidimensional Range Query

In database systems, multidimensional range query is solved using indexing techniques, and indices are mostly centralised. Recently distributed indexing is becoming popular, especially in the context of P2P and sensor networks. Indexing techniques tradeoff data insertion cost against efficient querying. The classical indexing structures are data-dependent, using locality preserving hashes [IMRV97] [GIM99] [IM98].

In [LKGH03], a multi-key constant branching index structure based on k-d trees [Ben75] is presented for geographic embedded sensors, where k indicates the dimension of the data space. This is related to spatial indexing systems [Sam95] [FB74] [Gut84]. The approach is similar to CAN, which constructs a zone-based overlay above the underlying physical network. CAN's overlay is purely logical, while the overlay described in [LKGH03] is consistent with the underlying physical topology. [MFH+03] also describes a distributed index Semantic Routing Tree (SRT) that is used to direct queries to nodes keeping relevant data.

Distributed indices exist that are restricted to exact match or partial prefix match queries such as DHT systems. [AS03] and [HHH+02] support range queries in DHT systems. However, if a locality-preserving hash to store data is used to enable efficient multi-dimensional range queries, load balancing will be curtailed (see Sections 5.4.1 and 5.4.4 for details).

Data centric storage [LKGH03] systems include geographic hash-tables (GHTs) [RKY+02], DIMENSIONS [GEH02], and DIFS [GEG+03]. Geographical Hash Table (GHT) is based on a geographical hashing function [RKS+03], where nodes are assumed to know their own locations. Hashing is based on geographic data on the nodes, and all the values on location will be stored at the rendezvous node responsible for that specific location. The advantage of this system is that it allows lookup of the location of data. GHT uses GPSR for Routing (Greedy Perimeter Stateless Routing). Data objects are associated with keys, and each node in the system is responsible for storing a range of keys. When a node does not know the exact destination, it can still send a packet towards a region where an accessible home node should exist. DIMENSIONS focuses on sensor networks.

2.6 Event Correlation

In event-based distributed systems, composite events represent complex patterns of activity from distributed sources. Although composite events have been a useful modelling tool in active database research and telecommunications network monitoring, little progress has been made in using them in large-scale, general-purpose distributed systems.

Much composite event detection work has been done in active database research. COMPOSE [GSAA04] provides expressive composite event operators similar to regular expressions and implements it using Finite State Automata (FSA). SAMOS [GD94] uses Petri nets, in which event instances are associated with parameter-value pairs. An early language for composite events follows the Event-Condition-Action (ECA) model and resembles database query algebras with an expressive syntax. Snoop [CM96] is an event specification language for active databases, which informally defines event contexts. The detection mechanism in Snoop is based on trees that expresses the composite events. Instances of the primitive events are inserted at the leaves.

The transition from centralised to distributed systems led to the need to deal with time. [CM96] presents an event-based model for specifying timing constraints and to process both asynchronous and synchronous monitoring of real-time constraints. Various event operators have been used for defining composite events. However, composite events specified with these operators are often interpreted differently in terms of their occurrences. [LMK98] proposes an approach that uses the occurrence time of various event instances for time constraint specification. GEM [MSS97] allows additional conditions, including timing constraints, to combine with event operators for composite event specification.

Event-based systems provide a way to design large-scale distributed applications and require event detection as a middleware functionality to monitor complex systems. In [MA02], formalised schema for composite event detection including the operations and event contexts have been defined for real-time systems. This system aims to ensure the resource bounds for event detection in resource constrained environments.

In event-based middleware, publish/subscribe can provide subscription to composite events instead of leaving it to the client to subscribe to and correlate multiple primitive events. This reduces communication within the system and potentially gives higher overall efficiency [PSB04]. In [Hay96], composite events in the Cambridge Event Architecture [BMB+00] are described as an object-oriented system with an event algebra that is implemented by nested push-down Finite State Automata (FSA) to handle parameterised events. [JH04] analyses different event composition languages and attempts to create common event notification services, but it does not consider time constraints. Thus far, few of these event notifications address time and resource constraint issues.

2.7 Summary and Outlook

This chapter has outlined related research areas (publish/subscribe systems, overlay multicast, and mobile wireless network communications), discussing current trends and future directions. It also surveys recent activities in the areas of stream processing, multidimensional query, and event correlation. This points out the importance of data processing over heterogeneous networks and its incorporation with communication facilities.

The next chapter describes our future vision and environments for ubiquitous computing. Chapters 4 to 7 then focus on specific subjects that exploit important issues for the future vision described in Chapter 2 and 3.

Chapter 4 discusses an event and query model from the event-based middleware perspective and presents a multidimensional event indexing with *Hypercube*. The use of *Hypercube* event indexing is demonstrated in a typed content-based publish/subscribe system (i.e., Hermes) in Chapter 5. Chapter 6 presents structureless asynchronous group communication over hybrid wireless ad hoc networks (i.e., publish/subscribe in wireless ad hoc networks). Unified event correlation semantics and adaptation to mobile devices are presented in Chapter 7.

3

Ubiquitous Computing

Wireless sensors and mobile devices are transforming the real world into a computing platform. Numerous devices now have computing power, and emerging networking techniques will ensure that devices are interconnected from tiny sensor networks to wide area networks. Heterogeneous collections of devices will interact with each other in our homes, offices and transportation systems, which will form an intelligent pervasive space in an ad hoc fashion. People will interact with invisible, ambient technology that is usable by non-experts. The resulting data will provide valuable information. For example, early discovery of health problems via sensed body data could save lives. To prevent disasters, sensor networks could be used to detect distortion and structural problems in buildings. Sensors in urban environments could be used for traffic monitoring to prevent and solve congestion. Ubiquitous computing requires dynamic formations of devices in an ad hoc mode. The individual components are heterogeneous, and it is complex to coordinate activities to achieve a goal. The interaction between components is usually carefully designed and manually programmed for sensor network applications. This has to change so that WSNs can organise themselves from components built by different applications. Programming abstractions need to be addressed to support abstractions of sensors and sensor data.

In many applications, the large volume of high-speed data streams makes storage and data processing impossible. Traditional stream processing needs distributed processing, where continuous queries may be located throughout the networks. Processing power on devices enables query processing to move directly to the data sources. Query processing strategies need to balance resource usage and expressiveness.

Applications are increasingly decentralised and distributed over various types of device that form dynamic networks and interact in an event-driven mode. This requires a new generation of middleware that can dynamically exchange data. Publish/subscribe systems need to consider large scale global computing, integrating scattered WSNs at the edge of wired networks. A concept of service must be introduced for ubiquitous computing.

Services can address network software entities and provide them to the users. This includes grid services, information services, network services, web services, messaging services and so forth.

This chapter discusses environments for ubiquitous computing and service oriented architecture

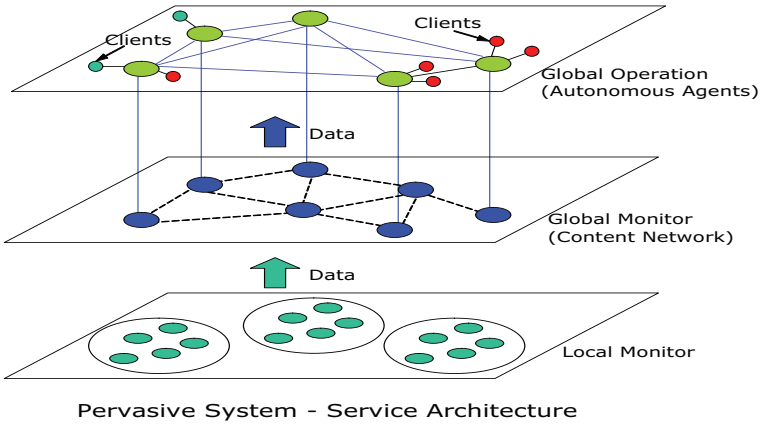


Figure 3.1: Service Overlay

(SOA). A SOA-based approach is a standardised way to address interoperability and development support for a wide range of distributed applications in heterogeneous environments. We have done initial research on SOA-based middleware (see [Yon05] [YB05a] and [YB06]). This chapter outlines our approach for future WSN-integrated ubiquitous computing, providing a vision for the next generation of event-based middleware.

3.1 Global System

The Internet along with computer hardware/software made large-scale distributed computing possible. The evolution of ubiquitous computing will add an additional dimension. Various heterogeneous sources produce information, and reactive systems make decisions based on this information. Instead of looking at individual data, a global view on information and knowledge fusion is critical. A global view results in more robust, flexible, and scalable systems. Sensed data need to provide pervasive access through a variety of wireless networks. There are inherent resource limitations in the technologies for processing, storage, and communication, leading to novel systems performance requirements. A new platform needs to cover the entire range from tiny MEMS to Internet scale P2P systems and must include not only quantitative performance but also quality of service.

A reactive system incorporating sensing, decision making and acting will be a common application characteristic. Ultimately, the architecture must be an open and component-based structure that is configurable and self-adaptive. A Web service based grid architecture is static and cannot support these diverse subsystems (e.g., ad hoc environments, local clusters, the global Internet) and the bridges that enable them to interoperate. Service broker grids based on service management are a recent trend in system architecture to support such platforms. A difficult issue here is that applications are tied to sensor deployments (see [YB05b] for more detail). We need to develop a new type of open platform, where sensed data can be shared among different applications over large-scale environments. Data management over such heterogeneous networks and social environments will be crucial.

WSNs have different functional components: detection and data collection, signal processing, data aggregation, and notification. Integration of these functions makes a WSN into a platform for hierarchical information processing. Middleware in ubiquitous computing must allow information to be integrated at different levels of abstraction, including detailed microscopic examination of specific targets, detailed views of aggregated target behaviour, and answers to

queries from end users. Any events in the environment should be processed on three levels: node, local neighbourhood, and global levels. Fig. 3.1 depicts the view of this three tier overlay. These three levels of data processing should be integrated within the underlying communication mechanism.

3.1.1 WSN Middleware

The majority of current middleware for WSNs is based on database management systems. The database community has taken the view that declarative programming, through a query language, provides the right level of abstraction for accessing, filtering, and processing relational data. Middleware that takes a database approach such as [MFH+02] provides an interface for data collection but does not provide general-purpose distributed computation. Thus, more general interfaces for global network programming are desirable. Middleware will acquire more programming functionality, propagation of code, self maintenance, and state maintenance to adapt to dynamic environments. In ubiquitous computing systems, data fragments are dispersed among various devices, services and agents, and the level of this data can differ. Yet knowledge must be represented for computations, whether the focus is the data itself or their application. When designing middleware for sensor networks, heterogeneity of information over global distributed systems must be considered, and the information sensed by the devices must be aggregated and combined into higher-level information or knowledge that will ultimately be delivered to the subscribers.

3.1.2 Grids and P2P

P2P networks and grids offer promising paradigms for developing efficient distributed systems. The grid community have initiated alignment of grid technologies with Web Services: the Open Grid Services Architecture (OGSA) [Gro] integrates services and resources over distributed, heterogeneous, and dynamic environments. The OGSA model uses the Web Services Description Language (WSDL) [W3C03] to define grid services, which take advantage from both the grid and Web Services. The architecture includes the definition of grid creation, naming, and discovery of grid-service instances. The convergence of P2P and Grid computing is a natural outcome of the recent evolution in distributed systems, because many of the challenging standard issues are closely related. This creates best practice that enables interoperability between computing and networking systems for the P2P community.

3.2 Service Oriented Architecture

There have been efforts to architect middleware for ubiquitous computing environments using SOA based on component interaction mechanisms [FHM+04]. SOA is a well-proven concept for distributed computing environments. It decomposes applications, data, and middleware into reusable services that can be flexibly combined in a loosely coupled manner. SOA maintains agents that act as software services performing well-defined operations. This paradigm enables users to be concerned only with the operational description of the service. All services have a network addressable interface and communicate via standard protocols and data formats (i.e., messages). SOA can deal with aspects of heterogeneity, mobility and adaptation, and offers seamless integration of wired and wireless environments.

Generic service elements are context model, trust and privacy, mobile data management, configuration, service discovery, and event notification. The following key issues are addressed in the proposed design:

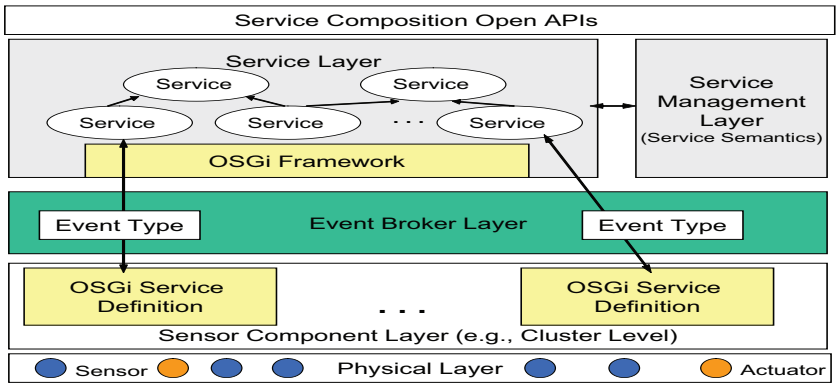


Figure 3.2: Service Oriented Architecture Overview

- Support for service discovery mechanisms (e.g., new services) for ad hoc networks.
- Support for an adaptive abstract communication model (i.e., event-based communication).

Typical service discovery architectures are realised with a dedicated directory agent that keeps all information of services and a set of protocols that allow services to find and register with a directory agent, and a naming convention for services. Service Location Protocol (SLP), Jini, Web Service Description Language, and UPnP are in this category. The Semantic Web addresses service description and discovery mechanisms using DAML. The Open Mobile Alliance addresses global service discovery for wireless networks.

I propose a generic reference architecture applicable to a common ubiquitous computing space. The middleware contains separate physical and sensor components, an event broker, services, and service management layers with an open application interface. Components need to satisfy certain non-functional properties, such as performance, reliability or security, and these components may be used to build distributed applications and services for intelligent networks. A service is an interesting concept to be applied in WSNs. It may be a role on a sensor node, or a function providing location information. Services may be cascaded without previous knowledge of each other. They enable the solution of complex tasks, where functional blocks are separated to increase flexibility and enhance scalability of sensor network node functions.

A key issue is to separate the software from the underlying hardware and to divide the software into functional blocks with proper size and functionality. Another important issue is that the sensed data need to be filtered, correlated, and managed at the right time and place when they flow over heterogeneous network environments. An overview is shown in Fig. 3.2.

The Open Services Gateway Initiative (OSGi) [OSG] is integrated with the application layer instead of creating a new architecture. The OSGi is open to any protocol, transport, or device layers. The key aspects of the OSGi are wide area networks, multiple services, and local networks and devices. Benefits are independence from platforms and applications. Thus, the OSGi specifies an open, independent technology, which can link diverse devices in a local home network. The service gateway is a central component of OSGi specification efforts.It enables, consolidates and manages voice, data, Internet, and multimedia communications to and from different locations.

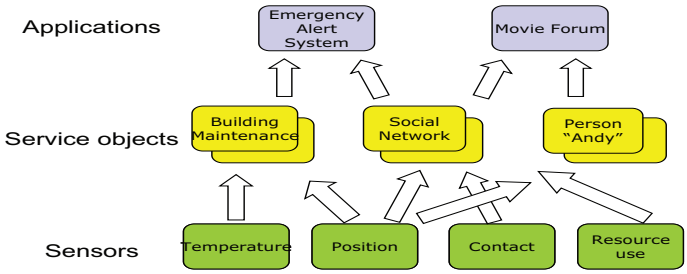


Figure 3.3: Mapping Service to Real World

3.2.1 Service Semantics

Service semantics is important, in addition to service definition, to coordinate services in space. The model of the real world is of a collection of objects, where objects maintain state using sensor data, and applications' queries and subscriptions are relevant sets of objects. Fig. 3.3 shows an example of object mappings among applications, middleware and sensor components. Objects are tightly linked to event types in an event broker. Exploiting semantics lets the pervasive space's functionality and behaviour develop and evolve. Domain specific ontologies enable such exploitation of knowledge and semantics in ubiquitous computing. This allows non-experts to use the systems.

3.2.2 Layer Functionality

The following list briefly describes the functionality of each layer depicted in Fig. 3.2.

Physical Layer: This layer consists of various sensors and actuators.

Sensor Component Layer: This layer can communicate with a wide variety of devices, sensors, actuators, and gateways and represent them to the rest of the middleware in a uniform way. A sensor component converts any sensor or actuator in *PhysicalLayer* to a service that can be programmed or composed into other services. Users can thus define services without having to understand the physical world. Decoupling sensors and actuators from sensor platforms ensures openness and makes it possible to introduce new technology as it becomes available.

Event Broker Layer: This layer is a communication layer between *SensorComponentLayer* and *ServiceLayer*. It supports publish/subscribe based asynchronous communication. Event filtering, aggregation, and correlation services are part of this layer.

Service Layer: This layer contains the Open Services Gateway Initiative (OSGi) framework, which maintains leases of activated services. Basic services represent the physical world through sensor platforms, which store service bundle definitions for any sensor or actuator represented in the OSGi framework. A sensor component registers itself with *ServiceLayer* by sending its OSGi service definition. Application developers create composite services by applying *ServiceManagementLayer*'s functions to search existing services and use other services to compose new OSGi services. Canned services, which may be globally useful, could create a standard library. A context is represented as an OSGi service composition, where it can be obtained. The context engine is responsible for detecting and managing states.

Service Management Layer: This layer contains the ontology of the various services offered and the appliances and devices connected to the system. Service advertisement and discovery use service definitions and semantics to register or discover a service. Service definitions are tightly

related to the event types used for communication in *EventBrokerLayer* including composite formats. The reasoning engine determines whether certain composite services are available.

Application Interface: An application interface provides open interfaces for applications to manage services including contexts.

3.2.3 The Underlying Communication Mechanism

The publish/subscribe paradigm fits well with the emerging SOA, where a distributed application is built using loosely coupled, reusable services. In existing commercial-based SOA architectures, an *EnterpriseServiceBus* (ESB) is provided (e.g., IBM Websphere [IBM00]). The creation of an event broker grid can be easily integrated into SOA. The grid consists of event brokers, and a broker performs the routing, receiving and sending of events. Brokers can form a group to provide scalability at the cluster level; a group of brokers can then be linked together in a flexible, fault-tolerant, efficient, high-performance fashion in the publish/subscribe model. Dynamic grid formation is essential, including context-awareness and an infrastructure such as hierarchy and grouping for better performance.

EventBrokerLayer has an important task, to integrate publish/subscribe systems for various devices under a unified interface. Event brokers can be on mobile devices over mobile ad hoc networks to support data sharing among roaming peers and exploit peer resources if possible. Mobile devices usually have limited capabilities and resources, and the indiscriminate use of P2P applications may result in low performance. Context-awareness allows applications to exploit information on the underlying network context to achieve better performance and group organisation. Information such as availability of resources, battery power, services in reach and relative distances can be used to improve the routing structure of the grid, thus reducing the routing overhead. Use of context-awareness and location awareness are strategies to overcome these limitations.

Ubiquitous computing is changing the Internet's architectural basis, which currently primarily expects homogeneous environments, while ubiquitous environments vary (e.g., address, packet, network link, bandwidth, packet header, transport protocol, naming). Although the Internet is already highly distributed, including mobility support in IPv6 [Dee98], an integrated view including different types of ad hoc wireless networks and communication mechanisms has not yet been achieved.

3.3 Application Spaces

There will be a wide range of applications. Application design principles are discussed in detail in [YB05b]. The application scenario of ubiquitous, pervasive, and mobile computing and their purpose differ from fixed network based computing. Ubiquitous computing collaborates with other devices, work stations and networks to accomplish the main task. An example of such an application is a location based service providing specific services based on the user's geographical position. Mobile devices will soon be equipped with GPS devices as standard to provide the precise location of the device. Location information will allow further refined services for subscribers. Location-based services, such as route and traffic information should be popular. Location based advertisement is another service for publisher initiation. Middleware must filter information for potentially millions of users, given their continuous changes of location, profiles (e.g., subscriptions), and the contexts surroundings the user environments. Technologies such as WiFi and 3G mobile phones are offering the infrastructure to realise information systems as ubiquitous information systems, i.e., systems that are accessible from anywhere, at any time,

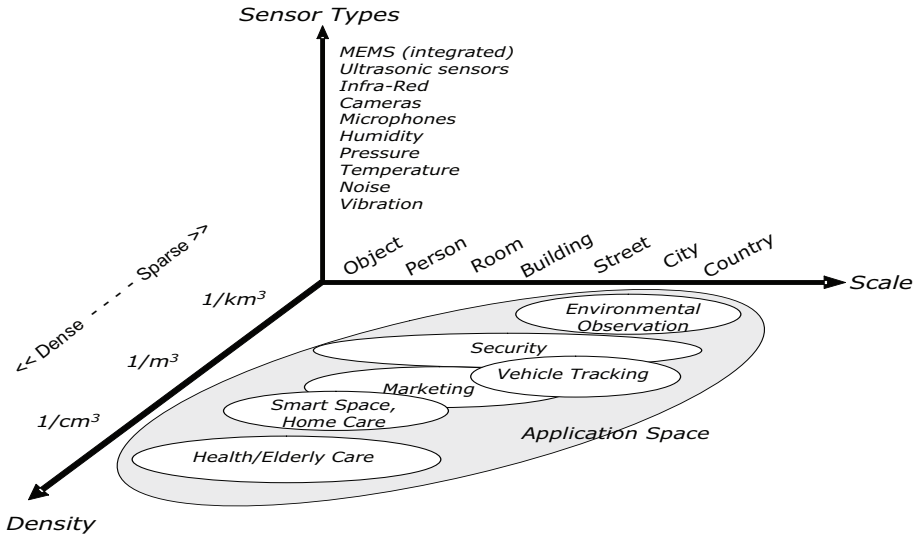


Figure 3.4: Application Spaces

and with any device.

A vision for future ubiquitous computing is that tiny processors and sensors are integrated with everyday objects to construct a smart space. Smart objects can explore their environment, communicate with other smart objects, and interact with humans. Fig. 3.4 shows an application space and potential real applications of distributed WSNs that allow monitoring of a wide variety of environmental phenomena with adequate quality and scale.

3.3.1 Car-to-Car Communication

Car-to-Car (C2C) Communication is potentially a major future application. Ad hoc network technology will prevent accidents by including new safety and communication features. For example, cars involved in an accident can send alert messages to surrounding cars, thus preventing motorway pileups. Roadside sensors can detect potential collisions by measuring car speed and congestion conditions. Ad hoc communication between vehicles can be realised by mobile phones or PDAs belonging to drivers and passengers (e.g., short messaging service (SMS), or hop-by-hop telephony).

3.4 Summary and Outlook

P2P with grids offers a promising paradigm for developing efficient distributed systems and applications and aligning grid technologies with Web Services. Integrated services and resources across distributed environments require a service description language, a data model for naming, and discovery of persistent and transient service instances.

This chapter described the vision of ubiquitous computing and introduced a SOA based architecture. The following four chapters focus on specific topics that explore key issues for the future vision of ubiquitous computing.

4

Event and Query Model

Distributed events are fragmented and dispersed among various devices, services, and agents, and interpretation of events can occur at any time and any location. Events flow based on queries to databases, subscriptions, notifications, and search results over the networks.

We need to look closely into events and find out how their semantics and representation should be defined. Events contain multidimensional attributes and should be well indexed for searching and complex correlation. Besides the existing attributes of events, continuous context information such as time or geographic location will be crucial to incorporate within an event model. The emergence of WSNs produces a huge amount of dynamic data, and to extract meaningful information from these data is even more complex and challenging.

Event modelling poses two challenges. First, different sources may interpret the same event in different ways. Second, the exact condition of the data source has to be clearly defined. The first aspect addresses the semantics of the event, which involves context modelling to translate raw sensor data into a meaningful context. The second aspect is dependent on the data structure of an event, which will be used for search, index, filter and correlation operations. The second aspect is the focus of this chapter.

Traditional databases support multidimensional data indexing and query, including a query language as an extension of SQL. For example, a moving object database can index and query position/time of tracking objects. Applications in ubiquitous computing require such functions over distributed network environments, where data are produced by publishers via event brokers, and the network itself can be considered as a database. The query is usually persistent (i.e., continuous queries). Stream data processing and publish/subscribe systems address similar problems. Ubiquitous computing highlights the need for distributed computing support in stream processing.

In this chapter, I discuss common problems of stream processing, and publish/subscribe systems that lead to a discussion of event modelling. I define a novel event model on primitive events and durative events with a time interval, followed by an analysis of the publish/subscribe model. I then introduce a multidimensional event representation (i.e., *Hypercube* in RTree [Gut84]) for efficient indexing, filtering, and matching. Experiments addressing the filtering capability

of multidimensional *Hypercube* are reported. This is fundamental for events and influences a higher-level event dissemination model.

4.1 Data and Query Characteristics

There are various types of data on the Internet such as unstructured documents, web data, and documents in databases. P2P is currently extensively used for media sharing. Sensor captured data are typically name-attribute pairs, and many of these attributes have scalar values. Sensor data are normally taken after a defined interval to save power. A real-time data stream is a time series of data that arrives in some order. One of the characteristics of stream data is the spatial and temporal information that are annotated to the data (see the following section). Sensor data may be redundant, and after a certain period data may not be valid. A mechanism to discard obsolete data may be necessary, where the data rate is high and dynamic. A system decision is whether all data should be retained and logged.

4.1.1 Spatio-Temporal Events

Unlike traditional database queries, spatio-temporal queries contain changes in both object and query locations over time. For example:

- Moving queries on stationary object: *as I am moving along a certain route, show me all gas stations within 3 minutes of my location.*
- Stationary queries on moving objects: *how many speeding cars are within the city boundary.*
- Moving queries on moving objects; *as the President moves make sure that the number of security guards within 50 metres of his/her location is more than 50.*

In stream processing, a data stream is a continuous and ordered sequence of data values; examples include sensor data [BGS01] [MF02], Internet traffic [GKMS01] [SH98], and transaction logs. It is also real-time and the ordering of data can be performed by arrival time or timestamp. Data stream processing is characterised by continuous queries, which share characteristics with publish/subscribe in ubiquitous computing. This dictates the following issues for data management:

- The data model and query semantics should allow order-based and time-based operations.
- The use of approximate summary structures or digests of the data should be considered, which results in non-exact answers.
- Stream processing algorithms are restricted to make one pass over the data for performance.
- Detection of unusual conditions must operate in real-time.
- Shared execution of multiple continuous queries must be considered for scalability.

Spatial-join can be performed for stationary objects using a simple RTree index [Gut84], which can probe the RTree as range queries. RTree forms a tree structure that is capable of indexing multidimensional information. Therefore it is used for spatial access methods (e.g., geographical data). It is similar to B-trees, and rectangles represent the data spaces, which can be hierarchically nested and overlapping. Search algorithms including intersection, containment, and nearest search use boundary rectangles to decide whether to search further nested rectangles.

4.1.2 Moving Objects

Moving object applications [KSF+03] are special cases of spatio-temporal applications. Numerous moving objects and concurrent continuous queries make it difficult to provide reasonable performance for query processing. Instead of keeping all geographic information of every moving object, the manipulation of changes related to the movement of objects can be used for query processing [PXK+02]. Applications such as transportation, navigation systems, and tracking systems are now available on the Internet. These systems deal with spatio-temporal information that requires new strategies for data modelling, update, and maintenance as well as for querying and access methods. For example, if both objects and queries are moving, spatio-temporal access methods [MGA03] can be used.

Location Data in distributed environments

It is often impractical or impossible to store the location database at a single location, because a centralised architecture could become a performance bottleneck. So the question is how to allocate, update and query trajectories in a geographically distributed environment. Another complication arises when the distributed location database is also mobile. This is the case in MANETs. Such networks provide an attractive and inexpensive alternative when a regular infrastructure is unavailable (e.g., in remote and disaster areas), inefficient, or too expensive to use (see [Haa98]). MANETs are used to communicate among nodes in collaborative mobile data exchange (e.g., the set of attendees at a conference). Each moving object stores and maintains its own trajectory. The challenge is to process the queries with acceptable delay, overhead and accuracy.

4.2 Indexing

Research communities such as P2P, stream processing, and publish/subscribe address data processing from their own views, while many issues are common. The progression of P2P networks and the emergence of ubiquitous computing will force them to meet. For example, the filtering function in publish/subscribe systems is a matching operation between publications and subscriptions, while stream processing is a matching operation between stream data and continuous queries. Both operations can be generalised as either point or range queries against a set of ranged data. Queries are persistent and could be distributed, and the data are typically a form of time series. See more details in the following sections: 5.2 for P2P indexing, 4.3.1 and 7.8 for data stream management, 4.5 for publish/subscribe systems, and 7.4 for wireless sensor networks perspectives.

Most P2P systems support only simple lookup queries. However, many new applications such as photo sharing and online analysis require a multidimensional query, where a query consists of different constraints on various data attributes. Publish/subscribe, WWW search, and many ubiquitous applications all require such multidimensional indexing and query mechanisms. Traditional spatio-temporal database technology can be integrated with P2P routing networks, which is increasingly in popularity.

Location awareness is an important feature of ubiquitous computing. Depending on the current context, users on mobile devices want to be informed about events around or related to them including real world and digital information. Spatial information will be important because of its strong influence on decisions by applications. Spatial events occur, for example, when two people meet. As the underlying world model is distributed, event observation also has to be distributed.

The number of potential spatial events is not restricted. When the spatial world model is distributed and local observation is not sufficient, the observation of events has to be distributed. Observations could be initiated by users or observers by extracting unknown information from the event flow. This raises further complex event observation and notification paradigms. Current event correlation support in publish/subscribe systems is simple compared to the continuous query support in stream data management, where complex spatio-temporal events are correlated. On the other hand, consideration of distributed computing concerns is more advanced in publish/subscribe systems. A large population of users leads to many profiles (i.e., subscriptions) or queries that are continuously evaluated for long-term interests. This requires the creation of more efficient filtering functions for data dissemination. See Sections 7.2 and 7.8 for more details.

Thus, intelligent indexing and data placement must be deployed. One important aspect is the rate of events. A large number of new events or dynamic subscriptions/queries will significantly impact the dissemination mechanism.

4.3 Query and Subscription Languages

Below, I describe characteristics of query/subscription languages in different systems.

4.3.1 Stream Data Management

Query languages for streaming data can be classified into three categories: relation-based, object-based, and procedural.

Relation-based Languages

Continuous Query Language (CQL) [ABW02] [MWA+03], StreaQuel [CCD+03], and AQuery [LS03] are examples of relation-based languages. They exploit SQL-like syntax with support for windows and ordering. CQL is used in the STREAM system [ABB+03], which considers streams and windows to be relations ordered by timestamps. StreaQuel, the query language used in TelegraphCQ [CCD+03], also has advanced windowing capabilities (see Section 7.4 for more details). A query consists of query algebra and an SQL-like language is supported. Table columns in the database are considered as arrays and order-dependent operators can be applied (e.g., next, previous, first, and last).

Object-based Languages

The Tribeca network monitoring system [SH98] classifies stream elements based on a type hierarchy so that the stream is modelled in an object-oriented manner. An alternative approach is to model the sources as Abstract Data Types (ADTs), as used in the COUGAR sensor database [BGS00]. Each type of sensor is mapped to an ADT and its interface includes the sensor's signal processing methods. The syntax of the query language is based on SQL.

Procedural Languages

Instead of using declarative query languages another approach is to specify the data flow. The Aurora system [CCC+02] exploits a procedural language, where users build a query plan for the data flow that may be later optimised by the system.

4.3.2 Publish/Subscribe Systems

Similar to database query languages, publish/subscribe systems normally filter events against filters created from subscriptions. However, complexity in time and space requires more sophis-

ticated composition languages. Filtering is used to select events from a single event type, and correlation is processed among different event types.

4.3.3 Expressiveness and Performance

In publish/subscribe systems, high scalability is normally achieved by reducing expressiveness of subscriptions or mapping complex filters to simple topics, so that large numbers of publishers and subscribers can be deployed. On the other hand, data stream processing systems are equipped with an expressive query language like STREAM [ABB+03] but they do not scale to large numbers of subscriptions.

4.3.4 Discussion

In publish/subscribe systems [ASS+99] [YSG03] [FJL+01], the focus is on achieving high performance instead of expressiveness. There are systems to filter streamed XML documents [DAF+03] [CFGR02] [GAA03]. Their query languages are usually fragments of XPath, which is actually more expressive than the languages provided in publish/subscribe systems. Currently, XML filtering systems do not address parametrisation. Thus, they are not capable of processing subscriptions over multiple XML documents.

Spatio-temporal data management has recently received a lot of attention, mainly due to the emergence of location based services and advances in GPS-capable devices or ubiquitous mobile networks. As a result, large amounts of spatio-temporal data are produced daily, typically in the form of trajectories. Cars moving on the highway system is an example. The common characteristic is that spatio-temporal objects move and/or change their states over time. The need to efficiently analyse and query this data requires the development of sophisticated techniques. Previous research has dealt with various spatio-temporal queries, focusing on range searches and nearest neighbour variations or mining tasks like extracting patterns and periodicities from spatio-temporal trajectories. However, pattern queries on spatio-temporal trajectories such as *Locate rental cars that left Heathrow airport a week ago and were parked in one of the carparks near Dover* may not be solved by range filtering. Thus, the correlation of multiple events must be addressed together. Event correlation is discussed in Chapter 7.

Cayuga [Pro05] provides support for parameterised composite events and aggregate subscriptions. It focuses on multi-query optimisation by applying a combination of state merging and indexing techniques. Several other groups have built systems with expressive query languages [CCC+02] [MWA+03] [CCD+03] [AAB+05]. The TREPLE language [MZ97b] defines formal language specification based on Datalog. It extends the parameterised composite event specification language of EPL [MZ97a] so that an aggregation mechanism with explicit recursion can be provided. The most established formal approach is STREAM's CQL [MWA+03]. CQL provides SQL-like syntax with window queries. SQL based languages use the unordered data model, which may raise fundamental issues on real-time event detection. Thus, if the query is order-based, it has to verify the event order among the set of events retrieved from the window operation.

In most publish/subscribe systems [LJ05] [LJ03], the focus is on the expressiveness of subscriptions. However, they mainly target performance increase in a distributed setting, and the degree of expressiveness in a query language is limited (e.g., no parametrisation).

Another complex issue is that more sophisticated data models may not yield good query processing due to the user's uncertainty. To deal with uncertainty, fuzzy logic can be used to express queries in a gradual and qualitative way. Current search engines such as *Google* use keyword-based queries to help users to find related information. These traditional tools assume that the

parameters of a query model represent exactly the features of the modelled objects. But some query processes are uncertain in nature and hard to express in the form of traditional query languages. This may be problematic for two reasons: 1) the real situation is vague and a query cannot be described precisely, and 2) a complete description of a query object requires more detailed knowledge than one can expect from non-expert users. This changes the way we issue and model a query.

4.4 Event Model

In the previous sections, I have described the characteristics of events that will be produced in ubiquitous computing scenes. In this section, I aim at modelling events in an unambiguous way to deal with types of events that require integration of multiple continuous attributes (i.e., time, space, etc.). This attempt is fundamental for establishing a common semantics on events, which will become tokens in a ubiquitous computing scene.

Ubiquitous computing aims to link the real and virtual worlds. Events tie the two worlds together, and relevant state changes in the real world have to be detected and signalled to the virtual world by generating appropriate events. Sensors can detect state changes in the real world, but low-level sensor information is often not directly useful. Typically, raw sensor input is therefore processed and combined to form high-level events that model real world actions.

There are two steps in sensor data operations. First, processing sensor data to generate a meaningful event. This involves signal processing and various algorithms such as Bayesian networks, Markov models, rule based, and neural networks. Second, after the generation of primitive events, higher-level information can be computed. Current research prototypes deal with the border between these steps that are specific to sensors and applications.

Defining an unambiguous event model provides a common interface for event correlation. [RM04] reported that a traditional event composition paradigm is not capable of in-network processing of WSNs, but an important issue here is to distinguish raw sensor data and events. Combining events to generate higher-level information raises complex issues. Much research on this has been done in our group [BBH+95] [Hay96] [PSB04].

I consider events and services associated with events to be of prime importance for ubiquitous computing and define semantics of events and instances.An event is a message that is generated by an event source and sent to one or more subscribers. Actual event representation (e.g., data structures) may be a structure encoded in binary, a typed object appropriate to a particular object-oriented language, a set of attribute-value pairs, or XML.

4.4.1 Event

The event concept applies to all levels of events from business actions within a workflow to sensing the air temperature. Primitive and composite events are defined as follows:

Definition 4.1 (Primitive Event) *A primitive event is the occurrence of a state transition at a certain point in time. Each occurrence of an event is called an event instance. The primitive event set contains all primitive events within the system.*

Definition 4.2 (Composite Event) *A composite event is defined by composing primitive or composite events with a set of operators. The universal event set* $\mathbb{E}$ *consists of the set of primitive events* $\mathbb{E}_p$ *and the set of composite events* $\mathbb{E}_c$*.*

The operator $\succ$ *identifies the events that contribute to a composite event.* $\succ$ *is defined: Let*

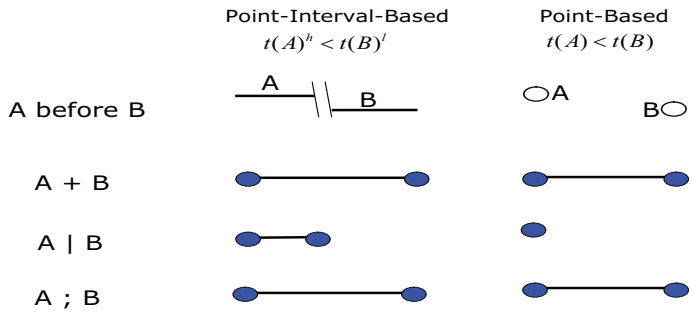


Figure 4.1: Timestamp of Composite Event

$e_1, ..., e_n \in \mathbb{E}$ *be event instances contributing to the composite event* $e \in \mathbb{E}_c$. $\{e_1, ...e_n\} \succ e$ *expresses this relation where* $\{e_1, ...e_n\}$ *can be instances of primitive or composite events.*

4.4.2 Timestamps

Definition 4.3 (Time) *Time in the real world may be continuous, but I define time as discrete and finite with limited precision. It is assumed that time has a fixed origin and equidistant time domain. Absolute time systems (e.g., 15:00 GMT) can map to the time-axis.*

Definition 4.4 (Timestamp) *Each event has a timestamp associated with the occurrence time. There is uncertainty associated with the values of timestamps in implemented systems. A timestamp is a mandatory attribute of an event defined within a time system, while the event occurrence time is the real-time of the occurrence of the event which itself cannot be known precisely. Thus, the timestamp is an approximation of the event occurrence time.*

The accuracy of timestamps depends on the event detection and timestamp model (see Section 7.7.1 for the time model). The stages of event capture are:

- Physically capturing an event, i.e., time of occurrence (T_p).
- Recognising it (by the processor), i.e., time of detection (T_d).
- Obtaining a point-based timestamp (T_g) or an interval-based timestamp (T_g^l, T_g^h), with l for lower bound and h for upper bound.
- Inserting the event in a communication line.

In most cases, it is expected that $|T_d - T_p|$ to be 0. T_g indicates T_p with the available values within the system. Depending on the time system, the timestamp can be point-based (T_g) or interval-based (T_g^l, T_g^h).

Most point-based timestamps consist of a single value indicating the occurrence time. If there is a virtual global clock, each event can have a global time value for timestamping when the event occurred. Granularity and non-zero precision of a virtual global clock cause errors. For example, two distinct events may have the same timestamp value, or if communication is involved, the timestamp of the message sender could be greater than or equal to the timestamp of the corresponding receiver. Thus, adding the margin by calculating the delay time from the communication with the global clock can be represented in either a point-based timestamp or an interval-based timestamp.

In [LCB99], the time when an event is detected is given as an interval-based timestamp, which captures clock uncertainty and network delay with two values: the low and high end of the

interval. Although an interval format is used, it represents a single point (*point-interval-based timestamp*).

Let e be a primitive instance of an event and $t(e)$ the timestamp for it. There are three possibilities in representing timestamps:

- Point-based timestamp: denoted $t_p(e)$.
- Point-interval-based timestamp: denoted $t_{pi}(e)_l^h$. It is a point-based timestamp in an interval-based format. An interval represents error margins from event detection delay, processing time, and network delay.
- Interval-based timestamp: denoted $t_i(e)_l^h$. Composition of events creates an interval. $t_i(e)^l$ and $t_i(e)^h$ themselves may be represented in either point-based or point-interval-based format. Thus, the timestamp could take a two layer structure of interval values.

Composite events are built up from events occurring at different times, and the associated real-time is usually that of the last of its contributory primitive events. This is natural in a context where the prime focus is on event detection, since typically a composite event will be detected at the time that its last contributory event is detected. However, this does lead to logical difficulties in the case of some composite events. A composite event is defined with duration and given a new interval-based timestamp to a composite event based on interval semantics (see Chapter 7). A point-interval-based timestamp is an accurate representation of a primitive event and is distinct from interval-based timestamps representing the duration of composite events. In theory, for a primitive event, either a point-based or point-interval-based timestamp can be used. However, to focus on the interval semantics, the point-based timestamp for a primitive event is used throughout this dissertation.

The duration of composite events depends on composition semantics and the time system. Fig. 4.1 depicts an interval-based timestamp for composite events in the system. For example, the disjunction operation of event A and B (i.e., $A \mid B$) detects A as a result of the event composition, and if a point-based time system is used, the timestamp of event A is maintained as a timestamp of the composite event, while the conjunction operation of A and B (i.e., $A+B$) results in the duration of A and B as a timestamp of the composite event. The sequence operation of event A and B (i.e., $A;B$) results in the same as the conjunction operation.

4.4.3 Spacestamp

Definition 4.5 (Spacestamp) *A spacestamp is an optional attribute of an event, indicating the location of event occurrence. The location can be absolute location, relative location, and grouping (e.g., position information (x,y,z), postcode, global node id). The Global Positioning System (GPS) [HW+94] can provide each node with its location information (latitude, longitude and elevation) with a high degree of accuracy. This information can be used for classifying events within the given space.*

Currently, the semantics of *spacestamp* is dependent on the applications. No global ordering scheme is defined. For example, when *postal codes* are used, the ordering could be simple string order or a specific order derived from the geographic locations. Spacestamp could represent 2-dimensional or 3-dimensional coordinates.

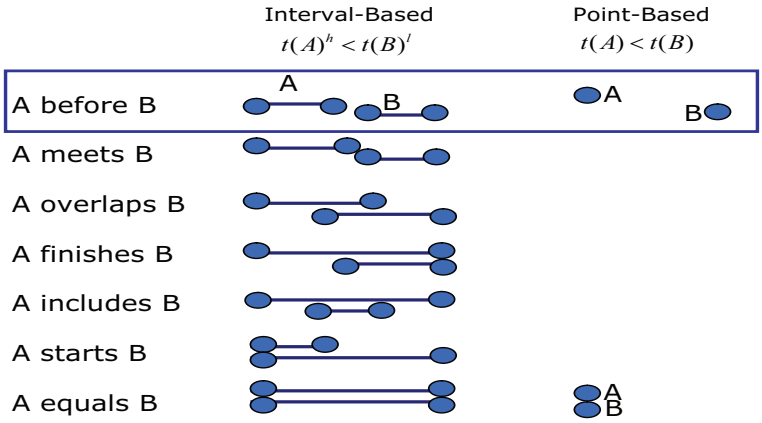


Figure 4.2: Interval and Point based Timestamps over 7 temporal relations

4.4.4 Duration

Definition 4.6 (Durative Event) *Composite events can have duration, where the time of occurrence denotes the duration which binds the event instances in the course of the detection of composite events.*

A durative event can be seen as an abstraction constructed over two primitive events, which binds its occurrence period instead of start-end instantaneous events. Considering the time of occurrence and the time of detection, these two times in primitive events usually coincide. For composite events, however, the time of occurrence (occurrence period) denotes the span that binds the event instance, while the time of detection is an instant or greater than the last instant of the occurrence period.

Allen [All83] argues that all events have duration and considers intervals to be the basic timing concept. A set of 13 relations between intervals is defined, and rules governing the composition of such relations control temporal reasoning.

A durative event can be seen as capturing the uncertainty over the time of occurrence and the time of detection of an event rather than modelling an event that persists over time. In this sense, durative events are akin to the point-interval-based-timestamps described above. The proposed model is based on primitive events that represent instantaneous changes of the system state, with uncertainty over their measurement. I would regard an event that persists over time as akin to a state, with an event at the start and one at the end of the time period. This could also be defined as a composite event. Determination of the duration of composite events requires semantics of composition and time system information. Fig. 4.2 shows 7 temporal relations defined in the system. Complex timing constraints among correlated event instances are precisely defined (see Chapter 7 for details).

4.4.5 Duplication

It is important to distinguish between multiple instances of a given event type, which may be primitive or composite events.

In sensor networks, to avoid loss of events by communication instability, duplicates of events may be produced to increase reliability. Duplicates have to be handled differently depending on the application and contexts within applications. In object tracking, for example, the most recent reading from a sensor is valid, and events prior to that will be obsolete except for the

historical record. On the other hand, for a transaction event in which a customer cancels an order, a duplicate event should be ignored as a transaction is being repeated.

Both primitive and composite events are recurrent. An event may occur multiple times, and simply applying event operators on events instead of specific event instances might cause semantic ambiguity. The semantics of event composition have to address handling of duplicates. [LMK98] take the approach of defining constraints on attributes of events and detect occurrences of events, before correlation conditions are evaluated. I propose duplicate handling in two ways: adding a selection operator as an event composition operator and adding subset rules as parameters without loss of meaningful data.

4.4.6 Typed Event

Definition 4.7 (Event Type) *The event type describes the structure of an event.*

Event types can be defined in XML with a certain document type definition, attribute-value pairs with given attributes and value domains or strongly typed objects. For example, an event notification from a publisher could be associated with a message m containing a list of tuples <type, attribute name(a), value (v)> in XML format, where type refers to a data type (e.g., float, string). Each subscription s is expressed as a selection of predicates in conjunctive form, i.e., $s = \bigwedge_{i=1}^{n} P_i$. Each element P_i of u is expressed as <type; attribute name(a); value range(R) > , where $R : (x_i; y_i)$. P_i is evaluated to be true only for a message that contains $< a_i; v_i >$. A message m matches a subscription s if all the predicates are evaluated to be true based on the content of m.

4.5 Publish/Subscribe Model

Events are described in the previous sections. Events are at the heart of publish/subscribe systems. This section outlines the publish/subscribe model that is evolving in ubiquitous computing scenarios. For example, geographical multicast delivers events based on the location of subscribers. Thus, not only do subscribers decide what to receive, but publishers also decide whom to deliver to. This section looks into publish/subscribe model and outline my vision of key elements (e.g., symmetric publish/subscribe) in publish/subscribe systems for ubiquitous computing environments.

4.5.1 Subscription Models

Subscription models can be classified into the following three categories: Topic-based, Content-based, and Type-based (see Chapter 2). In Topic-based publish/subscribe, events are divided into topics, and subscribers subscribe to topics. Common topic-based systems arrange topics in disjoint hierarchies so that a topic cannot have more than one super topic. In Content-based publish/subscribe, subscription is defined in a constrained manner and evaluated against event content. Type-based publish/subscribe ties events to the programming language type model, database schema, or semi-structured data model (e.g., XML). Type-based subscription [EFGK03] provides a natural approach for this if the language offers multiple sub-typing, thus avoiding explicit message classification through topics.

4.5.2 Symmetric Publish/Subscribe

In [RDJ02], the symmetrical nature of publications and subscriptions is discussed. In conventional publish/subscribe systems, if a publication matches a subscription, it is also implied that the

subscription matches the publication. A symmetric publish/subscribe system will only send notification to those subscribers whose subscriptions satisfy the publication. This symmetry allows subscribers to filter out unwanted information and lets publishers target information to a subset of subscribers. As an example, a publisher might want to publish information only to subscribers who are university students. A subscription can contain an active-attribute, which describes the actual information of the subscriber. This is an important concept for publish/ subscribe systems to support ubiquitous computing, where subscribers are mobile or the location or distance from a specific object is relevant. An example below shows that sensed data is published if subscribers are in a specific area:

```
Subscription: (store, (Tesco AND M&S)), (location, Cambridge)
Publication:  (store, Tesco), (location, Cambridgeshire)
               where, Cambridgeshire > Cambridge
```

Figure 4.3: Example Subscription and Publication in Symmetric Publish/Subscribe

In Fig. 4.3, the subscriber only receives the publication from *Tesco in Cambridge*. The event model therefore requires an expression of appropriate attributes for symmetric publish/subscribe.

4.5.3 Subscription Languages

Most event systems support a subscription language that allows a subscriber to express its information need. The resulting filtering expression is then used by the brokers to determine whether a particular event notification is of interest to a subscriber. If the language is expressive, efficient matching of notifications requires complex operations. However, if the language does not support rich constructs, its applicability is limited.

In a content-based subscription model, the subscription language takes an important role. Various subscription languages have been implemented (e.g., based on (attribute; value) pairs, SQL or XPath). Moreover, several filter algorithms have been proposed; extensive evaluations of main-memory filtering algorithms are given in [FLPS00] [PFLS01].

The placement of filters over networks is also critical. Some features of a subscription language bring complexity in time and space. State-based composition languages may be desirable for complex event composition especially for dealing with multiple events. Thus, two layers of language definition may be the best solution.

The expressiveness of the subscription language is crucial in publish/subscribe systems. Typical matching algorithms on subscriptions are conjunctions of predicates. The tradeoffs between expressiveness of the subscription language, including complex matching algorithms, and efficient processing are important when designing publish/subscribe systems. Examples of expressive subscriptions are shown below:

- Identify event patterns in a single event stream: 5 link-down events within 3 min from the same network segment
- Correlate events in multiple event streams: increase in temperature in room A and decrease in temperature in room B
- Validity of subscription: the event start time is before the event end time
- Notion of time (points, duration): all events at 15:00 and between 18:00 and 19:00
- Operators on time (before, after)
- Changes (rate of increase): when the price of the car decreases by 10 %, when the temperature changes by 2 % within 10 seconds

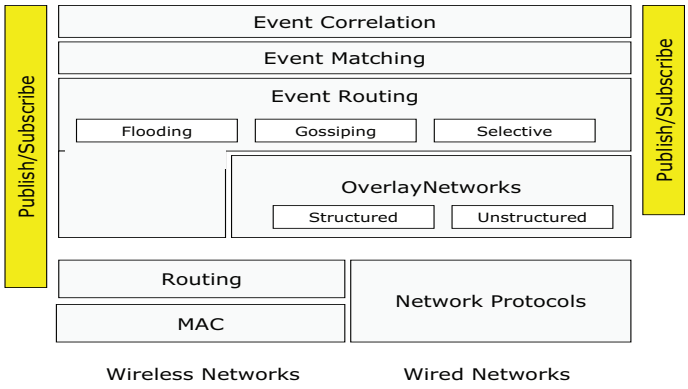


Figure 4.4: Publish/Subscribe Systems over Wired and Wireless Networks

- Location of objects: all events in postal code *CB30FD*
- Movement of objects: any objects moving east on $Highway1$
- Approximate relations: clothes with colour close to salmon pink
- Meta-model changes: when new publication types are advertised

4.5.4 Architectural Model

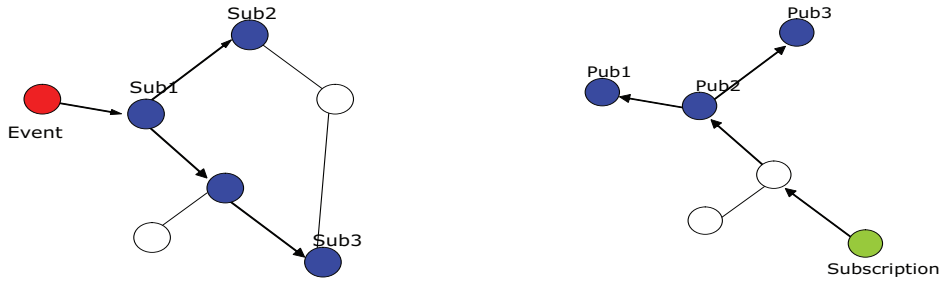
The architectural model is depicted in Fig. 4.4. It includes network, overlay infrastructure, event routing, and event matching layers. Publish/subscribe systems are often built over an application-level overlay network especially in wired network environments. An overlay layer could be an event dissemination tree from publishers to subscribers, an undirected acyclic graph spanning all brokers, or be based on transport level connections between nodes. In wireless networks, publish/subscribe systems need to take a cross layer approach. For details of overlay infrastructure, see Chapter 5 and for event routing, see Chapter 6.

4.5.5 Routing

Event routing strategies can be classified into three categories: simple flooding, parametric flooding (gossiping), and selective strategies (Fig. 4.4). In simple flooded routing, there are two extreme approaches: flooding events and subscription flooding, as depicted in Fig. 4.5. In event flooding, all events are flooded to all brokers, and each broker performs event matching operations against stored subscriptions. Event diffusion generates a large communication overhead. In subscription flooding, all subscriptions are flooded to all brokers, and published events are matched against the subscriptions before delivery. The communication overhead is small, but for dynamic systems it may not be practical. Neither of the extremes will scale, and various event dissemination algorithms have been attempted.

Flooding is based on the broadcast mechanism, and parametric flooding applies various parameters to control the broadcast to reduce network traffic. For example, when a node receives an event, it randomly forwards the event only to a certain number of nodes (i.e., gossip). The dynamics of an epidemic dissemination leads to robust, reliable, and self-stabilised communication.

The selective strategy provides a way to subset either events or subscriptions so that event routing is performed only to the members in the subset. In filter-based routing, events are forwarded towards nodes that are members of an overlay path leading to matching subscribers. This requires routing information to be stored at the nodes for building diffusion paths.



(a) Event Flooding (b) Subscription Flooding

Figure 4.5: Flooding Schemes

Rendezvous-based routing uses an anchor node for each topic. The subscribers' information is maintained within the rendezvous node, and when a publication event reaches the rendezvous node, it disseminates the event to registered subscribers. Rendezvous routing handles inherently dynamic changes in the overlay well. However, there is a bottleneck at hash bucket nodes, where subscribers are structured as a multicast diffusion tree. The subscription language may be restricted, since mapping between multidimensional multi-typed subscriptions to the uni- or bi-dimensional space of a structured overlay is complex. Optimisation includes two aspects: first, how to place the matching process over a distributed system, and second, how a subscription should be represented for an efficient matching process. Matching itself is already a complex problem, and content-based routing poses more challenges. Furthermore, the optimisation is influenced by data trends, source placement, and node ability, etc.

In simple flooding, if a publisher node keeps all subscriptions and performs matching, memory usage for subscriptions will be extremely high and not scalable. Thus, there have been several attempts for optimisation techniques. SIENA [Car98] takes filtering-based routing, where a undirected acyclic graph spans all the brokers, and each publisher maintains a dissemination tree (see Chapter 2 for more detail).

Efficient content matching requires all similar subscriptions to be grouped for efficient matching. Thus, event routing and content-based filtering become conflicting problems, because there is no guarantee that subscribers at close locations have similar interests. If applications are tolerant of the end-to-end latency and the message size is always small, then messages can be unicast in a bundle directly to subscribers.

4.5.6 Covering Relation of Filters

Covering relations and filter merging [Müh01] are used to reduce the size of the filters that are propagated in the event network, but they increase the cost of updating the topology to reflect changes caused by mobility and changes in subscriptions.

Existing content-based publish/subscribe systems [Car98] [Res01] focus on minimising event traffic, which is achieved by early matching and late replication. In Gryphon, subscriptions are replicated to all the nodes in the system, thus limiting scalability. An approach taken in Hermes is to minimise hop counts of notifications, but the resulting design is only beneficial when publishers happen to be close to subscribers or similar subscriptions are nearby. Thus far, there is no powerful architecture that is scalable in the number of nodes for distributed content-based filtering and routing.

4.6 Filter Matching

Efficient event filtering and matching algorithms are crucial for content-based publish/subscribe systems, and involve two tasks: to compare the similarity among subscriptions and publications and to provide a mechanism for efficient matching operations on publications and subscriptions. Most publish/subscribe systems focus on multiple one-dimension indices, which suffer from poor performance of the search mechanism for inequality operators.

A common approach for event filtering uses a set of one-dimensional index structures for the predicates in subscriptions. First, all events satisfied by the predicates are selected, and the result is matched against the subscriptions. The predicates are grouped based on all subscriptions. Thus, for each attribute, one predicate index is built. Examples of predicate indexing are the *Count Algorithm* [YGM99] and *Hanson's Algorithm* [H+99] [HCKW90]. The performances of these algorithms have the same complexity order.

Algorithms described in [HB02] insert subscriptions into a matching tree. When events are matched against the tree, they start from the root node and operate filtering through intermediate nodes. An event that goes through all intermediate nodes reaches the leaf nodes where matching subscriptions are stored. [ASS+99] and [HB02] built subscription index trees based on subscription schema.

In [W+04] [Zha04], the performance of multidimensional indexing (e.g., UBTree [BM98]) event matching is reported to be three orders of magnitudes faster than the *Count Algorithm*.

The publish/subscribe matching issue is how to provide a flexible abstraction model of an event, so that a system can set the line between publication and subscription data model.

4.7 Events in Hypercube

This section presents *Hypercube* for events in publish/subscribe systems.

Event filtering on content-based publish/subscribe can be considered as queries in a high dimensional space, but applying multidimensional index structures to publish/subscribe systems is still unexplored. Ubiquitous computing produces high volumes of sensor data, which may suit multidimensional indexing techniques. This section presents *Hypercube*, a multidimensional indexing scheme based on RTree, which provides more effective range queries. Thus, both publication and subscription are modelled as hypercubes in the implementation, where matching is regarded as an intersection query on hypercubes in an n-dimensional space. Point queries on *Hypercube* are transformed into range queries to make use of efficient point access methods for event matching. This corresponds to the realisation of symmetric publish/subscribe, and it automatically provides range queries, nearby queries, and point queries.

4.7.1 Multidimensional Event

The techniques of summarisation and aggregation of filters and matching mechanisms greatly impact the efficiency of search, especially since the events are becoming more multidimensional. The most popular additional dimensions are spatial and temporal data, which databases have integrated for supporting continuous queries such as tracking a moving object. The characteristics of spatial data are complex and dynamic. Without any standard spatial algebra, it is difficult to order spatial objects in a linear fashion.

Nevertheless, supporting spatial, temporal, and other event attributes with multidimensional index structure can dramatically enhance filtering and matching performance in publish/subscribe systems. For example, the event of tracking a car, which is associated with changes of position through time, needs spatio-temporal indexing support. GPS, wireless computing and mobile phones are able to detect positions of data, and ubiquitous applications desperately need this data type for tracking, rerouting traffic, and location aware-services. Queries against multidimensional data include:

- Exact match
- Point query
- Intersection or region query or overlap query
- Nearest neighbour query
- Enclosure query
- Adjacency query

I consider event filtering as search in high dimensional data space and introduce a hypercube based filtering model. It is popular to index spatio-temporal objects by considering time as another dimension on top of a spatial index so that a 3-dimensional spatial access method is used. I consider extension to n dimensions, which allows to include any information such as weather, temperature, or interests of the subscribers. Thus, this approach takes advantage of the range query efficiency by multidimensional indexing. The indexing mechanism with *Hypercube* can be used for filtering expression for a content-based filtering, aggregation of subscription, and part of the event correlation mechanism. Ultimately, the event itself can be represented as a hypercube for symmetric publish/subscribe.

Content-based subscription requires a filtering process, and it can be placed on the publisher and subscriber edge brokers, or distributed over the networks based on the coverage relationship of filters. If the publish/subscribe system takes rendezvous routing, a rendezvous node needs to keep all the subscriptions for the matching process such as in Comet [LP05], where a Hilbert Space Filling Curve is used. Multidimensional range queries support selective data to subscribers who are interested in specific data.

There are various data structures and access methods for multidimensional data, and an overview and comparison analysis are found in [dBKOS98] [ASS+99] [GG98] [AMW01]. Choosing the indexing structure is complex and has to satisfy the incremental way of maintaining the structure and range query capability.

RTree [Gut84] does not offer good average memory utilisation and performance, and R*Tree [BKSS90] underperforms on region splitting and merging while updating the index. Most multidimensional access methods like R*Tree, or Grid-Files do not allow for efficient incremental organisation, e.g., by requiring forced reinsertion, Grid-Splits or even complex reorganisations. Despite of these disadvantages RTree is widely used for spatio-temporal data indexing, and it supports tree splitting and merging operations in a dynamic form. Thus, I have chosen RTree to represent multidimensional events and event filtering.

UB-tree [Bay96] [FMB02] [RMF+00] is also designed to perform multidimensional range queries. It is a dynamic index structure based on BTree and supports updates with logarithmic performance like BTree with space complexity O(n). In [W+04], UB-tree is used for message filtering in publish/subscribe systems.

I have discussed symmetric publish/subscribe systems in Section 4.5.2. Both point and range queries can be operated over *Hypercube* in a symmetric manner between publishers and sub-

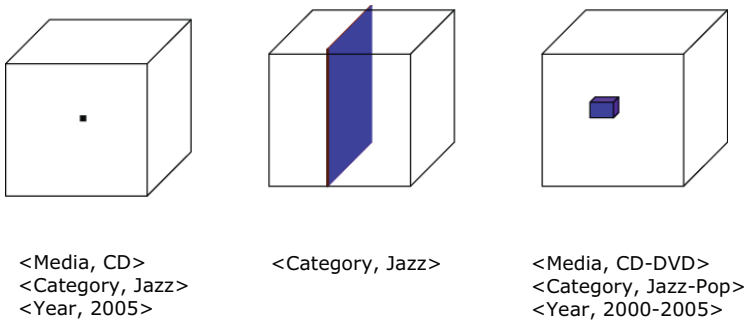


Figure 4.6: 3-Dimensional Subscription

scribers. The majority of publish/subscribe systems consider that subscriptions cover event notifications. I focus on symmetric publish/subscribe, and the case of when event notifications cover subscriptions is therefore also part of the event filtering operation. Thus, typical operations with *Hypercube* can be classified into the following two categories:

- **Event Notifications $\subseteq$ Subscriptions:** events are point queries and subscriptions are aggregated in *Hypercube*. For example, subscribers are interested in the stock price of various companies, when the price dramatically goes up. All subscribers have interests in different companies, and an event of a specific company's price change will be notified only to the subscribers with the matching subscriptions.
- **Event Notifications $\supseteq$ Subscriptions:** events are range queries and subscriptions are point data. For example, a series of news related to *Bill Gates* is published to the subscribers who are located in New York and Boston. Thus, an attribute indicates the location in the event notification to *New York and Boston*. Subscribers with the attribute *London* will not receive the event.

In general, *Hypercube* supports range subscriptions queried against range events.

Cube Subscription

Events and subscriptions can essentially be described in a symmetric manner with *Hypercube*. Consider an online market of music, where old collections may be on sale. Events represent a cube containing 3 dimensions (i.e., Media, Category, and Year). Subscriptions can be:

- Point Query: CDs of Jazz released in 2005
- Partial Match Query: Any media of Jazz
- Range Query: CDs and DVDs of Jazz and Popular music released between 2000 and 2005

Fig. 4.6 depicts the 3-dimensional *Hypercube* and the above subscriptions are shown accordingly.

Expressiveness

Typical examples of queries and subscriptions in geographical information systems are as follows:

- Find objects that crossed through region A at time t_1, came as close as possible to point B at a later time t_2 and then stopped inside circle C some time during interval (t_3, t_4).
- Find objects that first crossed through region A, then passed point B as close as possible and finally stopped inside circle C (the exact time when this occurs is not important).
- Find the object trajectory that crossed to point A as close as possible at time t_1 and then as close as possible from point B at a later time t_2.

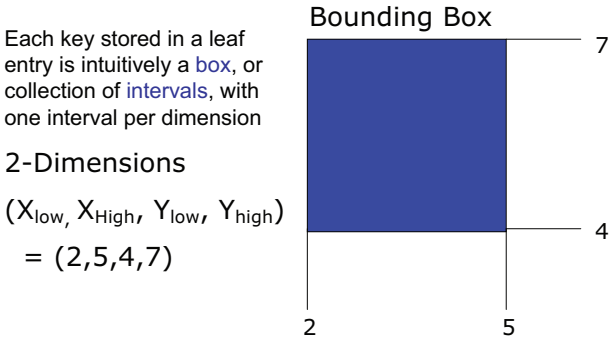


Figure 4.7: Minimum Boundary Rectangle in RTree

Hypercube can express these subscriptions and filtering by use of another dimension with time values. A simple real world example for use of *Hypercube* can be with geographical data coordinates in 2-dimensional values. A query such as *Find all book stores within 2 miles of my current location* can be expressed in an RTree with the data splitting space of hierarchically nested, and possibly overlapping, rectangles.

4.7.2 RTree

An RTree [Gut84], extended from a B^+Tree, is a data indexing structure that can index multi-dimensional information such as spatial data. Fig. 4.7 shows an example of 2-dimensional data. An RTree is used to store minimum boundary rectangles (MBRs), which represents the spatial index of an n-dimensional object with two n-dimensional points. Similar to BTrees, RTrees are balanced on insert and delete, and they ensure efficient storage utilisation.

Structure

An RTree builds a MBR approximation of every object in the set of data and inserts each MBR in the leaf level nodes. Fig. 4.8 illustrates a 3-dimensional RTree; rectangles A-F represent the MBRs of the 3-dimensional objects. The parent's nodes, R5 and R6, represent the group of object MBRs. When a new object is inserted, a cost-based algorithm is performed to decide in which node a new object has to be inserted. The goals of the algorithm are to limit the overlap between nodes and to reduce the dead-space in the tree. For example, grouping objects A, C, and F into R5 requires a smaller MBR than if A, E, and F were grouped together instead. Enforcing a minimum/maximum number of object entries per node ensures balanced tree formation. When

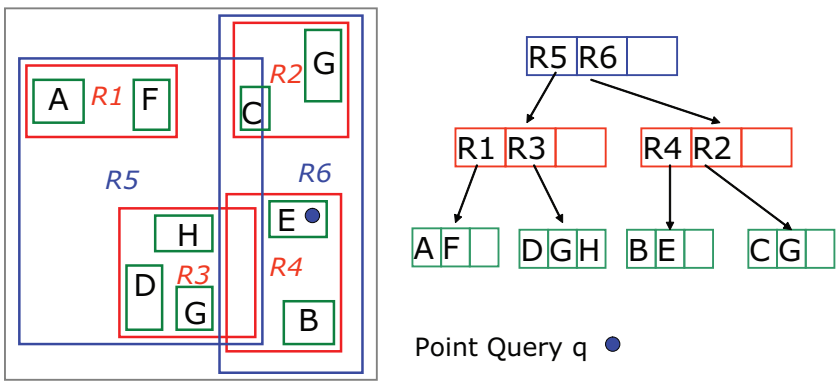


Figure 4.8: RTree Structure

a query object is coming into the tree for the intersection operation, the tree is traversed, starting at the root, by passing each node where the query window intersects a MBR. Only object MBRs that intersect the query MBR at the leaf-level have to be retrieved from the disk. A BTree may require a single path through the tree to be traversed, while an RTree may need to follow several paths, since the query window may intersect more than one MBR in each node. MBRs are hierarchically nested and can overlap. The tree is height-balanced; every leaf node has the same distance from the root. Let M be the number of entries that can fit in a node and m the minimum number of entries per node. Leaf and internal nodes contain between m and M entries. As items are added and removed, a node might overflow or underflow and require splitting or merging of the tree. If the number of entries in a node falls under the m bound after a deletion, the node is deleted, and the rest of its entries are distributed among the sibling nodes.

Each RTree node corresponds to a disk page and an n-dimensional rectangle. Each non-leaf node contains entries of the form $(ref, rect)$, where ref is the address of a child node and $rect$ is the MBR of all entries in that child node. Leaves contain entries of the same format, where ref points to an object, and $rect$ is the MBR of that object.

There are many variations of RTrees (see [MNPT05]). The original RTree, proposed by Guttman, has influenced all the variations of dynamic RTree structures. RTrees do not guarantee reasonable performance in the worst case scenario, but it is known that they perform well with real-world data. A Hilbert RTree [KF94] can perform well relative to all other variants. In this RTree, the Hilbert curve gives closer Hilbert values to the nearer points, leading to an efficient search. Better interval clustering can be achieved by using *packed* RTrees such as an STRTree [LLE97]). Interval fragmentation is used in the Segment RTree [KS91] for improved performance. Priority RTree claims to be the most efficient method among RTrees.

In this dissertation, the use of the original RTree is focused and leave comparisons between different RTree varieties as a future research agenda.

Insertion and Deletion

The algorithms of insertion and deletion use MBRs to ensure that *nearby* elements are kept in the same leaf node, especially in cases when a new element goes into the leaf node that may require the least enlargement in its MBR.

Insertion includes inserting its MBR to the RTree along with a reference to the object in the ref field of the new entry. Only one path of the tree is traversed, and the new entry is inserted in a leaf node. If the MBR of the object intersects many entries of an intermediate node, the child whose MBR is less enlarged after the insertion is followed. In the case of a tie, other criteria are applied such as the node's cardinality, or the MBR area size. The object is inserted only at one leaf, and if it causes the leaf page to overflow, the page is split in two, again after applying several criteria. The split can be propagated to the ancestor nodes. If an insertion causes enlargement of the leaf page's MBR, adjustment is required, which propagates the change upwards.

At first, deletion requires an exact match query for the object. If the object is found in a leaf, it is deleted. Again the deletion may trigger a structure change for the tree, as it can cause the leaf page where it is deleted from to underflow (the number of entries may fall under m). In the case of an underflow, the whole node is deleted, and all its entries are stored in a temporary buffer and reinserted into the tree. As for insertion, deletion may affect the MBR of the page. In that case, the change is propagated up along the search path.

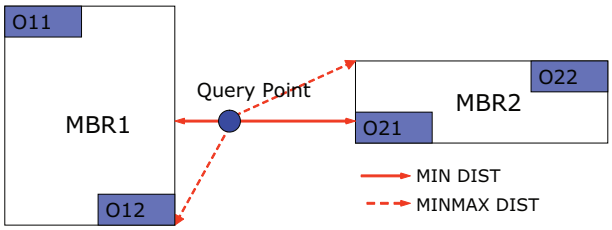


Figure 4.9: Nearest Neighbour Search

Search

Search in an RTree is performed in a similar way to that in a BTree. Search algorithms (e.g., intersection, containment, nearest) use MBRs for the decision to search inside a child node. This implies that most of the nodes in the tree are never *touched* during a search. The average cost of search is $O(logn)$ and the worst case is $O(n)$. Different algorithms can be used to *split* nodes when they become full, resulting in *Quadratic* and *Linear* RTree sub-types.

In Fig. 4.8, a point query q requires traversing R5, R6 and child nodes of R6 (e.g., R2 and R4) before reaching the target MBR E. When the coverage or overlap of MBRs is minimised, RTree gives maximum search efficiency.

For the *nearest neighbour* (NN), search for point data is based on the distance calculation shown in Fig. 4.9. Let $MINDIST(P, M)$ be the minimum distance between a query point and a boundary rectangle, and let $MINMAXDIST(P, M)$ be the upper bound of minimum distance to data in the boundary rectangle (i.e., among the points belonging to the lines consisting of MBR, select the one closest to the query point). However, there is no guarantee that the MBR contains the nearest object even if $MINDIST$ is small. In Fig. 4.9, the smaller MINDIST from the query point is MBR1, while the nearest object of O21 is in MBR2. The search algorithm for *nearest neighbour* is:

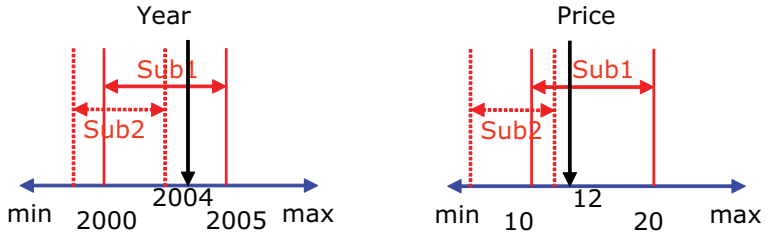
1. If the node is a $leaf$, then find NN. If non $leaf$, sort entries by MINDIST to create *Active Branch List* ABL.
2. if $MINDIST(P, M) > MINMAXDIST(P, M)$ then remove MBR. If the distance from the query point to the object is larger than $MINMAXDIST(P, M)$ then the object is removed (i.e., M contains an object that is closer to P than the object). If the distance from the query point to the object is larger than $MINDIST(P, M)$, then M is removed (i.e., M does not contain objects that are closer to P than the object).
3. Repeat 1 and 2 until ABL is empty.

4.7.3 Experimental Prototype

The prototype implementation of RTree is based on the Java implementation by Marios Hadjueleftheriou, which is based on the paper by Guttman [Gut84]. I extended it to become more compact. It currently supports range, point, and nearest neighbour queries. The prototype is a 100KB class library in Java with JDK 1.5 SE. The purpose of the experiments in this section is to establish the applicability of an RTree for event and subscription representation. A few basic examples with the publish/subscribe paradigm are shown, and various experiments using traffic monitoring data from the *Cambridge City Scoot System* are shown in Section 4.7.4. *Hypercube* is used for subscription filters in content-based publish/subscribe systems in Chapter 5.

2D Example

Fig. 4.10 illustrates 2-dimensional events for RTree, where the two attributes *year* and *price* represent two dimensions. The subscription predicates have range values. Both notification events and subscriptions are transformed to ranged values and are conjunctions of intervals. Fig. 4.11 shows MBRs of the subscriptions and a publication. This example uses the numeric value directly from the original data to simplify the experiment, but the input value for RTree can be a hashed value from the string value or any other numerical forms.



2-dimensional Subscription:

1. <year, 2000<=y<=2005><price, 10<=p<=20>
2. <year, 1998<=y<=2003><price, 3<=p<=11>

Event Publication: <year, 2004><price, 12>

Figure 4.10: 2-Dimensional RTree Publish/Subscribe

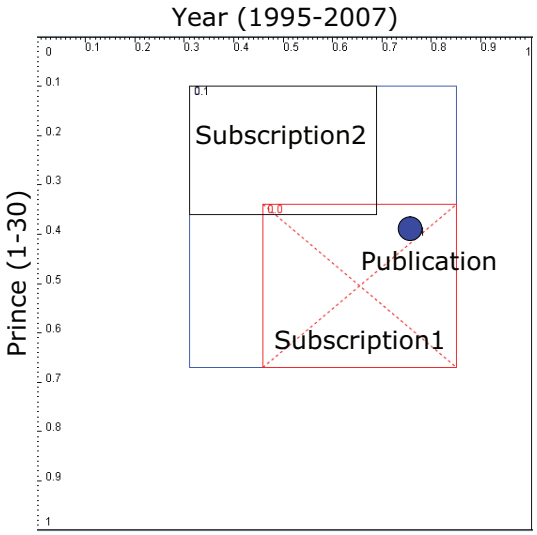


Figure 4.11: 2-Dimensional RTree Publish/Subscribe in RTree

3D Example

A subscription contains 3-dimensional attributes; release year (*year*), ranking (*rank*), and price (*price*) shown in Fig. 4.12 with 8 subscriptions. In Fig. 4.13, subscriptions are transformed to scaled values, which is the input for constructing the RTree. Against the above subscriptions, the event notification shown in Fig. 4.14 is published. Thus, this example is a publish/subscribe model with the query paradigm *Point publication to Range Subscriptions*, where publication represents a point and matched against many subscriptions. Fig. 4.15 shows the filtering process,

```
Subscriptions: YearLow YearHigh RankLow RankHigh PriceLow PriceHigh

1 1999 1999 10 80 20 20
2 1995 2005  1 22 10 15
3 1998 1999 30 99 12 20
4 2001 2006 30 88  8 15
5 2002 2004 60 90  5 18
6 1995 1999 10 80 11 16
7 2000 2006 22 33 15 19
8 1996 2001 30 55  5 20
```

Figure 4.12: 3-Dimensional Subscriptions

```
Transformed Scaled Values:
1 0.017110266  0.017110266 0.019011406  0.15209125  0.036121674  0.036121674
2 0.009505703  0.02851711  0.0019011407 0.041825093 0.017110266  0.02661597
3 0.015209125  0.017110266 0.05703422   0.18821293  0.020912547  0.036121674
4 0.020912547  0.03041825  0.05703422   0.16730037  0.013307985  0.02661597
5 0.022813689  0.02661597  0.11406844   0.17110266  0.0076045627 0.032319393
6 0.009505703  0.017110266 0.019011406  0.15209125  0.019011406  0.02851711
7 0.019011406  0.03041825  0.041825093  0.06273764  0.02661597   0.03422053
8 0.0114068445 0.020912547 0.05703422   0.10456274  0.0076045627 0.0361216741
```

Figure 4.13: 3-Dimensional Subscriptions: Scaled Values

```
Event Notification:

1999 1999 10 10 15 15

0.017110266 0.017110266 0.019011406 0.019011406 0.02661597 0.02661597
```

Figure 4.14: Event Notification for 3D Publish/Subscribe

```
Result:
Matching ID=2
Matching ID=6

Indexed space:0.009505703 0.0019011407 0.0076045627:0.03041825 0.18821293
0.036121674
Dimension: 3
Utilization: 40%
Tree height: 1
Number of data: 8
Number of nodes: 1
Query results: 2
```

Figure 4.15: Matched Subscriptions

and the event is delivered to subscribers with subscription 2 and 6. Fig. 4.16 shows a visualisation of the publications and subscriptions in the RTree.

4.7.4 Traffic Data in Cambridge (Scoot)

The Computer Laboratory currently receives a live traffic feed from Cambridge City Council (Scoot). This experiment uses Scoot data, where range queries are operated on sets of Scoot point data. Point data are transformed into range data so that intersection operations can be used to query range subscriptions over range data. The purpose of this experiment is to demonstrate the functionality of RTree and compare the RTree operation with $brute force$ operations, where the set of predicates are used for query matching.

Complex range queries directly mapping to real world incidents can be processed such as *speed of average car passing at junction A is slower than at junction B at 1:00 pm on Wednesdays*. It

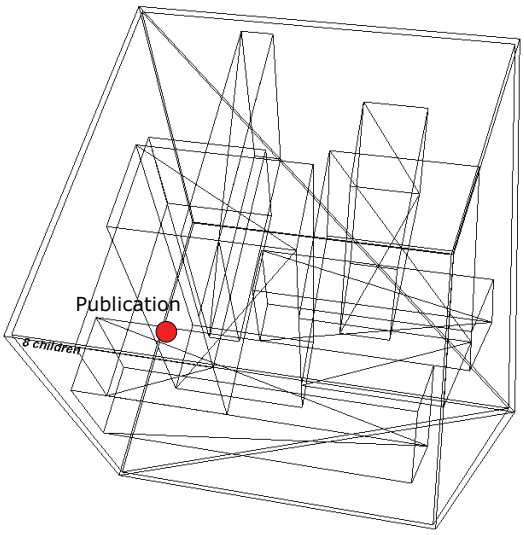


Figure 4.16: Visualisation of 3-dimensional Publish/Subscribe in an RTree

is not easy to show the capability of the *Hypercube* filtering for expressive and complex queries in a quantitative manner. Thus, experiments focus on the performance of a high-volume range filtering processes.

4.7.4.1 Cambridge City Scoot System

Data is gathered from inductive loops installed at various key junctions across Cambridge. In Fig. 4.19, black dots mark the location of induction loops. The stream data contains several messages: M31 (traffic light settings), M63 (aggregate vehicle counts), and B12 (bus lane data). Data is collected every five minutes from raw signal information. Three example messages are shown in Fig. 4.17, followed by data formats in Fig. 4.18.

```
1143873113    Sa 07:30:39 B12  N06141J  Bus 9981872  AVL : Imp 3  Jt 12  Vq
32  Vy 5  Rx 2  Ok

11143873114   Sa 07:30:40 M31 N03111G  COLOUR 0  CONTROL

01143873254   Sa 07:33:00 M63 N03111A1 PERIOD 300  FLOW   0  OCC 176
```

Figure 4.17: M31, M63, and B12 Data Examples

Message Type	Example	Description
M31	COLOUR 0 CONTROL 0	new colour of traffic light scoot would like (0=red, 1=green) and whether scoot is in control or not (0=yes, 1=no)
M63	PERIOD 300 FLOW 0 OCC 176	The interval period between messages (in seconds), the vehicle flow count in the period and the raw occupancy count (number of *quarter* seconds the loop was occupied)
B12	Bus 9981872 AVL : Imp 3 Jt 12 Vq 32 Vy 5 Rx 2 Ok 1	A message to state that a bus was detected on a road section. Fields are: bus identity, bus importance factor (0 to 7), bus journey (cruise) time, vehicle queue clear max queue(s), bus vary(s), bus RX lag(s), bus ok (0=ignored, 1=ok)

Figure 4.18: M31, M63, and B12 Data Format

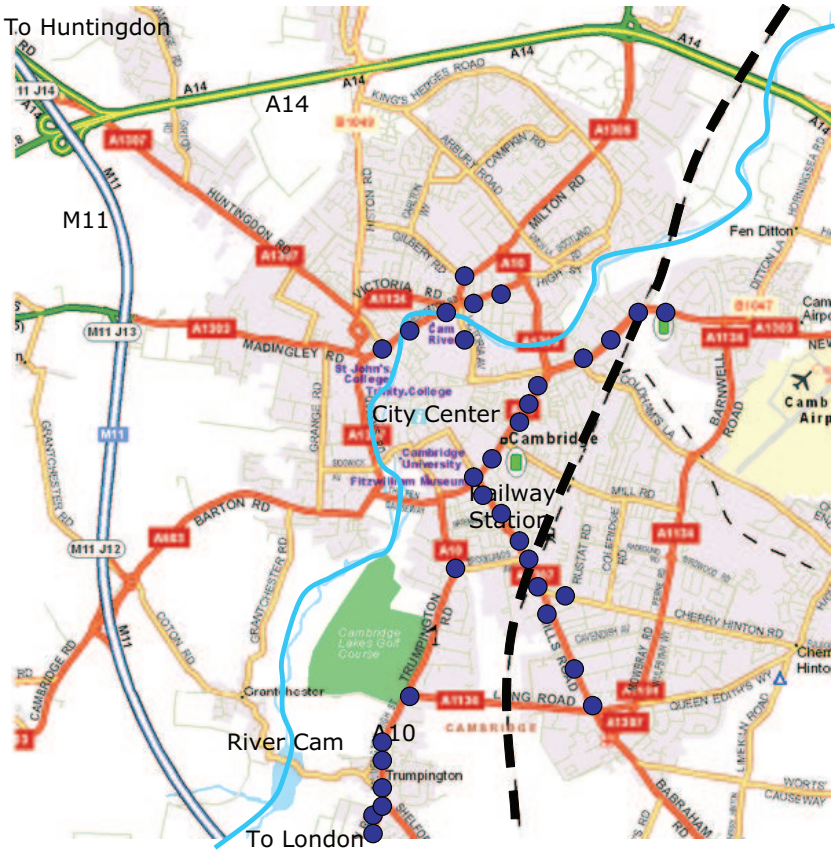


Figure 4.19: Location of Inductive Loops in Cambridge

4.7.4.2 Constructing an RTree

The M63 data from April 3rd 2006 is used for the experiment, which is transformed into 1-, 3-, and 6-dimensional data with attributes *Date, Day, Time, Location, Flow* and *Occupancy*. The M63 raw data are point data, which are transformed into zero size range data so that range queries can be issued against them by the intersection operation. The input data format to the RTree is shown in Fig. 4.20, and the scaled RTree data is shown in Fig. 4.21.

Element	Example	Description
1	1	RTree Operation (1: Insert 2:Query 3:Remove)
2	-1588619048	Unique identifier of data
3-4	20060403	Date (Year Month Day)
5-6	1	Day (1-7 map to Monday to Sunday)
7-8	010300	Time (Hour Minute Second)
9-10	3111	Junction of Induction loop location
11-12	0	Vehicle flow count during period (300 seconds – 5 mins.)
13-14	176	Raw occupancy count (number of quarter seconds the loop was occupied)

Figure 4.20: Input Data format to RTree

```
1 -1588619048 0.19391635 0.19391635 0.0 0.0 0.038245603 0.038245603
0.25769043 0.25769043 0.0 0.0 0.33460075 0.33460075

1 1888699798 0.19391635 0.19391635 0.0 0.0 0.038245603 0.038245603 0.25769043
0.25769043 0.0076045627 0.0076045627 0.009505703 0.009505703

1 1071051348 0.19391635 0.19391635 0.0 0.0 0.038245603 0.038245603 0.25769043
0.25769043 0.0 0.0 0.33460075 0.33460075
```

Figure 4.21: Scaled Data of M63 for RTree

Query:

A query shown in Fig. 4.22 is issued against the Scoot data. The result of the query process is shown in Fig. 4.23, where the matching operation shows the correct result.

```
2 1 0.19391635 0.19391635 0.0 0.0 0.038245603 0.075377256 0.013549805
0.28479004 0.0 0.19011407 0.19011407 0.34220532
```

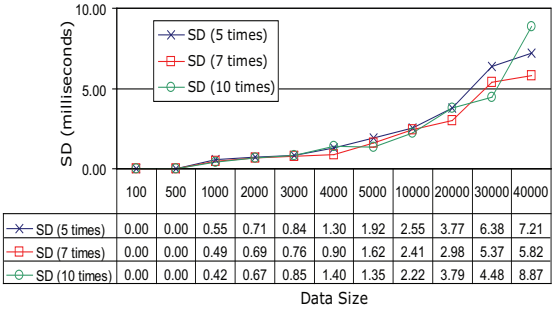
Figure 4.22: Query for M63 6-Dimensional Data

```
Matching ID=1861830593
Matching ID=1860132506
Matching ID=1860907072
Matching ID=1862903069
Matching ID=72586709
Matching ID=1914608655
Matching ID=-1584001443
Matching ID=1075668953
Matching ID=258020503
Matching ID=207205817 . . .

Indexed space: 0.19391635 0.0 0.038245603 0.013549805 0.0 0.0 : 0.19391635
0.0 0.8755644 1.0023193 0.3745247 1.8992395

Dimension: 6
Fill factor: 0.7
Index capacity: 20
Leaf capacity: 20
Near minimum overlap factor: 32
Reinsert factor: 0.30000001192092896
Split distribution factor: 0.4000000059604645Utilization: 64%
Tree height: 4
Number of data: 43637
Number of nodes: 3645
Level 0 pages: 3364
Level 1 pages: 260
Level 2 pages: 20
Level 3 pages: 1
Splits: 0
Query results: 169
```

Figure 4.23: Query Result for M63 6-Dimensional Data

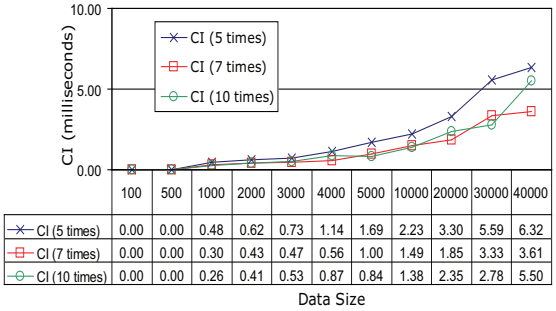
4.7.4.3 Evaluation

A dedicated PC (Intel Pentium M 760 - 2GHz - RAM 1GB) has been used for the experiments. Different sizes of data sets are used for the experiments, ranging from 100 to 40,000.



	100	500	1000	2000	3000	4000	5000	10000	20000	30000	40000
SD (5 times)	0.00	0.00	0.55	0.71	0.84	1.30	1.92	2.55	3.77	6.38	7.21
SD (7 times)	0.00	0.00	0.49	0.69	0.76	0.90	1.62	2.41	2.98	5.37	5.82
SD (10 times)	0.00	0.00	0.42	0.67	0.85	1.40	1.35	2.22	3.79	4.48	8.87

a. Standard Deviations



	100	500	1000	2000	3000	4000	5000	10000	20000	30000	40000
CI (5 times)	0.00	0.00	0.48	0.62	0.73	1.14	1.69	2.23	3.30	5.59	6.32
CI (7 times)	0.00	0.00	0.30	0.43	0.47	0.56	1.00	1.49	1.85	3.33	3.61
CI (10 times)	0.00	0.00	0.26	0.41	0.53	0.87	0.84	1.38	2.35	2.78	5.50

b. Confidence Intervals

Figure 4.24: Standard Deviations and Confidence Intervals of 6D RTree

Each experiment is executed 5 to 20 times, and the standard deviation (SD) and confidence interval (CI) values are calculated. Fig. 4.24 depicts SD and CI values of RTree for 6 dimensions in the experiment described in the following section (see Fig. 4.25). SD and CI are derived using the following formula. This procedure applies to all the experiments in this dissertation.

$$\sigma = \sqrt{1/n \sum_{i=1}^{n} (x_i - \bar{x})^2}$$

where σ is the standard deviation for sample data x.

The confidence interval at a 95% confidence level was derived from the following formula:

$$\bar{x} \pm 1.96(\sigma/\sqrt{n})$$

where $\bar{x}$ is the sample mean and n is the sample data size.

Fig. 4.24 shows that the CI from 10 executions is similar or lower than the CI from 5 executions. The distribution of the results in Fig. 4.25 does not get any impact from values within the range of CI. Thus, increased repetition of experiments does not give extra accuracy for the experiments and it indicates 5 executions of the experiment are sufficient. The experiment results shown in this dissertation are averaged from results of at least 5 executions. This applies to all the experiments in this dissertation unless stated otherwise.

The evaluation involved the following metrics:

- Dimension size
- Matching time
- RTree storage size
- RTree construction time
- Data trend

Dimension Size

Fig. 4.25 and Fig. 4.26 show the processing speed of a range query. The X axis indicates the data size; it is not linearly scaled over the entire range. The sizes of data sets are selected between 100 and 40,000 as seen on the X axis. Two partitions (between 1000 and 5000, and between 10,000

and 40,000) are scaled linearly. This applies to all the experiments, where different data sets are used. An RTree has been created for data of 1, 3, and 6 dimensions.

The operation using the *brute force* method is also shown, where each predicate is compared with the query. For 1-dimensional data, the use of RTree incurs too much overhead, but the RTree outperforms at increasing numbers of dimensions.

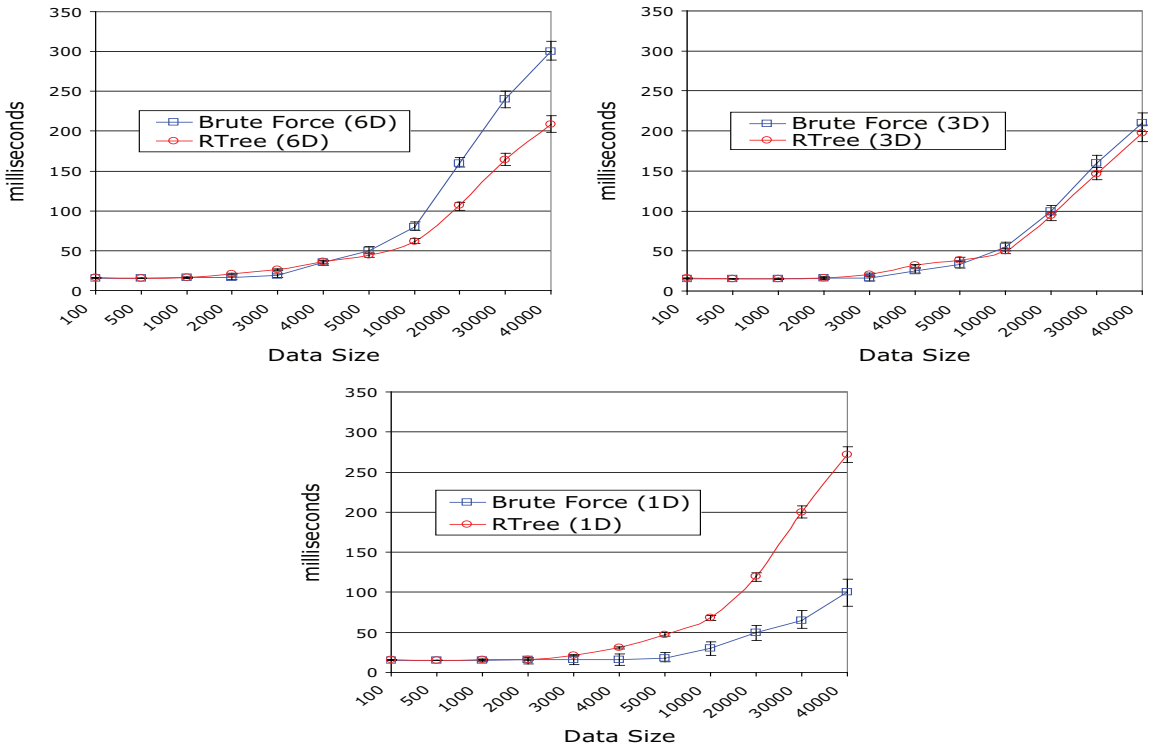


Figure 4.25: Single Range Query Operation: RTree vs. Brute Force

The difference in the number of dimensions has little influence over the RTree performance. Thus, once the RTree structure is set, it guarantees an upper bound on the search time.

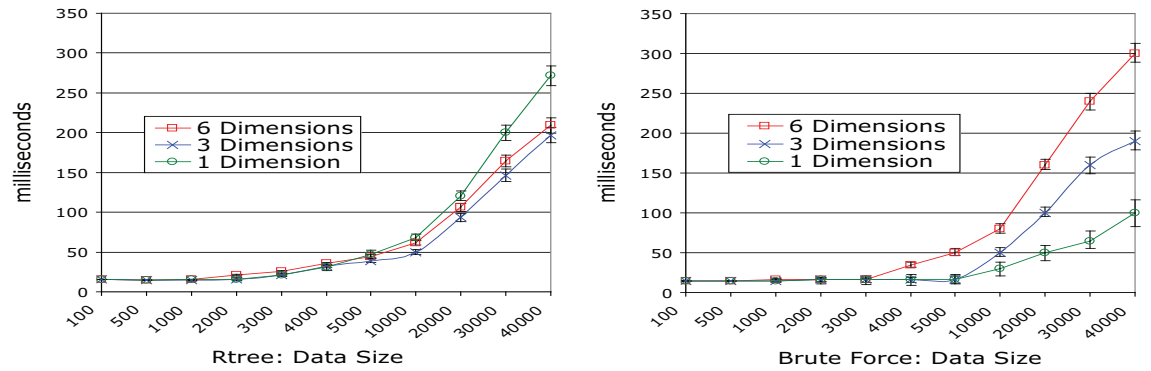


Figure 4.26: Single Range Query Operation: Dimensions

Matching Time

Fig. 4.27 and Fig. 4.28 show average matching operation times for a data entry against a single query. The Y axis indicates *total matching time / number of data items*. The standard deviations are ≤ 0.0005 and the confidence intervals for the population mean at a 95% confidence level in this experiment are ≤ 0.00053. For example the experiment for 1 dimension in Fig. 4.27 shows the confidence intervals, which are negligible range. Thus, error bars are not shown in the other figures in this section.

For 1-dimensional data, the use of RTree incurs too much overhead, but increasing dimensions does not affect operation time. In these figures, the X axes are in non-linear scales. The cost of the brute force method increases with increasing dimension of data, which is shown in Fig. 4.28.

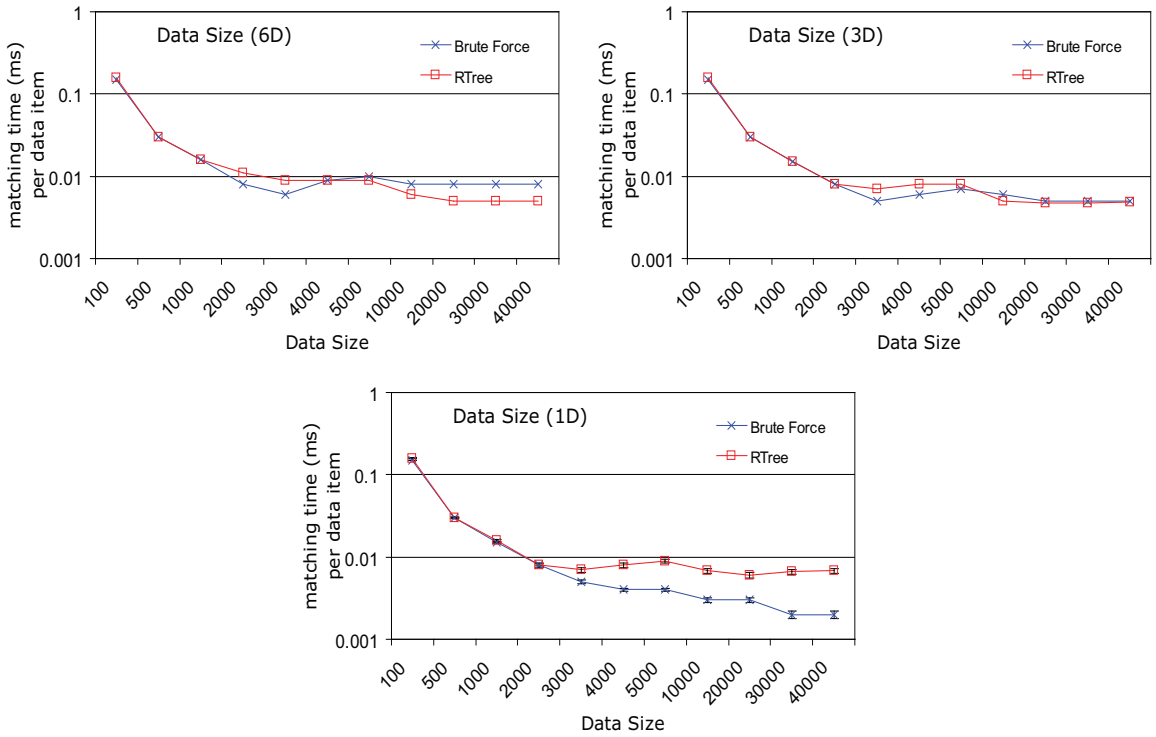


Figure 4.27: Single Range Query Matching Time

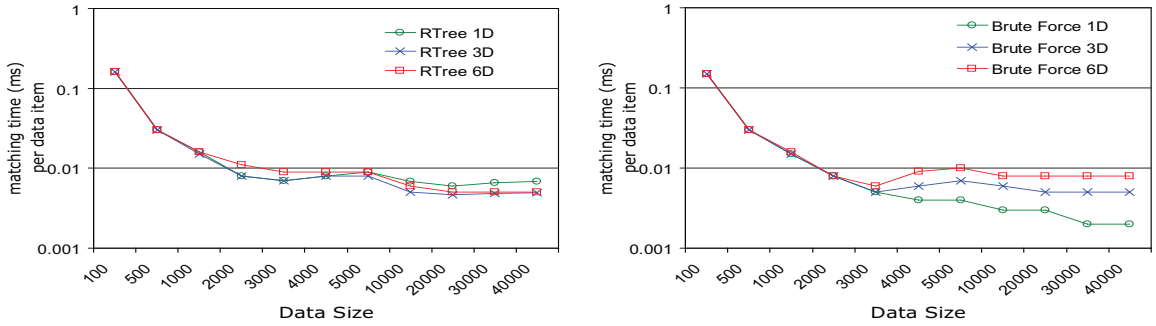


Figure 4.28: Matching Time

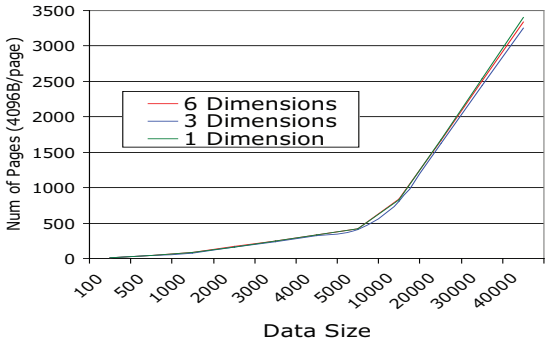


Figure 4.29: Storage Usage

RTree Storage Size

Fig. 4.29 shows the storage requirement for RTree. The current configuration uses 4096B per page. Since the index may also contain user defined data, there is no way to know how big a single node may become. The same sets of data is used for the repeating experiments and the standard deviation is therefore 0. The storage manager will use multiple pages per node if needed, which will slow down performance. There are only few differences with changing dimension size, because the data size in each element is about the same in this experiment. The standard deviation value is $\cong 0$, because the input data for each experiment is identical.

RTree Construction Time

Fig. 4.30 shows the RTree construction time. Construction time vs. data size is close to linear. The RTree index [Gut84] is a balanced tree structure consisting of index nodes, leaf nodes and data.

Since the index is balanced, nodes should be under full but not empty. A fill factor defines the minimum number of entries in any node, and is normally around 70%. Fig. 4.31 shows the configuration values. If a stored RTree is reused, the index ID is the only information which can be modified. The index and leaf capacity, the fill factor, and the dimensionality stay the same. Fig. 4.33 depicts gradient values for the linear curve and actual numbers are shown in Fig. 4.32. The Y axis indicates construction time per data item. The correlated value shows a

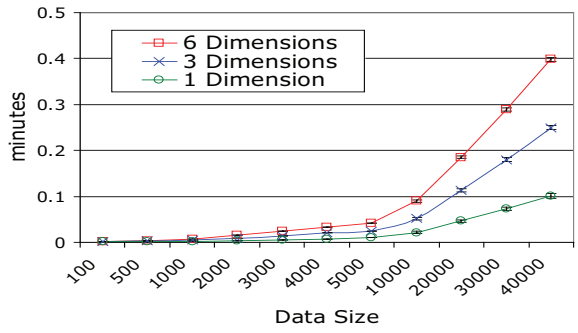


Figure 4.30: RTree Construction Time

```
Fill Factor: 0.7 (Node capacity)
Index Capacity: 20
Reinsert factor: 0.3
Split Distribution factor: 0.4
Utilization: 100 * num of Data / (num of Nodes in Level(0) * Leaf Capacity)
Tree Height: Maximum depth of RTree
Total Nodes: Total num of nodes in RTree
Nodes on Level: num of nodes on each level of tree
Splits: num of node splits for balancing the tree by Insert/Remove operation
Adjustments: num of adjustments to modify the tree by Insert/Remove operation
```

Figure 4.31: Configuration of RTree Construction

slight increase with the data size.

Data size	(1000 - 5000)	(10000 - 40000)
1 dimension	y = 0.0000020x	y = 0.0000023x
3 dimensions	y = 0.0000045x	y = 0.0000058x
6 dimensions	y = 0.0000080x	y = 0.0000095x

Figure 4.32: Correlated Values for a Curve

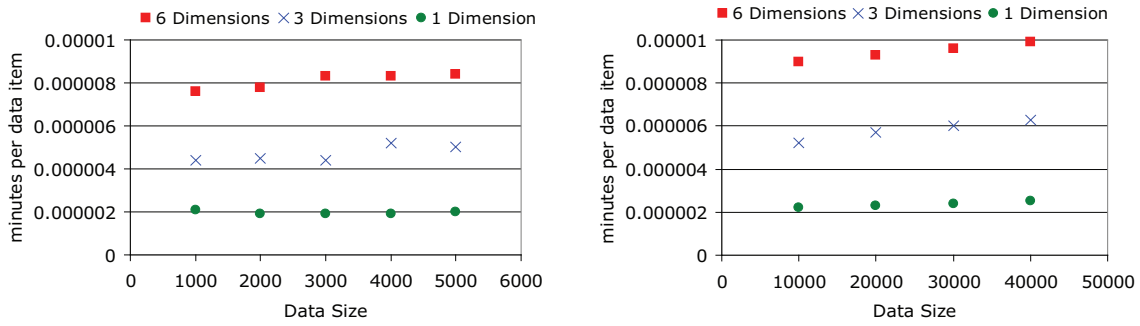


Figure 4.33: Correlated Value for Linear Curve

Data Size: 100

	1D	3D	6D
Utilization	62%	62%	62%
Tree Height	2	2	2
Total Nodes	9	9	9
Nodes on Level	L0(8) L1(1)	L0(8) L1(1)	L0(8) L1(1)
Splits	7	7	7
Adjustments	29	36	39

Data Size: 1000

	1D	3D	6D
Utilization	67%	67%	64%
Tree Height	3	3	3
Total Nodes	81	80	84
Nodes on Level	L0(74) L1(6) L2(1)	L0(74) L1(5) L2(1)	L0(77) L1(6) L2(1)
Splits	78	77	81
Adjustments	201	821	1018

Data Size: 10000

	1D	3D	6D
Utilization	64%	66%	64%
Tree Height	4	4	4
Total Nodes	831	811	831
Nodes on Level	L0(771) L1(55) L2(4) L3(1)	L0(749) L1(57) L2(4) L3(1)	L0(770) L1(60) L2(5) L3(1)
Splits	827	807	832
Adjustments	3023	12969	15356

Data Size: 40000

	1D	3D	6D
Utilization	64%	66%	64%
Tree Height	4	4	4
Total Nodes	3407	3249	3343
Nodes on Level	L0(3153) L1(2370) L2(16) L3(1)	L0(3009) L1(221) L2(18) L3(1)	L0(3085) L1(239) L2(18) L3(1
Splits	3403	3245	3339
Adjustments	15330	69170	70226

Table 4.1: RTree Summary

Table 4.1 gives a brief summary of the RTree construction process, including the height of the tree and split occurrence.

Data Trend

Fig. 4.34, Fig. 4.35, and Fig. 4.36 depict the same experiment for randomly generated data using the same event type as the $FreeDB$ database. Data attributes are generated at random and spread as much as possible to cover the possible range. The standard deviations and the

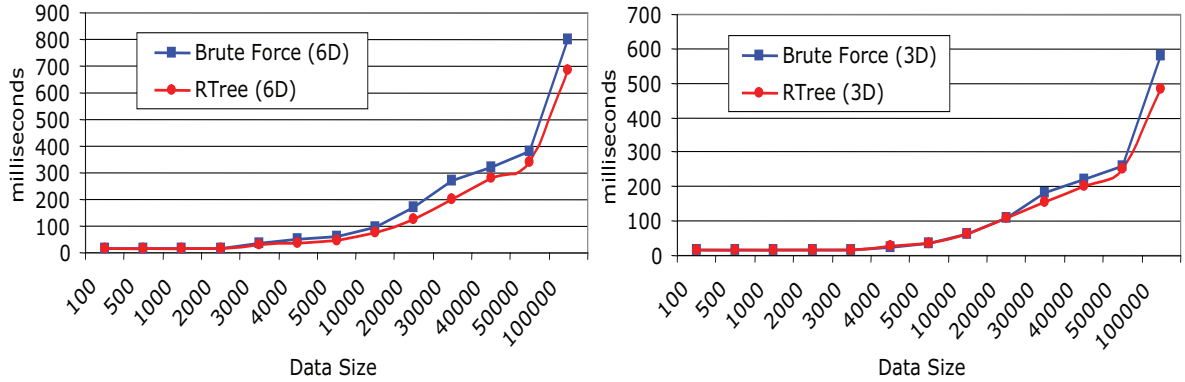


Figure 4.34: Single Range Query

confidence intervals are small, therefore error bars are not shown in Fig. 4.34, Fig. 4.35 and Fig. 4.36. The RTree construction time and storage size are similar to those in the Scoot data-based experiments.

More efficient query processing is evident with the Scoot data. This is an interesting observation, and it demonstrates that simulations based on simple random data generation may be misleading. My attempt to incorporate *Hypercube* will have a positive impact in dealing with real world data in ubiquitous computing.

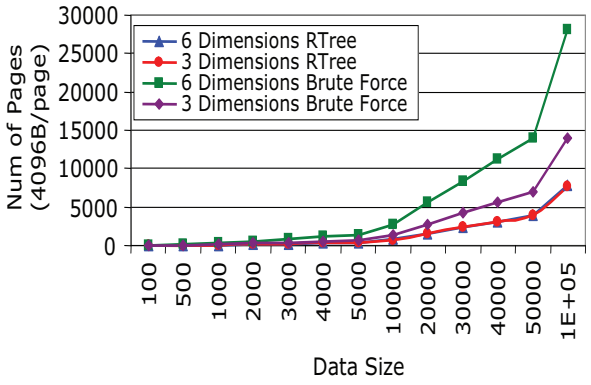


Figure 4.35: CD: Storage Usage

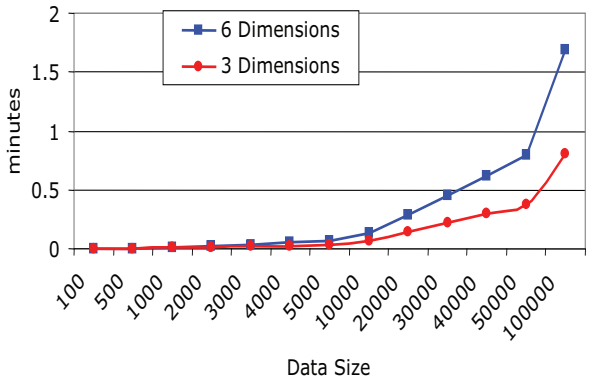


Figure 4.36: CD: RTree Construction Time

4.7.5 Discussion

The experiments highlight that RTree based indexing is effective for providing data selectivity among high volumes of data. It gives an advantage for incremental operation without the need for complete reconstruction. These experiments are not exhaustive and different trend of data may produce different result. Thus, it is worth to explore further experiments with various real world data as future work. That includes performance evaluation with a number of point queries and range queries against a point-based or range-based data set. The current experiments perform in a single thread, and the use of multi-threading may improve matching performance.

RTree indexing enables neighbourhood search, which allows similarity searches. This will be an advantage for supporting types of subscriptions that do not pose an exact question or only need approximate results. Approximation or summarisation of sensor data can be modelled using this function.

4.8 Summary and Outlook

In this chapter, I define an event model that incorporates time related issues in distributed systems and show *Hypercube* as a multidimensional indexing method for the defined event model. This approach is novel to publish/subscribe systems and has much potential. An event becomes a token for communication among applications, and the described event model and indexing will help to design accessible and traceable event-based systems.

Subscriptions for sensor data may not be defined precisely and can be ambiguous. A subscriber may not know the exact query or event they are looking for and it will be interesting to use fuzzy logic to estimate queries for such subscriptions. *Nearest neighbour query* will be an interesting property for supporting ambiguous subscriptions, where the definition of *nearest* can be interpreted in various ways.

I believe that the multidimensional access method will be a key technology to support data processing in global ubiquitous computing tasks such as load balancing, selective event dissemination, and data aggregation. Multidimensional access methods are constantly evolving. For example, in [CGR02] a new index structure for regular expressions, called RETree, is proposed. RETree is similar in concept to RTrees but handles regular expressions rather than multidimensional rectangles.

In the next chapter, I will exploit *Hypercube* in the context of publish/subscribe filtering services in P2P networks.

5

Expressive Publish/Subscribe in P2P

Searching and indexing in P2P networks share many issues of event and subscription modelling in publish/subscribe systems. Essential elements in these environments are: high volume of distributed data, many distributed queries, and continuous queries (subscriptions). An important issue is the expressiveness of queries and selectivity of data. Schema-based P2P networks potentially require query capabilities beyond a simple keyword-based search. Publish/subscribe systems have explored this issue with topic-based, content-based, and combination models. In each model, data (i.e., events) and queries (i.e., subscriptions) constrain their applications in different ways.

The P2P paradigm can integrate heterogeneous networks by grouping peers and constructing an overlay (i.e., grid) for providing specific functions. This functional overlay consists of servers (i.e., agents or brokers), which require a communication mechanism for their interactions based on application scenarios. Publish/subscribe asynchronous communication is suitable to support this paradigm. A Distributed Hash Table (DHT) gives excellent functionality for distributed indexing, especially for topic-based publish/subscribe. However, constructing content-based publish/subscribe, which provides a more selective indexing mechanism, needs to solve more complex issues such as filtering, clustering, and routing of events and subscriptions.

In this chapter, I overview P2P indexing and then discuss an importance of expressive publish/subscribe over P2P networks in wired network environments. I apply *Hypercube* event/subscription introduced in Chapter 4 on a typed content-based publish/subscribe system to exploit expressiveness of subscriptions.

5.1 Overlay Network

An overlay network is a computer network that is built on top of another network. Nodes in the overlay create logical links. For example, P2P networks are overlay networks, because they run on top of the Internet. P2P is typically used for information discovery or message exchange across a network in a decoupled manner among applications.

Most overlays form either a mesh or tree. In the mesh, more than one path between two nodes

are defined. The tree defines a single path between two nodes. An integrated approach of a mesh and a tree can be achieved by applying a mesh first, followed by a tree construction to take advantage of both.

5.1.1 Broker Overlay

Supporting distributed applications requires covering a wide area, and many systems are modelled as a set of independent servers (i.e., brokers). Brokers form an application-level overlay, and communication among brokers are through an underlying transport such as P2P networks. Permanent connections between brokers are not required, and the topology is formed by the brokers themselves. Clients can access the applications via any broker, and brokers use a distributed strategy to operate the functions such as indexing, searching, and delivering. The broker overlay network is the most common choice in publish/subscribe implementations. Examples are TIB/RV [OPSS93], Gryphon [Gry], SIENA [SIE], and JEDI [CNF98a].

5.1.2 P2P Structured Overlay

A P2P structured overlay is a self-organised, application-level network composed of a set of nodes over a virtual key space. A unique key is assigned to each node and nodes form a structured graph. This provides efficient search of data items. Dynamic node joining and leaving are handled by maintaining the overlay structure to ensure efficient communication. It is therefore more suitable for unmanaged environments such as large-scale decentralised networks.

Due to the popularity of structured overlays, several such systems have been developed. Examples are Pastry [RD01], Chord [SMLN+04], Tapestry [ZHS+04], CAN [RHK+01], and Astrolabe [vRBV03]. Structuring a publish/subscribe system over a P2P overlay network infrastructure takes advantage of the self-organisation capabilities. The event routing algorithm is realised by exploiting the communication primitives provided by the underlying overlay. Bayeux [Z+01] and Scribe [CDK+02] create topic-based systems, and Meghdoot [GSAA04], Hermes [PB03] and Rebeca [TBao03] construct content-based systems.

I look into P2P structured overlay networks in the following section.

5.2 P2P Indexing

Applications of distributed P2P networks require finding available information or objects. This can be done by a system of advertisement and queries, where resource providers advertise resource availability, while resource consumers express search queries of their needs across the network. These queries may be continuous. The advertise/continuous query model can be seen as a matching mechanism in publish/subscribe systems. In content-based search, routing the messages is based on the content of queries, where the most relevant peers receive messages. The peers can be structured according to the content for fast traversal. Content-based search techniques are deployed in content mapping overlay networks: CAN, Chord, Tapestry, and Pastry.

5.2.1 Distributed Hash Tables (DHT)

A DHT provides structure and deterministic behaviour in P2P systems. Each node is assigned a unique identifier (nodeID) from a large address space. Each data object can be mapped to a unique key associated with a nodeID. DHTs can locate a $< data\ object,\ key >$ pair based on the key identifier. Routing requires $O(log(N))$ hops to locate a data item (where N is the number of nodes in the network).

Other benefits of DHTs are scalability and self-organisation, which support the complex administration task of large network topologies. Chord, CAN, Pastry, and Tapestry use DHT technology. A node in such systems acquires an identifier based on a hash of a unique attribute such as its IP address. A key for a data object is also assigned from hashing. The DHT stores data objects as values with indices by their corresponding keys. The nodes are connected to each other in a predefined topology (e.g., a circular space in Chord, a n-dimensional Cartesian space in CAN, or mesh in Tapestry). DHT provides self-configured systems, while the peers are continuously joining and leaving. [LCP+04] provides a good survey and comparison of P2P networks. I look into more detail of Chord, Pastry and CAN in the next three sections.

Query processing over DHT is still basic, and only simple exact-match queries on unstructured data sets are currently supported.

5.2.2 Chord

In Chord [SMLN+04], a one-dimensional key-space is used and node identifiers are assigned uniformly in a circular overlay. Every node node maintains links to k closest nodes, where their id must be greater than the node id. These nodes are defined as the successor of the key. Each node also maintains the links to nodes placed further away in the key space.

5.2.3 Pastry

In Pastry [RD01], keys and node ids are sequences of 2^b digits belonging to a circular 128 *bit* identifier space, generated by the *SHA1* secure hash. A node is mapped to other nodes with numerically closest node id. In a network with N nodes and K keys, each node maintains K/N keys with high probability. Pastry's routing algorithm is based on a prefix of id, where each node forwards the message to a neighbour in the leaf set. Otherwise, the message is forwarded to the node whose id shares a common prefix of the key minimum b bits longer than the current matched prefix. This routing algorithm makes routing performance in $log_{2^b}N$ ($O(logN)$) hops.

5.2.4 CAN

Content Addressable Networks (CAN) [RFH+01] creates a logical d-dimensional Cartesian coordinate space. Each node has assigned a region within the space. Geographical proximity in the d-dimensional space defines neighbouring relations. Every node has d neighbours and each zone is addressed with a virtual identifier, which is deterministically calculated from the location of the zone. Zones are partitioned or merged depending on node-join and node-leave. On average each node has $2d$ neighbours. A node forwards the message to the closest neighbour based on the Euclidean distance (Fig. 5.1). Each node maintains $O(d)$ states, and on average $(dN^{1/d})/4$ hops are required (where N as the total number of nodes) for a message to reach its destination.

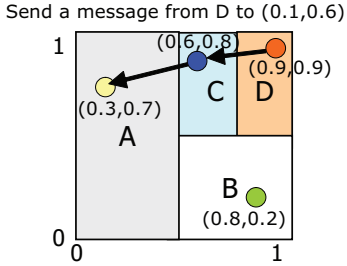


Figure 5.1: Message Routing in CAN

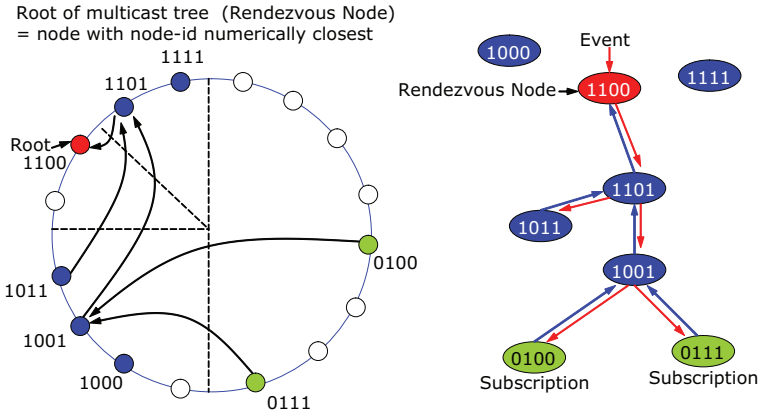


Figure 5.2: Scribe Architecture

Extensions to CAN have been proposed to enable range queries. A Hilbert space filling curve is used to map an attribute value to a location in the CAN key space, which is integrated with DHT hashing function. This approach enables mapping between the attribute proximity and corresponding key space proximity.

5.3 Overlay Multicast and Publish/Subscribe

The application level of multicast (i.e., overlay multicast) and publish/subscribe systems over P2P networks are discussed in Chapter 2.

There are two distinct approaches to routing algorithms for multicast: divide-and-conquer (e.g., Chord, Pastry and Tapestry) and a Cartesian hyper-space (e.g., CAN). Two basic methods for implementing multicast in these approaches are available, tree-based and flooding. The tree-based approach uses a single overlay building a tree for each multicast group in hierarchy. The flooding approach creates individual overlays for each group and first routes messages to each group then broadcasts them to each node within the overlay.

Scribe [CDK+02] [Cas03] is built on Pastry [RD01], while Bayeux [Z+01] is built on Tapestry [ZHS+04]. Both use unicast routing based on prefix-routing and take advantage of proximity neighbour selection mechanisms on the underlying physical network. In this section, several P2P DHT-based multicasts are described in more detail highlighting the characteristics of each approach.

5.3.1 Scribe

Scribe [CDK+02] is an application level of multicast built on top of Pastry [RD01]. Scribe assigns a unique group identifier to each topic, and the rendezvous node is selected from the node identifier that is numerically closest to the group identifier. For each topic, a multicast tree is rooted at the rendezvous point created by the diffusion of paths from subscribers. It accomplishes $(2^b - 1)log_{2^b}N$ states at members and $O(log_{2^b}N)$ application level hops between members.

In Fig. 5.2, routing mechanisms of Scribe are shown, where the node have 4 bits addresses for simplicity. The node *1100* has the value of the hashed topic value, and it becomes a root of the multicast tree. Two subscriptions for this topic (*0100 and 0111*) are forwarded to the rendezvous node (*1100*), and the reverse path to the subscriber is used for event delivery. The arrows in the ring indicate the search path for the target *1100*. The root node replicates knowledge of the topic to its k nearest neighbours for resilience.

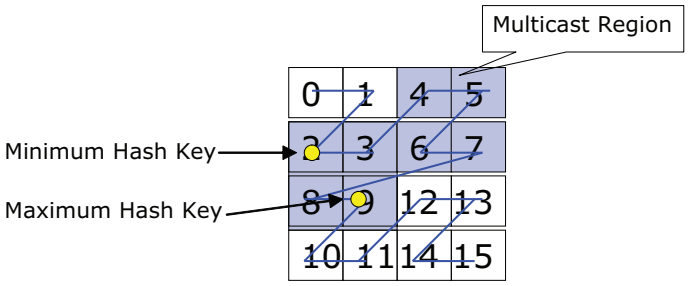


Figure 5.3: Multicast Range in Z-Curve Ordering

SplitStream [CDK+03] is an application-level multicast system based on Scribe. SplitStream deploys splitting content into k stripes and maps each stripe to a Scribe multicast tree. This approach multiplies the use of high bandwidth for data dissemination. However, [BBR+05] reports that the maintenance cost of multicast trees is non-trivial (40% or more) due to non-DHT links in Scribe/SplitStream.

5.3.2 CAN multicast

CAN-based multicast [RHK+01] is an application level multicast system. It forms a separated CAN for each multicast group and floods data over the CAN of the target group.

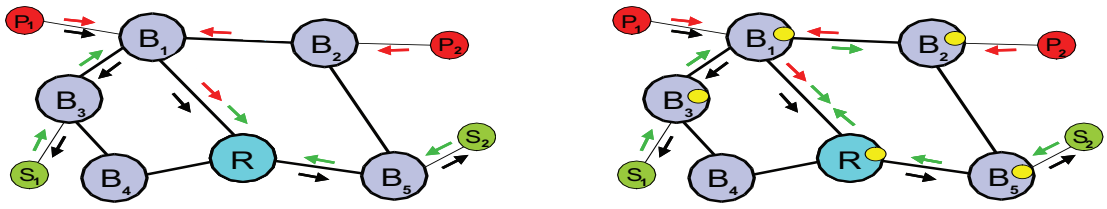
In [jxt], Z-ordering [OM84] is used for the implementation of CAN multicast. Z-ordering interleaves the bits of each value for each dimension to create a one-dimensional bit string. Thus, Z-ordering can be used to convert one-dimensional ordering into multidimensional ordering or vice versa.

Z-ordering ensures the locality preservation in identifiers (i.e., same high order bits), and similar identifiers are mapped to the same region. Fig. 5.3 shows z-ordering in 2-dimensional space.

5.3.3 Hermes

Hermes [PB02] is a typed content-based publish/subscribe system built over Pastry. The basic mechanism is based on the rendezvous mechanism that Scribe uses [CDK+02]. Additionally, Hermes enforces a typed event schema providing type safety by type checking on notifications and subscriptions at runtime. The rendezvous nodes are chosen by hashing the event type name. Thus, it extends the expressiveness of subscriptions and aims to allow multiple inheritances in event types.

The content-based publish/subscribe routing algorithm is an adaptation of SIENA [SIE] and Scribe using rendezvous nodes. Both advertisements and subscriptions are sent towards the ren-



(a) Type-based Routing (b) Content-based Routing

Figure 5.4: Routing for Publish/Subscribe in Hermes

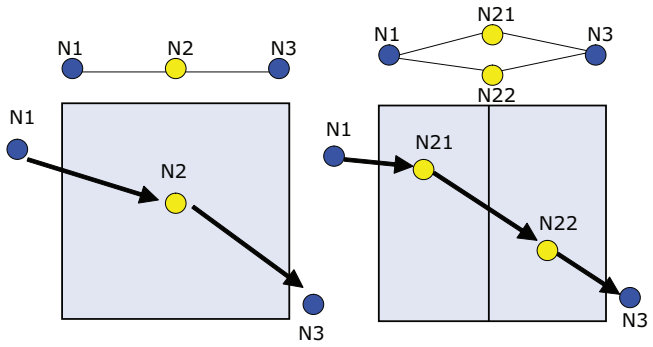


Figure 5.5: Splitting a space for load balancing in Meghdoot

dezvous node, and each node on the way keeps track. Routing between the publisher, where the advertisement comes from, and the subscriber is created through this process. An advertisement and subscriptions meet in the worst case at the rendezvous node. The event notification follows this routing, and the event dissemination tree is therefore rooted from the publisher node. This will save some workload from the rendezvous nodes. However, this approach is only advantageous if the rendezvous nodes are distant from either publisher nodes or subscriber nodes. If publisher and subscriber nodes are close, a more direct routing mechanism from publishers to subscribers can be established, which is SIENA's original approach.

Hermes achieves $logN$ routing for both subscription and notification propagation, with approximately $logN$ replication in the worst scenario. Fig. 5.4 shows routing mechanisms for type-based and content-based publish/subscribe. Arrows are red for advertisements, green for subscriptions, and black for publications. The black arrow from broker 1 to broker 3 shows a shortcut to subscriber 1 that is different from the routing mechanism of Scribe. Subscription 2 in content-based routing travels up to the broker hosting the publisher Fig. 5.4(b). Yellow circles indicate where filtering states are kept.

5.3.4 Meghdoot

Another P2P system enabled over DHTs is Meghdoot [GSAA04]. Meghdoot is based on CAN [RFH+01] and constructs a CAN space of dimension $2k$ for subscription with k attributes. Subscriptions are assigned to a corresponding CAN space and stored at the space owner node. In Meghdoot, the overlay dimensionality is determined by the number of attributes in subscriptions. The matching operation between publications and subscriptions traverses all potential matching regions. The routing load is balanced by replication of the zone. In Fig. 5.5, the node $N2$ is overloaded and two nodes ($N21$ and $N22$) replicate the load of the original $N2$. One of two nodes can be the original $N2$. Meghdoot does not split the load between these nodes and reducing the load is realised with creating alternate event propagation paths to its neighbours.

5.3.5 Discussion

Topic-based publish/subscribe is realised by a basic DHT-based multicast mechanism [ZHS+04], [Z+01], [RHK+01]. More recently, some attempts on distributed content-based publish/subscribe systems based on multicast become popular [BCM+99], [CRW01], [TBao03], [TE04].

An approach combining topic-based and content-based systems using Scribe is described in [TAJ03]. In these approaches, the publications and the subscriptions are classified in topics using an appropriate application-specific schema. The design of the domain schema is a key element for system performance, and managing false positives is critical for such approach.

Event filtering in content-based publish/subscribe can provide better performance if similar subscriptions are in a single broker or neighbour brokers. Physical proximity provides low hop counts per event diffusion in the network with a content-based routing algorithm [MFB02]. If physical proximity is low, on the other hand, routing is becoming similar to simple flooding or unicasting.

Filtering and matching queries on range queries are recently getting attention in P2P publish/subscribe systems. I envision that CAN-based approaches (e.g., Meghdoot) with Z-ordering is a promising direction.

5.4 Expressiveness of Subscription

Data from WSNs can be multidimensional and searching for these complex data may require more advanced queries and indexing mechanisms than simple hashing values to construct DHT so that multiple pattern recognitions and similarities can be applied. Subscribing to unstructured documents that do not have a precise description may need some way to describe the semantics of the documents. Another aspect is that searching by DHT requires the exact key for hashing, while users may not require exact results. This section discusses the expressiveness of query and subscription.

5.4.1 Flexible DHT

DHT mechanisms contain two contradictory sides: the hash function distributes the data object evenly within the space to achieve a balanced load, whereas the locality information among similar subscriptions may be completely destroyed by applying a hash function. For example the current Pastry intends to construct DHT with random elements to accomplish load balance. Nevertheless similarity information among subscriptions is important in publish/subscribe systems.

DHT needs more advanced built-in indexing mechanisms. It requires locality preserving key generation so that the similarity search can be performed as a spatial information query. Spatial locality information should be preserved for range queries. There have been some extensions to DHT such as the use of trigrams for text retrieval, bloom filters with hash-based AND, or feature vectors for multimedia. However, extensions are for application specific purposes. What would be required here is indexed DHT, where DHT can contain a secondary key or hierarchical structure. Thus, a potential relational data structure on DHT would be ideal.

5.4.2 Hierarchical topic coordination

P2P overlays can be used to implement Internet-scale application level multicast with pre-defined multicast channels. These channels can map to the name space, which can be divided into subgroups. For example, the domain or name space "shop.com" can be divided into groups such as uk.shop.com or ch.shop.com, which can be structured in the hierarchy. Many publish/subscribe systems provide hierarchical topics. However, content-based publish/subscribe settings are dynamic, and pre-assigned channels are not useful.

Type-based publish/subscribe automatically provides a hierarchical structure and multiple subtyping, thus avoiding explicit message classification through topics.

Another approach is translating a content-based subscription to a topic-based subscription that constructs a topic hierarchy [TAJ03]. After creation of the dissemination tree, it can simply use multicast mechanisms. To automatically build such topics, it is essential that the content pro-

vided by the user application follows rules or a *template*. This realises a content-based distributed P2P DHT-based publish/subscribe system over a topic-based system.

5.4.3 Clustering Subscriptions

[TAJ03] introduces a partition based on the content of notifications, which selects certain attribute combinations from the scheme and creates a pre-configured sub-channel. This indexing approach is commonly known from database systems. It combines high selectivity with good performance. However, for some subscriptions there may be no matching index.

Subscription summary is related to clustering subscriptions, which significantly reduces the complexity of the system [TE04]. The number of multicast groups is often limited in a system, and this causes unnecessary event dissemination to the receivers. Clustering techniques can be used for classify events with similar sets of receivers to reduce extraneous traffic. A difficulty of this approach is that effectiveness of clustering is dependent on the distribution of events and subscriptions.

5.4.4 Range Query

A DHT is not suited for range queries, which makes it hard to build a content-based publish/ subscribe system over a structured overlay networks. When the subscription contains attributes with continuous values, it becomes inefficient to walk through the entire DHT entries for matching.

Range queries are common with spatial data and desirable in geographic-based applications of ubiquitous computing, such as queries relating to intersections, containment, and nearest neighbours. However, DHT mechanisms in most of the current structured overlay distribute data uniformly without spatial locality. This therefore prevents supporting spatial queries efficiently but only through an exhaustive search. Range queries introduce new requirements such as data placement and query routing in distributed publish/subscribe systems.

Recently, several proposals have been made to extend P2P functionality to more complex queries (e.g., range queries [GAA03] [SGAA04], joins [HHH+02], XML [GWJ+03]). [GS04] describes the Range Search Tree (RST), which maps data partitions into nodes. Range queries are broken to sub-queries corresponding to nodes in the RST. Data locality is obtained by the RST structure, which allows fast local matching. However, queries are broken down into sub-queries, which makes the matching process complex. [HRS03] uses prefix hash trees, which hash common prefixes into multi-way retrieval trees, to support range queries. [SP03] uses the Hilbert Space-Filling Curve [Bia67] in the index space for partial keywords and range queries. In [LHL05], the use of a locality preserve hash function is introduced that provides capability of approximate range selection queries.

The database community provided various solutions for processing multidimensional queries, but adapting these solutions to the P2P indexing faces difficult issues. In [GYGM04], four challenges are described as follows:

- Distribution: Data needs to be partitioned across a large number of nodes while ensuring both load balance across nodes and efficient queries.
- Dynamism: Nodes in a P2P system may join and leave frequently. Therefore, the data partitioning needs to be over a dynamic set of nodes while retaining good balance and efficient queries.

- Data Evolution: Data distributions may change over time and can cause load imbalance even if nodes remain stable. Thus, data may need to be frequently re-partitioned across nodes to ensure load balance.
- Decentralisation: P2P systems do not have a central site that maintains a directory mapping data to nodes. Instead, a query submitted at any node must be efficiently transmitted to the relevant nodes by forwarding the query along an overlay network of nodes.

5.4.5 Semantic-Based Query

In RDF schema-based P2P networks, more than simple key and keyword-based queries are needed. RDF and RDFS [L+99] [BG03] can be used to annotate resources, and this provides means for exchanging information between systems. Metadata can also be used for describing properties of resources, where resource can be distributed and described in other metadata standards. This makes it possible to construct distributed repositories. RDFS can represent schema, properties, and property constraints to define the vocabulary used for describing the resources. RDF schema is flexible and extensible and can evolve over time. An example of the use of W3C metadata standards and RDF schema [L+99] [BG03] to describe distributed resources are provided in the JXTA framework [Gon01].

Our work [YB04c] demonstrates the case for using RDF as a common schema to realise the federation of various publish/subscribe systems.

5.5 Hypercube Publish/Subscribe

This section presents *Hypercube Publish/Subscribe* for content-based publish/subscribe, which is based on multidimensional data representation in a *Hypercube*, as described in Chapter 4. This uses geometrical intersection of publications/subscriptions represented in hyper rectangles in a multidimensional event space. This will provide selective data dissemination in an efficient manner including *symmetric publish/subscribe.* I present an extension to a typed content-based publish/subscribe system (i.e., Hermes) with *Hypercube* filtering (i.e., n-dimensional indexing).

The visualisation of topology (shown throughout the experiment section) is added to Hermes.

5.5.1 Hypercube Event Filter

In content-based networks such as SIENA [CRW04], the intermediate server node creates a forwarding table based on subscriptions and operates event filtering. Under high event publishing environments, the speed of filtering based on matching the subscription predicates at each server is crucial for obtaining the required performance.

The channelisation approach is described in [AGK+01]. A limited number of multicast trees are deployed to reduce unnecessary message delivery. In [RLW+02], [CS04], [PC05], [WQA+02], subscriptions are clustered to multicast trees. Thus, filtering is performed at both the source and receiver nodes. In contrast, the intermediate nodes perform filtering for selective event dissemination in [OCD00], [SRD02]. This approach is also operated in Hermes in attribute-based routing. In Hermes, a route for event dissemination for a specific event type is rooted at the publisher node through a rendezvous node to all subscribers by constructing a diffusion tree. The intermediate broker nodes operate filtering for content-based publish/subscribe. The filtering mechanism is primitive, with each predicate of the subscription filter being kept independently without any aggregation within the subscriber edge broker. The coverage operation requires a comparison of each predicate against an event notification.

The *Hypercube* event model is integrated to subscription filters to provide efficient matching and coverage operations. In the experiments, the effectiveness and expressiveness of typed channels and filtering attributes are compared. The advantages of this approach include efficient range query and filter performance (resource and time). The balance between typed-channel and content-based filtering is a complex issue. In existing distributed systems, each broker has a multi-attribute data structure to match the complex predicate for each subscription. The notion of weak filtering for hierarchical filtering can be used as summary-based routing (see [WQV+04] and [EFGH02]), so that the balance between the latency of the matching process and event traffic can be controlled. When highly complex event matching is operated on an event notification for all subscriptions, it may result in too high message processing latency. This prevents reasonable performance of publishing rates to all subscribers. The subscription indexing data structure and filter matching algorithm are two important factors to impact the performance in such environments including filter coverage over the network. For example, online stock traders sometimes need to handle over 50,000 messages per second. A large number of customers may subscribe to these data.

5.5.2 Locality Preserved String Hash

Locality preserving string hashing techniques such as [LHL05] can be used as the string-based predicate value for *Hypercube* filtering. This will enable appropriate nearest neighbour matching or range queries. It will help to exploit approximate matching mechanisms when exact matching is not an issue. In [RRHS04], a distributed structure called Prefix Hash Tree (PHT) is proposed, which is built over a DHT P2P network supporting range queries and prefix string queries. The complexity of processing range and string queries results in low performance in the PHT-based approach. PHT exhibits a message complexity of $O(l \times log(N))$, similar to [AT05], given that one DHT lookup is needed per character of the string at length l.

5.5.3 Type Name

The current strategy of type implementation in Hermes is a predefined type definition, universal in the name space (e.g., XML schemata). The type safety mechanism depends on using an actual data structure such as XML schema.

The rendezvous nodes forward the publication to the rendezvous nodes of the super-types so that subscribers that subscribe to super-types can obtain event notifications. However, if there is any support for locality-preserving string hashing, it automatically preserves similarity among subscriptions. It will help nontype-based (e.g., keyword or meta-data-based approaches) and hierarchical topic-based publish/subscribe. Note that applying locality awareness to DHT may destroy the load balancing discussed.

5.6 Experiments

The filtering function of Hermes publish/subscribe is extended with an n-dimensional hypercube with RTree described in Chapter 4. First, the event filter is converted to *Hypercube*. The subscriber edge broker can use all subscriptions with less storage, and *Hypercube* filters give flexible range query capability. Event type integration within the filter, as a content-based approach, is experimented in Section 5.6.6.

The main goal of the experiment is to demonstrate a selective and expressive event filter that can be used to provide flexibility to explore the subscriptions. The performance of scalability

issues in Hermes is reported in [PB02] and general control traffic (e.g., advertisement, subscription propagation) are also reported in [RD01].

Thus, to keep the results independent of secondary variables, only the message traffic for the dissemination of subscriptions is therefore measured, and the focus of the experiment is the efficiency of the *Hypercube* filter in this section. The metrics used for the experiments are the number of publications disseminated in the publish/subscribe system. The number of hops in the event dissemination structure varies depending on the size of the network and the relative locations between publishers and subscribers.

5.6.1 Experimental Setup

The experiments are run on FreePastry [RD01], a Pastry simulator. The experiments are carried out on a single computer: a Pentium-4 1GHz machine. Publishers, subscribers and rendezvous nodes are configured with deterministic *node ids*, and all the other brokers get *node ids* from Pastry simulations.

One thousand Pastry Nodes are deployed. All pastry nodes are considered as brokers, where the Hermes event broker function resides, and the total number of nodes (N=1000) gives average hop counts from the source to the routing destination as $log_{2^4}(1000) \approx 2.5$, where 4 is given as configuration value. Eight subscribers connect to the subscriber edge brokers individually. One publisher publishes all the event notifications for convenience. The subscriptions are listed in Fig. 5.6. 1000 publications are randomly created for each event type.

The experiments have been repeated at least 5 times. Note that the simulator over FreePastry creates the network topology in a non-deterministic way. The standard deviations are ≤ 0.32, and the confidence intervals for the population mean are ≤ 0.20 at 95% confidence level in all experiments. These values do not impact the experimental results shown in this chapter. These measurements are highly reproducible (see Section 4.7.4.3). Thus, no error bars are shown in the figures in this chapter.

This is a relatively small scale experiment, but considering the characteristics of Hermes, where each publisher creates an individual tree combining the rendezvous node, the experiment is sufficient for evaluation.

Subscriptions and publications

A single type *CD* with two attributes (i.e., *released year* and *ranking*) are used for the content-based subscription filter. In Fig. 5.6, eight subscriptions are defined with different ranges on two attributes. The publications take the form of a point for the *Hypercube* RTree. Four different publications are defined and 250 instances of each publication are published: 1000 event notifications are processed in total. Same sets of publications and subscriptions are used in all experiments unless stated otherwise.

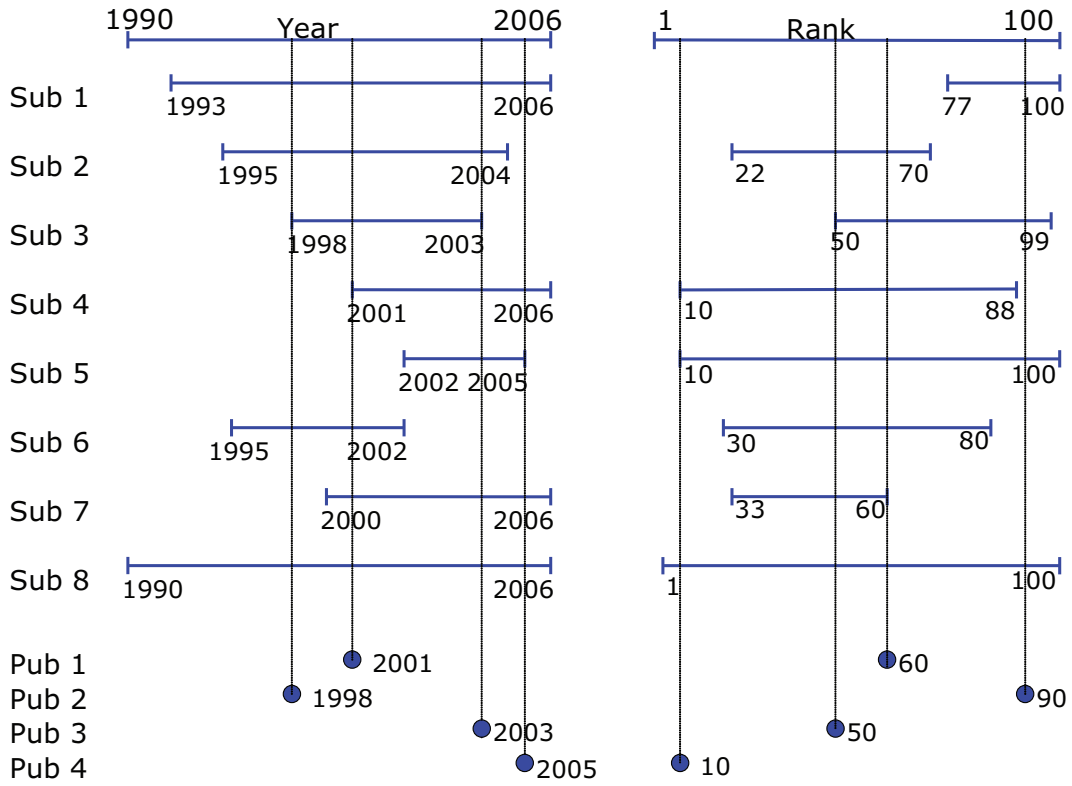


Figure 5.6: Subscriptions and Publications

5.6.2 Hypercube Event Filter

This experiment demonstrates the basic operation of *Hypercube* event filters. Fig. 5.7 shows the logical topology consisting of 8 subscribers, a publisher, and a rendezvous node along router nodes. Identifiers indicate the addresses assigned by the Pastry simulation. Fig. 5.8 shows the number of publications delivered to the subscribers. Fig. 5.8b shows these data with *Hypercube* filtering, while no filtering has been applied in Fig. 5.8c (e.g., Scribe). The X axes in Fig. 5.8b and Fig. 5.8c indicate the location of subscribers mapping to the Fig. 5.8a, where the logical topology is transformed from Fig. 5.7. The Y axes indicates the number of publications received at the node in all the experiments.

This coordination of figures applies on Fig. 5.11, Fig. 5.13, Fig. 5.14, Fig. 5.15, Fig. 5.18, Fig. 5.19, Fig. 5.20, Fig. 5.21, and Fig. 5.23.

Fig. 5.9 depicts the matching publication rate for each subscriber node. With hypercube filters, there are no false positives and subscribers receive only matching publications. It is obvious that filters help to control the traffic of event dissemination. Drawbacks are the processing time of *Hypercube* and the increased data size in the packet, as shown in Chapter 4.

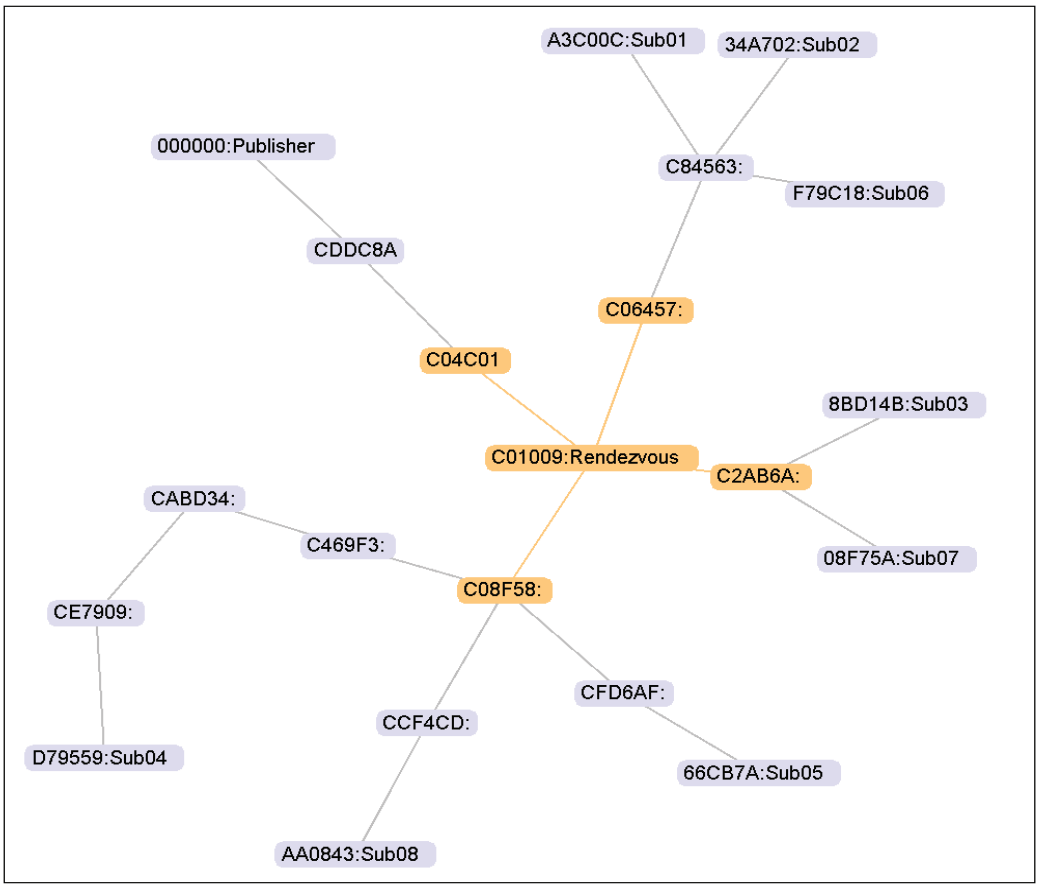
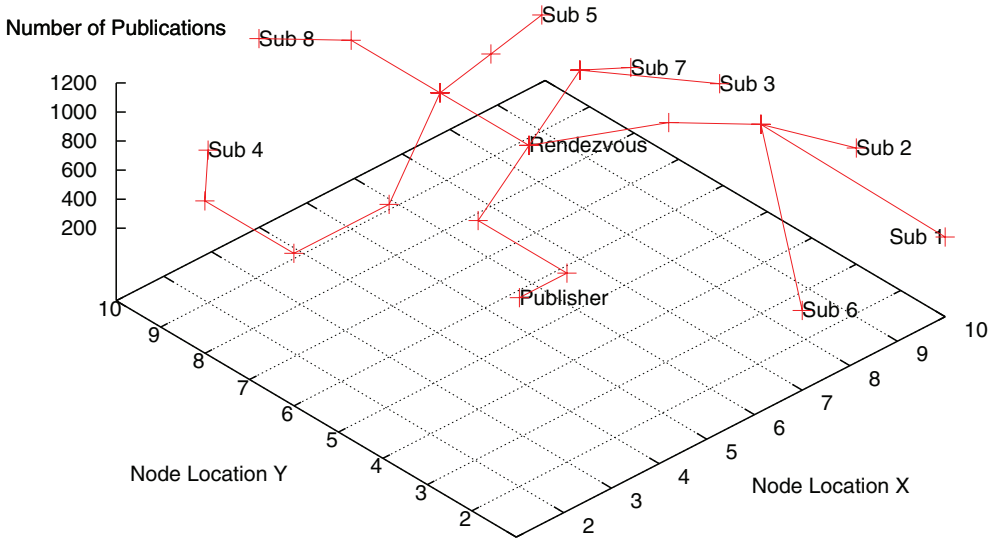
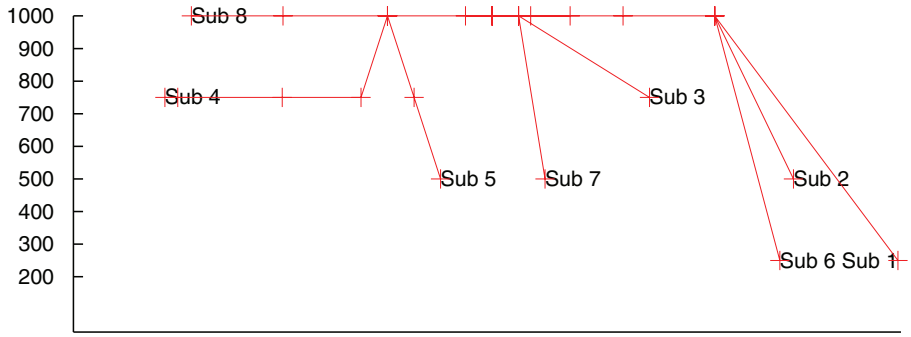


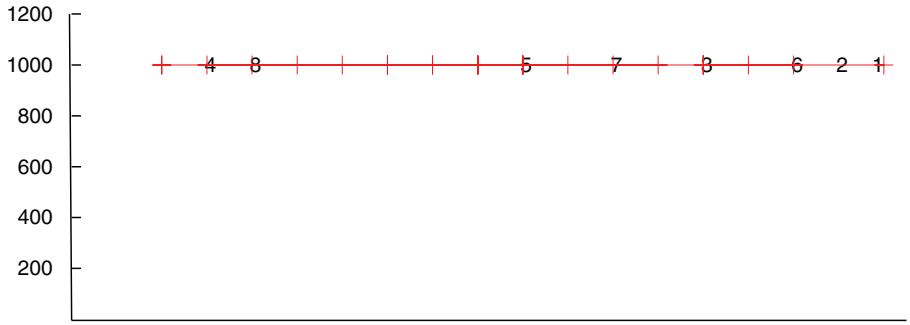
Figure 5.7: Publish/Subscribe System over Pastry



a. Logical Topology



b. Delivered Publications to Subscribers with *Hypercube*



c. Delivered Publications Subscribers w/o Filtering (Scribe)

Figure 5.8: Event Traffic and Filtering: Basic

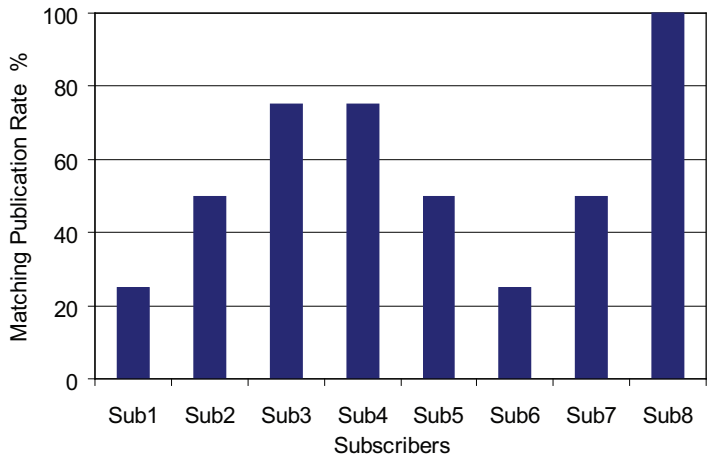


Figure 5.9: Matching Publication Rate with No Filtering (Basic)

5.6.3 Random Generation of Events

In this experiment, 1000 events are created at random for publications. Fig. 5.10 shows the topology and Fig. 5.11 shows the number of publications delivered to the subscribers. Fig. 5.12 depicts the matching rate among received publications for each subscriber node.

A more random generation of events, where events cover the range of filters, shows better edging results. Increased coverage of subscriptions results in high matching rates, and more selective subscription result in low matching rates. The spread is quite large. Thus, deploying publish/ subscribe systems requires careful consideration of filtering and channel/type deployment.

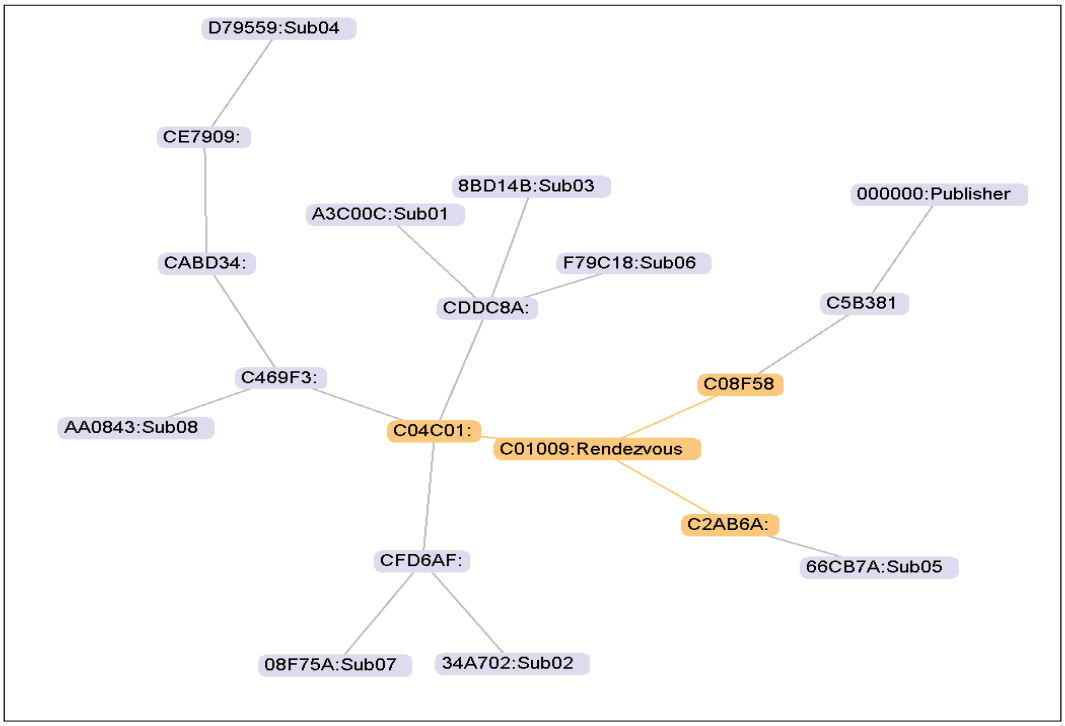
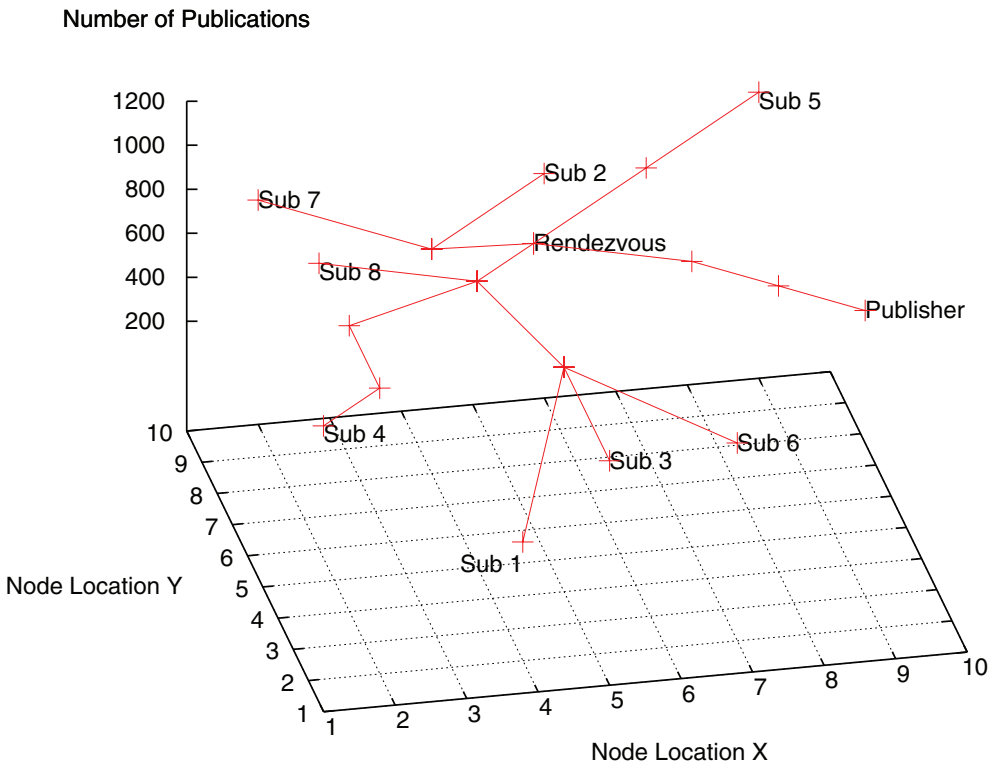
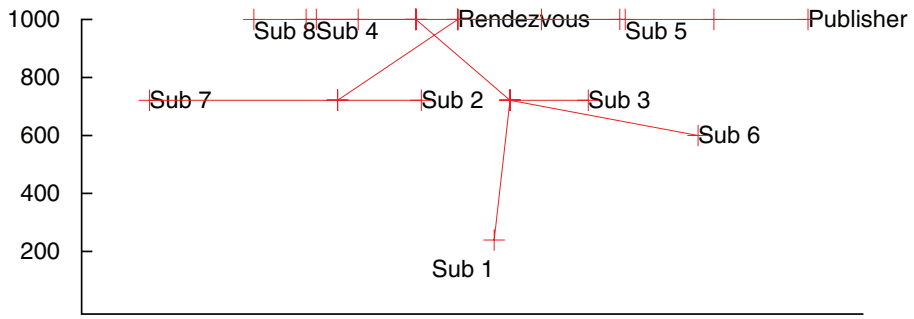


Figure 5.10: Topology of Publish/Subscribe System: Random Event Generation



a. Logical Topology



b. Delivered Publications to Subscribers

Figure 5.11: Event Traffic and Filtering: Random Event Generation

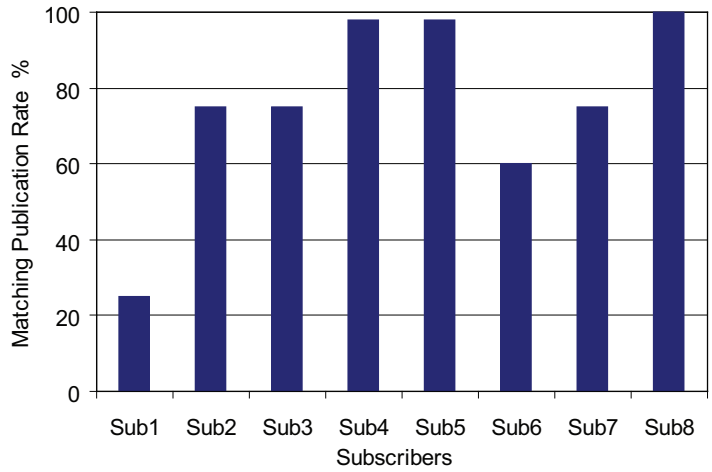


Figure 5.12: Matching Publication Rate with No Filtering (Random Events)

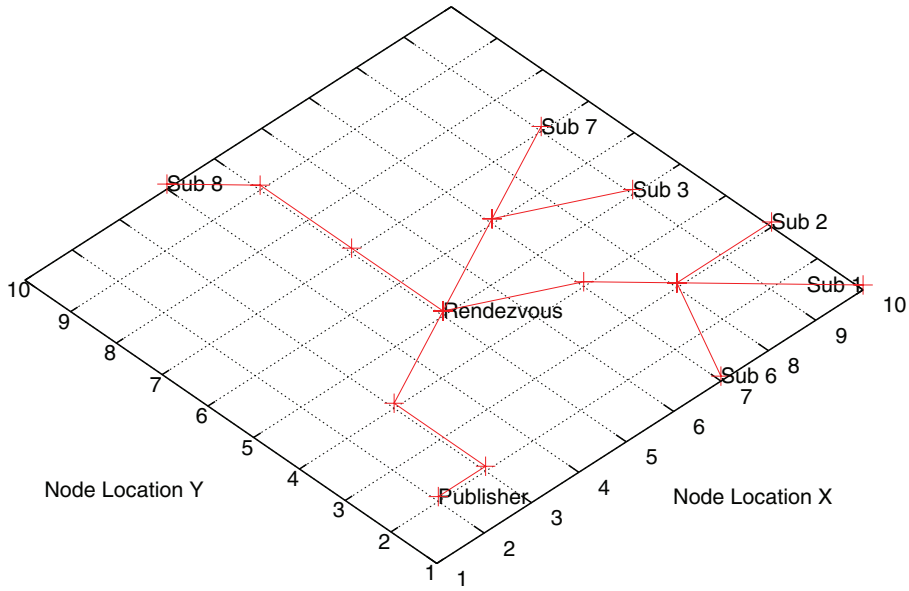
5.6.4 Predefined channels

In this experiment, publication traffic with 4 predefined channels is measured. Table 5.1 shows the defined channels, and the match with subscriptions/publications. Fig. 5.13, Fig. 5.14, and Fig. 5.15 show the topology and the traffic for each subscriber. Fig. 5.16 depicts the matching publication rates for each subscriber node.

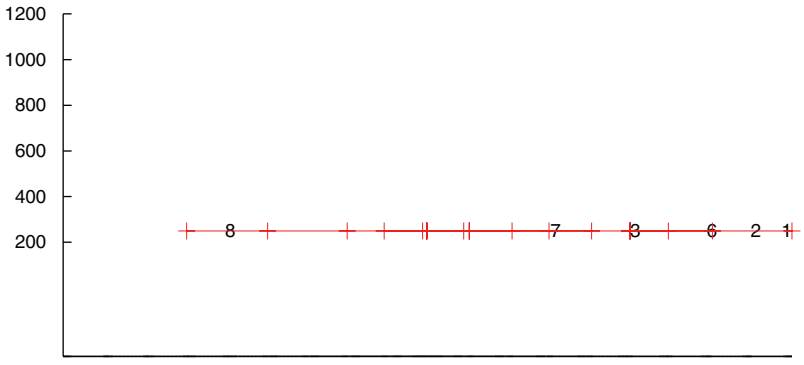
In [TAJ03], pre-configured sub-channels similar to a common database scheme are used. The experiment reports that well designed schema such as the indices used incur hit rates greater than 25% and the impact of range queries is moderate.

	Filters		Subscriptions								Publications			
Channels	Year	Rank	S1	S2	S3	S4	S5	S6	S7	S8	P1	P2	P3	P4
CH 1	1990-2000	1-50		✓				✓	✓	✓				
CH 2	1990-2000	51-100	✓	✓	✓			✓	✓	✓		✓		
CH 3	2001-2006	1-50		✓	✓	✓	✓	✓	✓	✓			✓	✓
CH 4	2001-2006	51-100	✓	✓	✓	✓	✓	✓	✓	✓	✓			

Table 5.1: Predefined 4 Channels

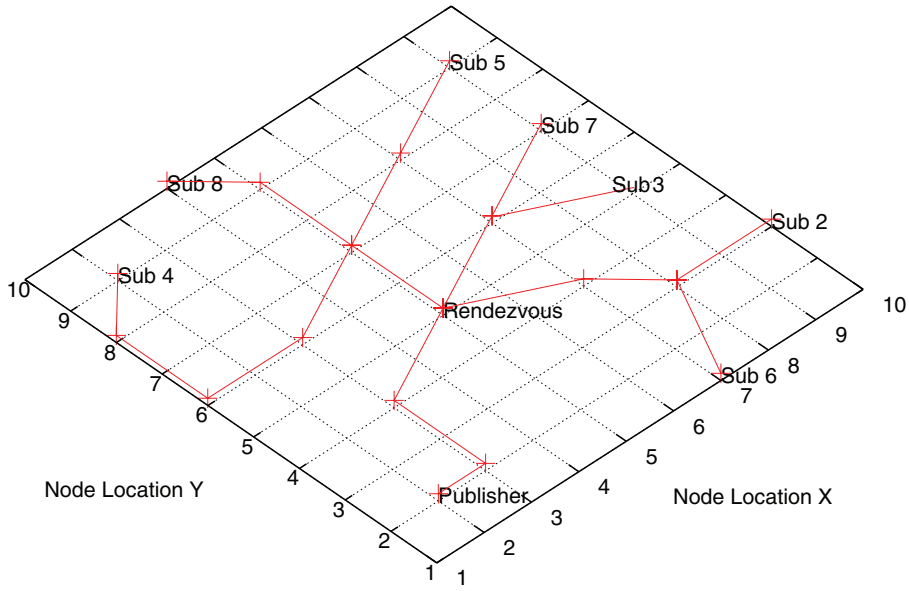


a. Logical Topology

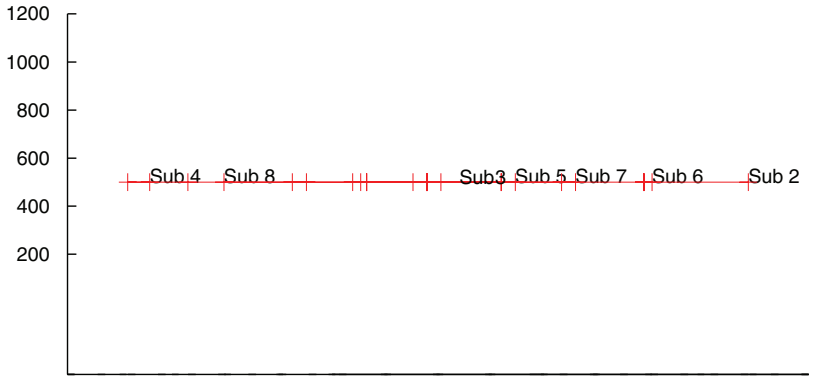


b. Delivered Publications to Subscribers

Figure 5.13: Event Traffic: Channel 2

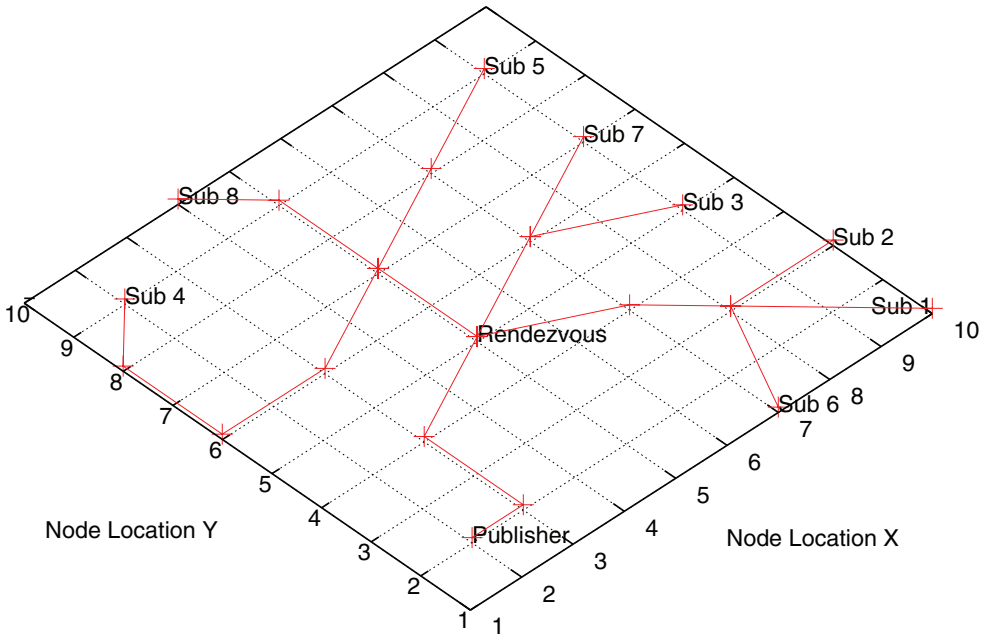


a. Logical Topology

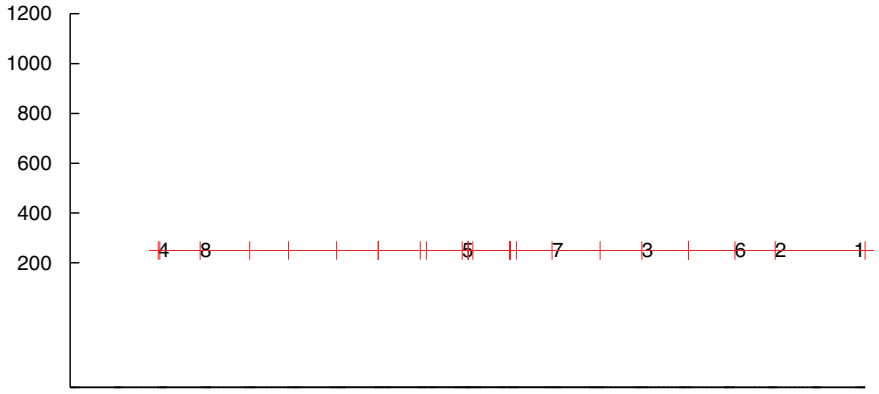


b. Delivered Publications to Subscribers

Figure 5.14: Event Traffic: Channel 3



a. Logical Topology



b. Delivered Publications to Subscribers

Figure 5.15: Event Traffic: Channel 4

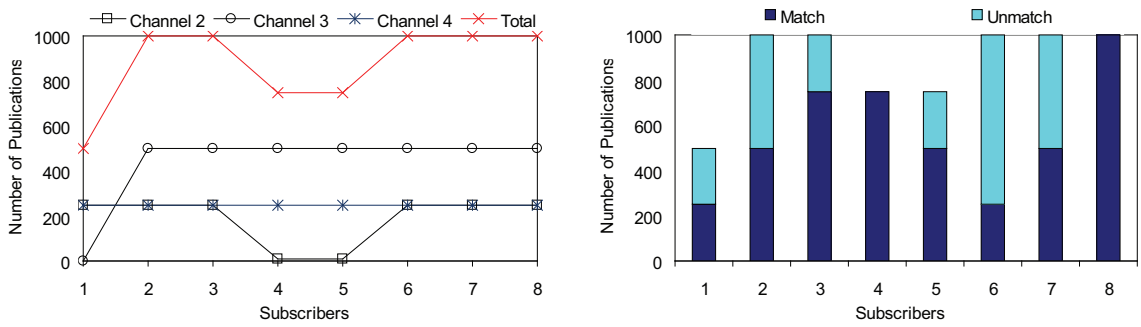


Figure 5.16: Matching Publication Rate in Channel based Publish/Subscribe

5.6.5 Multiple Types

In this experiment, 3 types are used: *Classic*, *Jazz*, and *Pop*. Thus, 3 rendezvous nodes are created. All three types share the same attributes. Table 5.2 shows the defined types along the subscriptions. The publisher publishes 1000 events for each type, 3000 publications in total. Fig. 5.17 depicts the topology of these experiments, and Fig. 5.18 shows the combined event traffic. Fig. 5.19, Fig. 5.20, and Fig. 5.21 show the event traffic for each type.

Unless there is a super type defined for three types, each type creates an independent dissemination tree and causes multiple traffic. Adding one dimension in the hypercube filter transforming from *type* will give better performance as shown in the next section.

Subscriber	Classic	Jazz	Pop
Sub 1	✓	✓	
Sub 2		✓	✓
Sub 3	✓		✓
Sub 4	✓	✓	
Sub 5	✓	✓	✓
Sub 6	✓	✓	✓
Sub 7	✓		✓
Sub 8	✓	✓	✓

Table 5.2: 3 Types and Matching Subscriptions

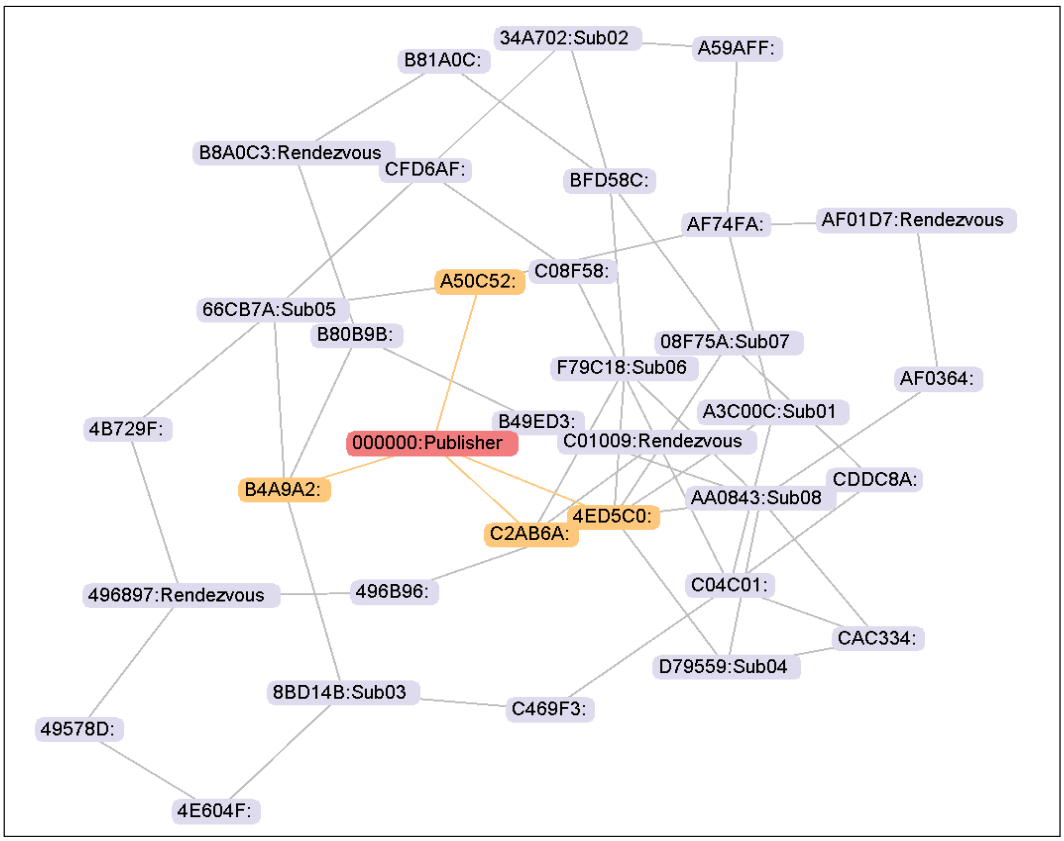


Figure 5.17: Publish/Subscribe System with 3 Types

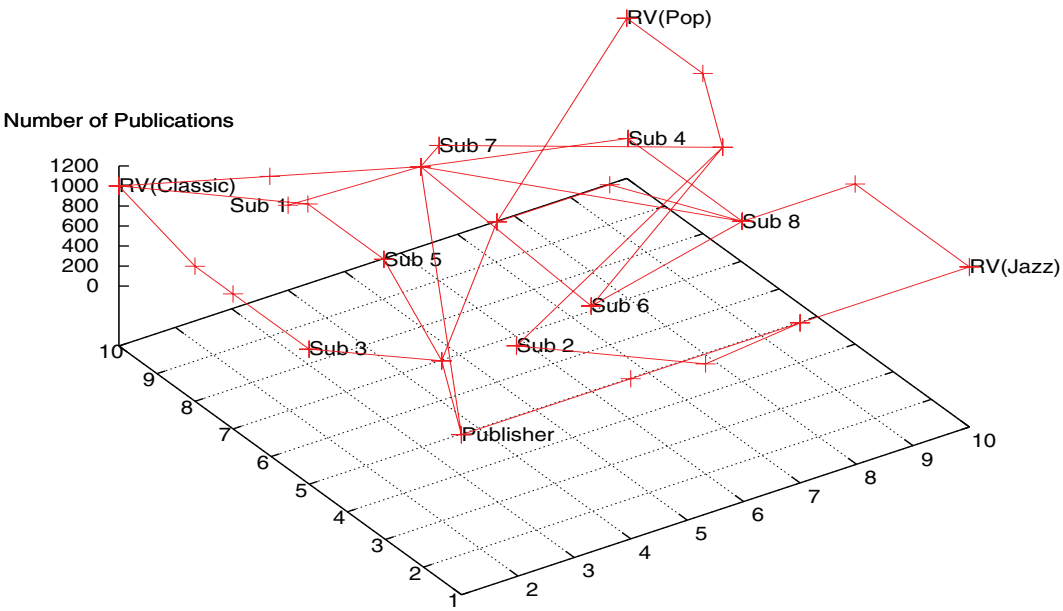
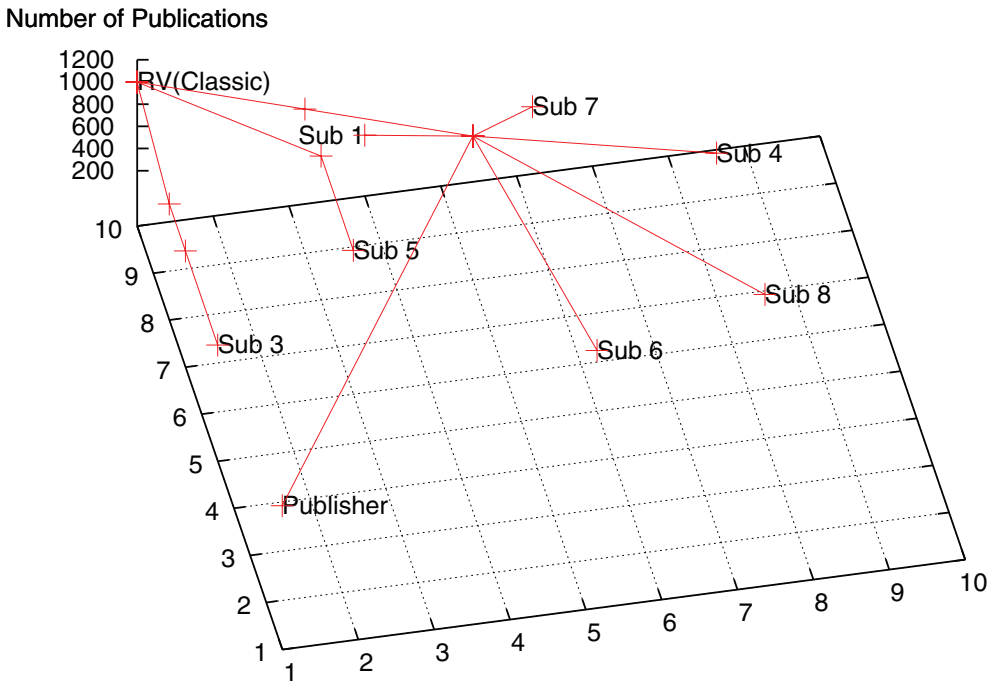
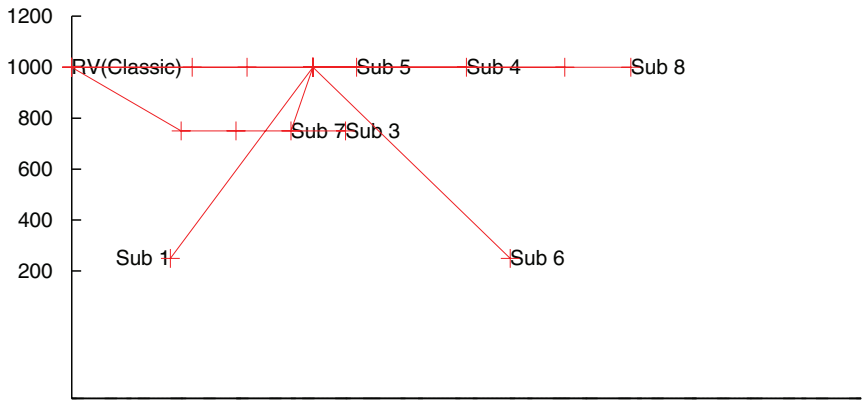


Figure 5.18: Combined View

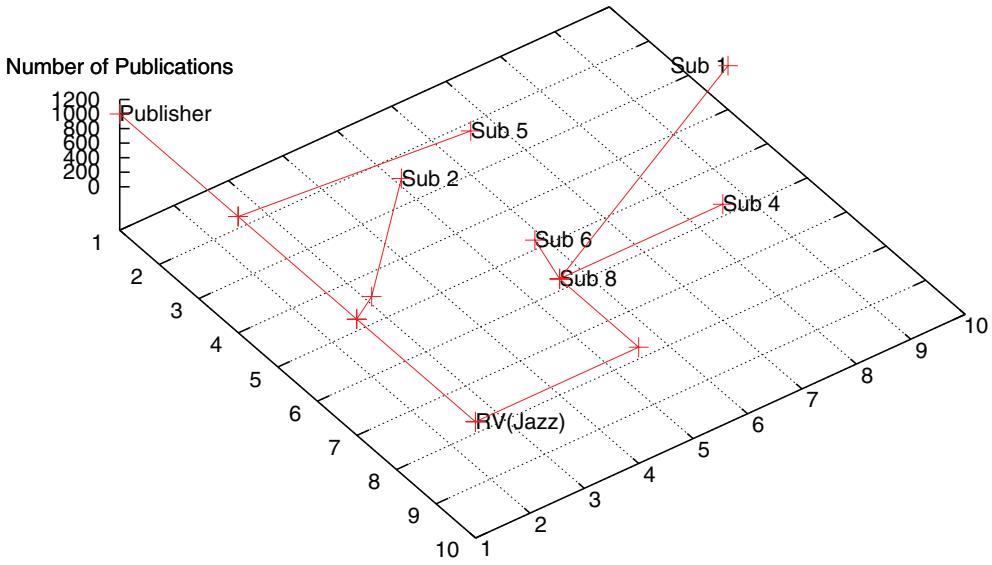


a. Logical Topology

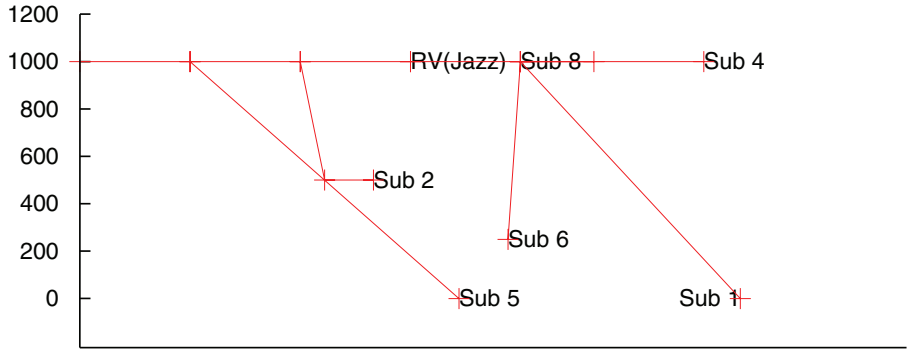


b. Delivered Publications to Subscribers

Figure 5.19: Event Traffic: Type Classic

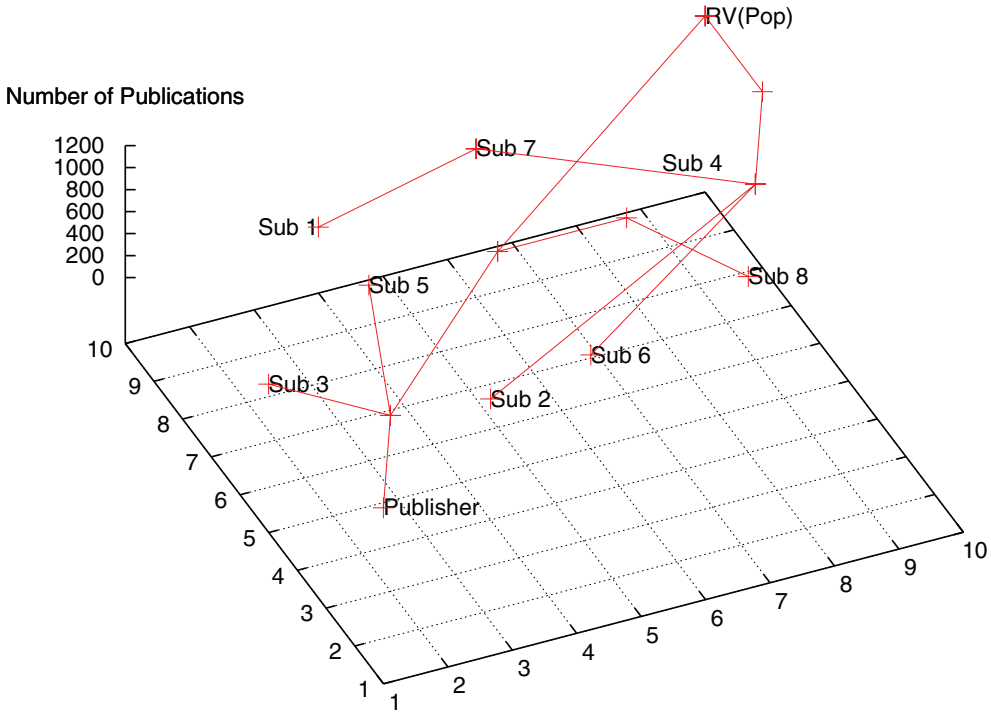


a. Logical Topology

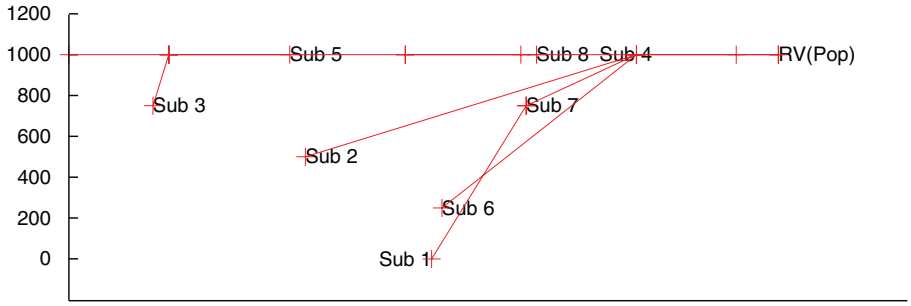


b. Delivered Publications to Subscribers

Figure 5.20: Event Traffic: Type Jazz



a. Logical Topology



b. Delivered Publications to Subscribers

Figure 5.21: Event Traffic: Channel Pop

5.6.6 Additional Dimension as Type

In this experiment, instead of using multiple types, an additional dimension is added to the hypercube filter. Fig. 5.22 depicts the topology and only one rendezvous node appears, while three appear in Fig. 5.17. Fig. 5.23 and Fig. 5.24 depicts the event traffic. The apparent result shows significant improvement of the traffic with the additional dimensional approach.

When different event types are used, which are not hierarchical, separated route construction for each event type is performed for event dissemination. Different types, which may contain the same attributes, may not have a super type. Also super types may contain many other subtypes, of which the client may not want to receive notifications. Thus, additional dimensions on the filtering attributes may be a better approach for flexible indexing. Transforming the type name to the dimension can preserve locality, similarity or even hierarchy. This will provide an advantage for neighbour matching.

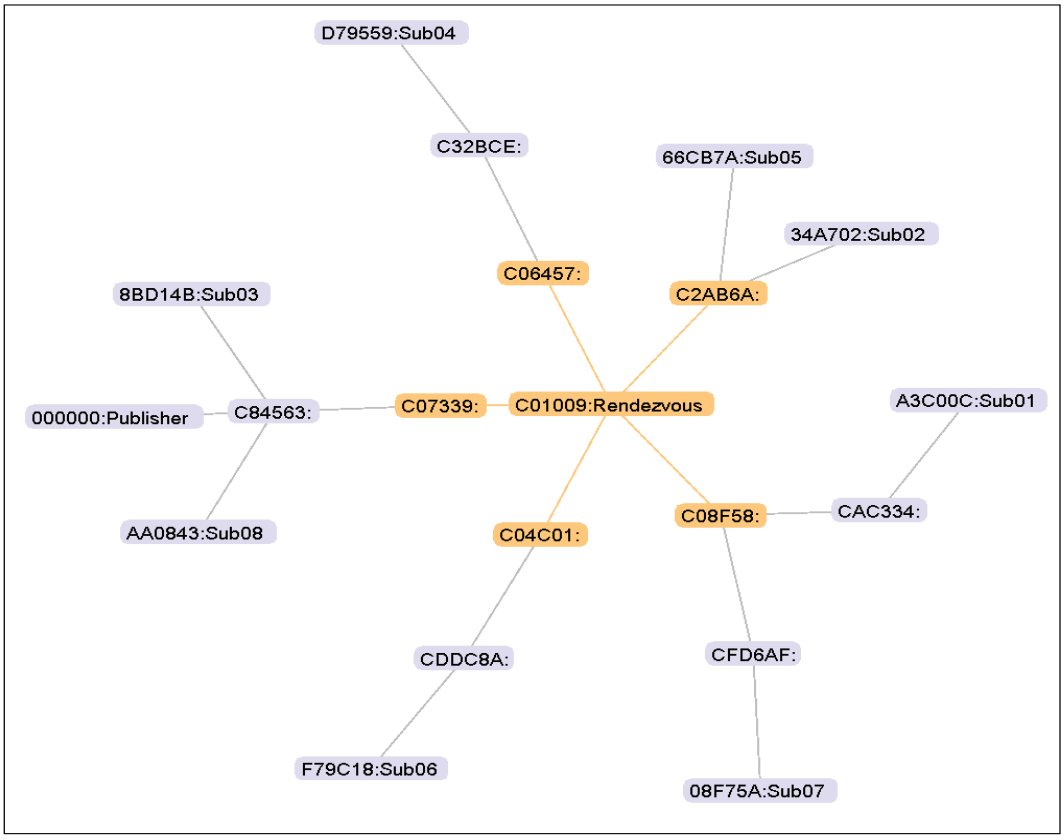
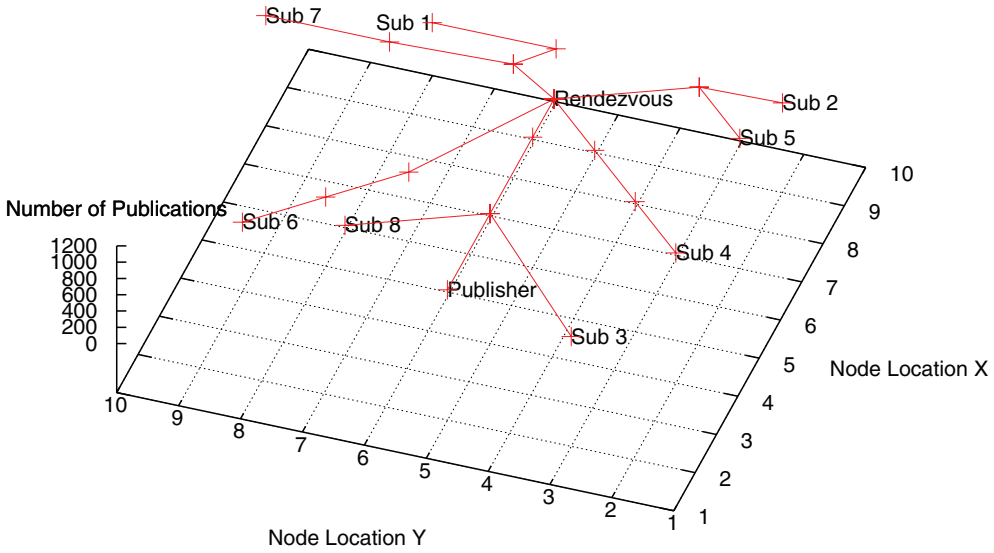
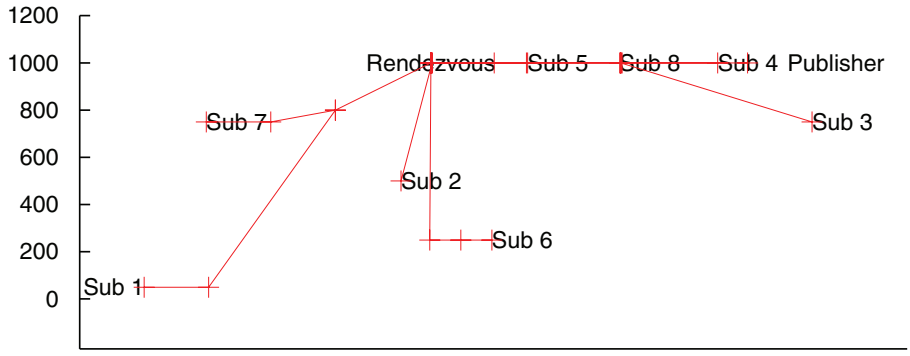


Figure 5.22: Topology in Publish/Subscribe System with Additional Dimension



a. Logical Topology



b. Delivered Publications to Subscribers

Figure 5.23: Event Traffic with 3 Dimensions Filtering

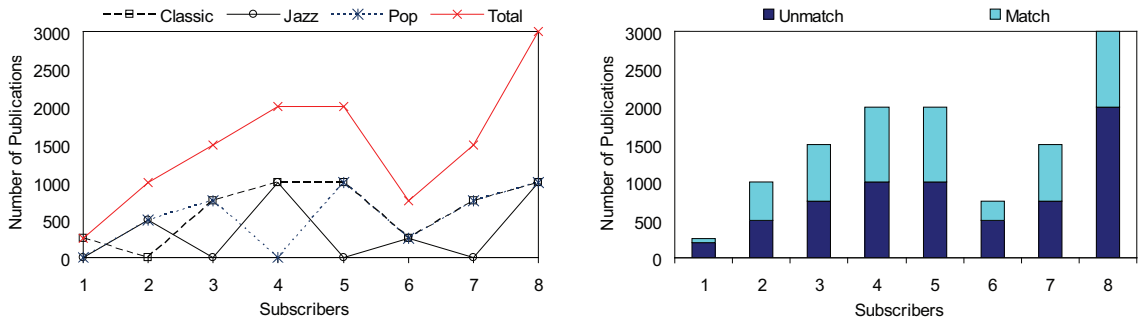


Figure 5.24: Comparison between Channels on Types vs. Additional Dimensions

5.7 Summary and Outlook

In this chapter, I demonstrate the use of *Hypercube* for event filtering. The *Hypercube* event filter introduces flexibility between the topic and content-based subscription models. In enterprise publish/subscribe systems, it is well known that a combination of topic and content-based subscription models brings the best performance to support business oriented systems. This helps application developers to determine the optimal set of indices for their specific application domains, where query trends may be specific or dependent on subscriptions.

The experiments show that adding additional dimensions in the hypercube filter transforming from *type* over-performs than constructing individual channel for *type*. Transforming the type name to the dimension can preserve similarity and hierarchy, that automatically provides neighbour matching capability. Further experiments for flexible indexing will be desirable future work.

P2P systems over the Internet need to be scalable. An indexing mechanism therefore needs to be controlled by using typed data structures. With a popular topic, a few nodes get highly overloaded, and the matching algorithm is computationally expensive. These approaches use general-purpose, structured P2P overlays, thus relying on a high-level of randomisation to achieve load balancing. Thus, constructing more efficient broker grids with optimisation of filtering is becoming a crucial issue.

An important aspect is that the values used to index *Hypercube* will have a huge impact. For example, the use of a locality sensitive hashing value from string data and the current form of *Hypercube* filter can both be exploited with the locality property. This will be worthwhile future work.

6

Context Adaptive Publish/Subscribe

A mobile ad hoc network (MANET) is a collection of self organised mobile nodes in a wireless network without an existing infrastructure. Wireless network environments are characterised by limited battery life, intermittent connection, and non continuous operation. MANETs exhibit frequent topology changes and much resource variability. Applications executing under these circumstances need to react continuously and rapidly to changes in operating conditions and must adapt their behaviour accordingly. The time-triggered approach is expensive when the expected rate of data communication is low and the applications are typically peer-to-peer rather than client-server. Thus, an event-based approach based on a publish/subscribe communication paradigm, which offers asynchronous and multi-point communication, is well suited to constructing reactive distributed computing applications. Selective event dissemination is important to support publish/subscribe systems in such network environments.

Future wireless networks will be more hybrid, and pure ad hoc networks will play relatively minor roles for deployment in the real world. The emergence of ubiquitous computing with WSNs requires construction of a global system through tiny sensors, to mobile ad hoc networks, to Internet-scale systems. P2P-based event brokering will extend seamless messaging capability over scalable heterogeneous network environments, and a data centric communication abstraction (i.e., publish/subscribe) will impact the construction of reactive distributed applications.

This chapter focuses on the abstraction of a publish/subscribe system over wireless network environments, which provides the base of mobile P2P (MP2P) communication. First, I discuss and analyse various existing publish/subscribe models for wireless network environments to show where the proposed approach fits. Then, I introduce ECCO-PPS, a structureless asynchronous group communication in Section 6.7. This highlights the architecture of ECCO-PPS followed by the experiments in Section 6.8. ECCO-PPS realises context adaptive casting using restricted flooding, which exploits a cross layer approach between the middleware and the network layer. Asynchronous group communication is integrated as publish/subscribe into MANET environments using events and subscriptions as tokens for message routing. It supports content-based publish/subscribe for selective event dissemination and enables an asymmetric publish/subscribe paradigm. Group communication channels among applications tend to be instantaneous and are terminated when contexts change, when nodes move or when resources become

unavailable. Ephemeral group management is an efficient service to enable the establishment of device communities. A situation-triggered group communication service with an impromptu group addressing scheme (i.e., dynamic channelisation) is investigated for ad hoc formation of communities of devices within ECCO-PPS.

6.1 Application Domain

Pure MANET [CM99] applications are rare in the real world. A popular example is an application for a natural disaster scenario. The network may consist of satellite-to-ground radios, mobile phones, and ad hoc modes of WiFi networks. Some devices will be backbone nodes to connect to the Internet, where they have dual networking capability. Depending on the tasks and locations, devices and carriers will form different group communications, which require multicast and more selective content-based routing, driven by the location of information and rescue materials or roles of persons.

However, the real advantage and usefulness of MANETs is brought about by the recent emergence of ubiquitous computing. The integration of smart sensors with the Internet or wireless infrastructure networks increases the coverage area. Mobile devices play important roles for collecting sensor data over ad hoc networks and conveying it to Internet backbone nodes, where no network infrastructure is supported in remote locations. Networks will be structureless and rely on ad hoc connections between nearby nodes to establish multi-hop dynamic routes to propagate data. MANETs, opportunistic networks, and pre-constructed wireless meshes have great potential for conveying time-critical sensor data. A new type of communication paradigm using an abstraction layer over hybrid wireless network environments to maximise existing wireless connectivity or data dissemination reliability will appear. The abstraction layer can be a different type of overlay such as moving groups of humans or logical mobile agents.

Consider an urban area with a high number of cars; drivers and passengers in these cars are interested in information relevant to their trips. For example, drivers would like to continuously see a map of their destination together with parking places. Such information can help to optimise their trips. Each moving object has power, and communication capability with surrounding cars via IEEE 802.11, Ultra Wide Band (UWB), or Bluetooth. These protocols provide broad band but short-range P2P communications by multi-hop transmission relayed by intermediate moving objects.

Ultimately, P2P-based communication will be deployed from sensor networks to the Internet to realise global computing. The service function provided by overlay applications will become an important issue. The services are search, dynamic routing, and storage.

6.1.1 Potential Applications

A wide range of applications for this new communication paradigm will be developed. Many information based applications such as sensor data collection (water quality, air pollution), weather forecast distribution, social and community network building, and disaster recovery coordination are just a few examples. Car-to-car communications have potentially huge application areas such as traffic data exchange and emergency notification using both infrastructure-based communications (e.g., via GPRS, UMTS, GSM, SMS and WLANs) and infrastructure-less communications (e.g., UMTS, Bluetooth, sensors). In city environments, taxi networks and temporal networks at football stadiums could be formed.

6.1.2 Selective Information Delivery

Selective information dissemination services including user interest based content delivery, subscription based notification, and targeted advertisement are important in ubiquitous applications. Recent mobile devices are equipped with Global Positioning Systems (GPS) and location-based services on mobile devices become popular. We need to avoid waves of advertisements reaching our phones but want to receive useful and timely information. More selective and effective information for the user should be delivered. High volumes of sensor data have to be aggregated within the wireless ad hoc network on the way to the Internet backbone nodes. The middleware has to filter information for potentially millions of users, given their continuously changing locations, and other contexts.

6.1.3 Mobile Peer-to-Peer

A mobile ad hoc network is often built to support specific applications. Thus, it is application-driven and interactions are P2P. For constructing an overlay, where some of the nodes connect to external networks, I aim at forming a service grid as a paradigm to support ubiquitous computing environments. Overlay nodes have high storage capabilities to support the reliability of end-to-end delivery. For example, a node may temporarily become a member of MANETs for uploading data, before leaving the group. An overlay node takes responsibility to cache the loaded data within MANETs or eventually store it on hard disk.

To support such overlay structures, routing algorithms must be adaptive and deal with a wide range of heterogeneity in network characteristics such as availability or delay. Each node acts autonomously on local node information and ensures end-to-end communication. This is a similar approach to active networks, where routing decisions are made at each node based on mobile code containing rules for routing. This is the concept of context adaptive routing in ECCO-PPS, where contexts from applications are carried within the event and are compared with local information to decide on the next action.

6.2 Wireless Networks

This section discusses characteristics of wireless networks. The current Internet provides a transport service, which supports various applications in a general manner. The fundamental function is forwarding packets, and the application knowledge stays within applications or the endpoint host nodes. This paradigm must shift to cooperate between applications and network components to optimise the resources over wireless network environments.

The following five primary categories of wireless mobile ad hoc networks are considered:

- Mobile Spontaneous Ad Hoc Networks (Pure MANETs)
- Wireless Personal Area Networks (PANs)
- Wireless Mesh Networks (WMNs)
- Delay Tolerant Networks (DTNs)
- Wireless Sensor Networks (WSNs)

Pure MANETs

Mobile ad hoc Networks (MANETs) are decentralised networks consisting of autonomous mobile nodes able to communicate with each other over wireless links. The topology of the network may change rapidly because of node mobility. It is difficult to use a conventional routing scheme based on fixed points (routers). Nodes are not homogeneous but show different signal strength,

power capacity, reliability etc. These differences require more complex coordination of nodes and distributed algorithms to maintain a dynamic network topology.

PANs

Wireless Personal Area Networks (PANs) enable communication between the devices of people or the external world. The applications can range from games to assistance of elderly people (e.g., monitoring heart rate). In PANs, heterogeneous devices with different capabilities interoperate towards a central communication point. The devices can be small enough to be implanted. Bluetooth is an industrial specification for PANs (i.e., IEEE 802.15.1).

Meshs

In *Wireless Mesh Networks* (WMNs), nodes comprise mesh routers and mesh clients forming dynamically self-organised and self-configured ad hoc networks. Mesh networks may consist of either fixed or mobile devices.

WMNs are decentralised, reliable, resilient, and cost effective. Each node transmits data only to the next node, and nodes are like repeaters, resulting in a network spanning large distances. Mesh networks are significantly robust, since each node has multiple connections to other nodes. Adding more nodes increases the capacity of networks. WMNs can communicate in difficult environments such as tunnels and high speed mobile video applications on board public transport.

DTNs

Delay Tolerant Networks (DTNs) [IET04] are significant, where continuous network connectivity does not exist. Store-and-forward mechanisms are used for routing asynchronous messages with various network transport technologies. The fundamental model is an overlay over heterogeneous transport mechanisms, and a new naming architecture based on URIs is provided.

A key difference between traditional networks and DTNs is that in traditional networks, an end-to-end path is expected to exist within a communication range, while DTNs allow more loose connections between source and destination. Network storage allows DTN nodes to buffer data bundles until connections are available.

DTN research shares a similar paradigm to asynchronous messaging. The progress of ubiquitous computing pushes DTNs to extend the communication paradigm over general heterogeneous network environments [Fal04].

WSNs

Wireless sensor networks (WSNs) consist of sensors that are able to collect various data from the physical world. The data can be exchanged with other sensors or sent to an application via a base station. Such sensor nodes are tiny devices operated by a battery equipped with a micro controller and a low-power radio for communication. Application scenarios include disaster control, traffic monitoring, and environment monitoring.

The goal of WSNs is detection and estimation of some event, similar to data processing. The goal of MANETs, on the other hand, is providing communication mechanisms. The communication pattern in WSNs is many-to-one (sensor nodes to sink), one-to-many (sink to sensor nodes), or one-hop distance location communication, while MANETs provide general communication such as *unicast* and *multicast*. In WSNs, individual nodes are unreliable and group-based robust coordination is required. Duplication and redundancy of data are common, and temporal and spatial information on each node is key for coordination.

6.2.1 Heterogeneous Hybrid Networks

The practical future of MANETs will be a hybrid form, where some nodes are connected to the Internet, or good size storage, and wireless networks are able to cover a wide area with better reliability by integrating WMNs, MANETs, DTNs and WSNs. Routing of DTNs selects the best next hop based on buffer availability. DTNs could be the base of an integrated future hybrid wireless network scheme.

6.2.2 Routing in Wireless Networks

Route optimality is not necessarily the most important feature for MANET routing. The tradeoff depends on traffic and mobility patterns.

The IETF MANET Working Group has developed a number of protocols (see [P+02], [JMH02], [YLSG02] and [GHP02]). There are a series of survey documents and comparisons on MANET protocols [AWD04]. These protocols can be classified into two groups: proactive and reactive.

Proactive protocols take a similar approach to the one used in wired networks, where a routing table on each node is continuously maintained. Thus, the route is known when the node forwards the packet. Proactive protocols cause a large amount of network traffic to maintain a constantly changing network graph due to new, moving or failing nodes. However, when the network is dynamic, the accumulated routing information may not be used at all since the routing table is only valid for limited time periods. Destination-Sequenced Distance Vector (DSDV) [P+02] and Optimised Link State Routing (OLSR) [JMQ99] are two examples.

In reactive protocols, a logical path from a source node to the destination is discovered on-demand and maintained afterwards. Examples of protocols of this family are Dynamic Source Route (DSR) [JMH02] and Ad hoc On Demand Distance Vector (AODV) [PR99]. Normally, the route is built upon the request of route construction. This on-demand mechanism does not require constant broadcasts and discovery for route maintenance but causes delays in message delivery. Using flooding may cause additional control traffic leading to a further limitation on bandwidth.

A hybrid approach such as ZRP [HPS02] is proactive within a defined number of hops. Like geographic-based routing, node location based routing (LAR, GPSR) is based on the location of nodes.

6.2.3 MANET Multicast

For wireless networks, the most natural communication type is broadcast. The dynamic topology of the network makes it difficult to maintain a multicast group. There are several multicast routing protocols for MANETs (see Section 2.3.2). The basic idea to define multicast routing in MANETs is to form a path to all group members with minimal redundancy. It is also critical whether the routing table constructed is on-demand, or optimal paths are determined once and updated periodically. Another important issue is whether control packets are flooded throughout the network or limited to the nodes in the multicast delivery tree. Most multicast protocols exploit specific characteristics of MANET environments. For example, the mesh-based approach exploits topology variation, soft-state is used by stateless protocols, location-aided multicast exploits the knowledge of location, and the gossip-based approach is based on randomness in communication and mobility.

A good performance study of various multicast routing protocols can be found in [L+00] [WC02]. The reported results show that mesh protocols perform significantly better than tree protocols

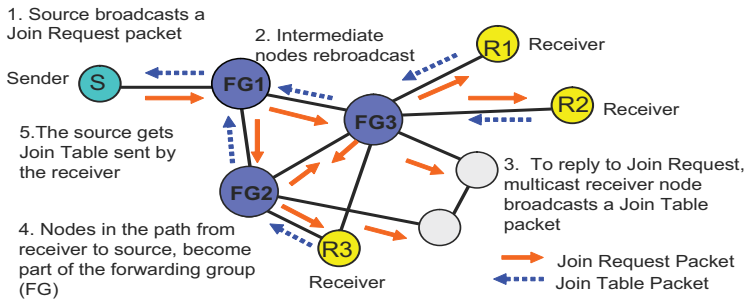


Figure 6.1: ODMRP

in mobile scenarios, especially for supporting multiple small groups. Publish/subscribe fundamentally needs to address what multicast does, because precise layering of networks is not appropriate. I discuss the architecture and routing strategies in more detail in section 6.3.1.

ODMRP

ODMRP is an incorporation of the Forwarding Group Multicast Protocol (FGMP) and an on-demand scheme, which is characterised by simplicity and exploitation of the broadcast nature of wireless environments. ODMRP applies an on-demand routing technique to avoid channel overhead and improve scalability (Fig. 6.1). It will attempt to create a group of forwarding nodes between the source and the multicast receivers. The concept of the forwarding group is a set of nodes responsible for forwarding multicast data on shortest paths between any member pairs, to build a forwarding mesh for each multicast group. These forwarding nodes re-broadcast any packet they receive to reach all interested multicast receivers. The multicast mesh is created through a reply-response phase that is repeated at intervals to keep the routes to the multicast receivers fresh. With the concept of a forwarding group, only a subset of nodes forwards multicast packets (scoped flooding). ODMRP provides richer connectivity among multicast members using a mesh-based approach. It supplies multiple routes for one particular destination, which helps in case of topology changes and node failure. ODMRP takes a soft-state approach to maintain multicast group members. Nodes do not need to send any explicit control message to leave the group. This can work with any unicast protocol.

ODMRP improves its performance using mobility and location information. When GPS is available, the network provides location and mobility information to each node. Routing protocols can use this information for improving performance and robustness. ODMRP adapts route expiration time, using location and mobility information to estimate node movement.

Geographical Casting

Geocast is a routing protocol using spatial context to decide on the receivers. The protocol is flooding-based, routing-based, or cluster-based. Flooding-based protocols use broadcast for dissemination of packets to the geocast region. Protocols in this category include *Location-Based Multicast* (LBM) [KV99]. Routing-based protocols construct routes from the source to the geocast region using control packets. Examples include the *Mesh-based Geocast Routing Protocol* (MGRP) [BCT01], *Geocast Adaptive Mesh Environment for Routing* (GAMER) [CL03a], and *GeoTORA* [KV03]. Cluster-based protocols geographically partition a wireless network into several disjoint and equally sized regions, and select a cluster head in each region to operate message exchange. Protocols with clustering techniques include *GeoGRID* [LTLS00], *Obstacle-Free Single-Destination Geocasting Protocol* (OFSGP), and *Obstacle-Free Multi-Destination Geocasting Protocol* (OFMGP) [CCT03].

In geocast, the membership of the multicast group is implicitly defined as the set of nodes within a certain area. This approach is similar to the symmetric publish/subscribe paradigm with spatial information as context.

6.2.4 DTN Multicast

DTN messages contain Endpoint IDs (EIDs) to address an end point, where EIDs refer to one node (*unicast*), one of a group of nodes (*anycast*), or all of a group of nodes (*multicast*). *Multicast* [ZAZ05] can address a group of EIDs or a single EID, that represents a group of nodes. The membership is defined to be *in* at the time of delivery. *Anycast* delivers a message to a current member of an *anycast* group, and the delivery time is less than the node's membership lifetime.

6.3 Publish/Subscribe in MANETs

Most work in the publish/subscribe area has been done for wired and fixed networks. In recent years, however, publish/subscribe systems for MANETs have been reported. Publish/subscribe becomes powerful in MANET environments, when not all the nodes are in an exclusively ad hoc topology. Some nodes may be connected to the Internet backbone or may be relay nodes for different network environments. For example, a publisher broker node can act as a gateway from a sensor network, performing data aggregation and distributing filtered messages to other mobile networks based on content. Thus, to achieve improved asynchronous communication, a semantics of publish/subscribe must be introduced. Nodes that are mobile have a limited transmission range; there will be several intermediate nodes between a source to destination path, leading to a dramatic increase in complexity. Context-awareness is important for improving the accuracy of data dissemination and performance in such ubiquitous environments.

Characteristics of Publish/Subscribe in MANETs

Extensive research work has been done in protocol design and performance analysis for multicast over MANETs. However, to provide more selective message dissemination and to support dynamic group membership, a content-based publish/subscribe paradigm is desirable.

The publish/subscribe model (see Section 4.5) in MANETs is fundamentally similar to wired network environments. However, to deal with dynamic network environments, the approach appears to be more broadcast based. The main difference is that the groups are created dynamically based on the content of the event instead of a pre-assigned channel. As a result, the groups tend to be smaller, frequently short-lived, and more numerous. This is significantly different from group membership in traditional multicast, where groups are defined in advance and only the membership is dynamic. Thus, a function to create and remove multicast groups based on data content is particularly important for supporting resource constrained ad hoc networks so that selective data delivery can be realised.

Mobile nodes are sparsely distributed so that networks are often partitioned due to geographical separation or node movement. Therefore, nodes may often become disconnected, but most of the work thus far does not consider network partitions. There will be two distinct directions for network partitioning situations: use of epidemic dissemination to increase the probability of delivery, and the construction of store-and-forward type overlay functions for either dealing with disconnected operation or aiming at opportunistic event dissemination (e.g., DTNs).

6.3.1 Architectural model

[BV05] provides a good summary of distributed event systems in wide-area networks. Fig. 6.2

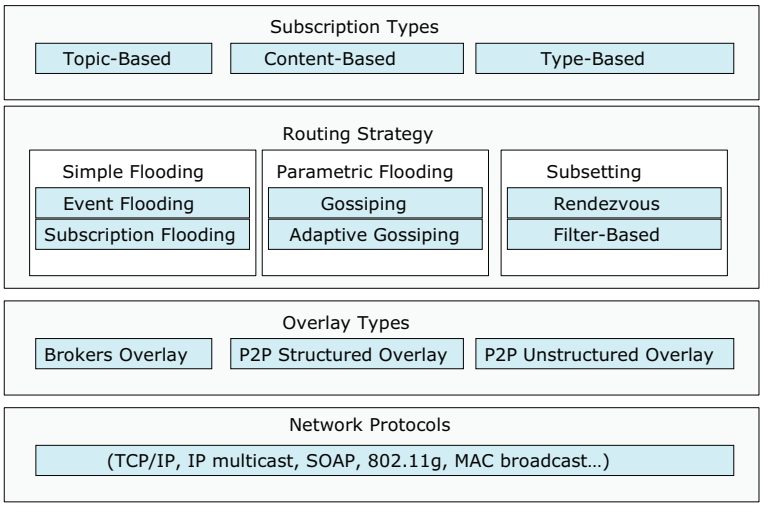


Figure 6.2: Publish/Subscribe System Architecture

depicts the further refined architectural overview of publish/subscribe systems including mobile wireless network environments. In this section, I analyse various event dissemination schemes in mobile ad hoc network environments. Categories are:

- Structured overlay (e.g., DHT, GHT) on top of the routing layer
- Broadcast (flood) and Gossip based approach
- Cross-layer between the routing layer and the publish/subscribe layer
- Mobile Agent

6.3.1.1 Overlay

Application Level Multicast (ALM) over wired P2P networks can obtain automatic network repair when node failure or disconnection occurs. However, mobility in MANETs may not tolerate the ALM approach. Also a mobile network can consist of many different nodes, each with its own characteristics (e.g., battery power, storage): this is more heterogeneous compared to wired networks, where P2P is more homogeneous and static. Existing multicast and publish/subscribe systems using overlays in MANETs are described below.

Overlay Tree/Mesh

In [CMPC04], an *Overlay Tree* creates a dissemination tree and maintains it in response to changes in the topology by reconfiguring routes traversed by events (Fig. 6.3). Thus, the assumption is that the underlying tree is kept connected and loop-free. If events get lost during reconfiguration, gossip-based operation is used to obtain missing events. It assumes that the network never has a partition for long enough to maintain the tree. [CP05] provides an improved approach, by subscription diffusion, for better reaction to topology changes, which is similar to the Zone Routing Protocol (ZRP) [HPS02].

In [HGM01], a distributed protocol to construct an optimised publish/subscribe tree in ad hoc wireless networks is presented. Each publisher node becomes a root in a multicast tree. Their algorithm builds multicast trees directly over broadcast primitives, depending on the underlying network infrastructure to provide basic network connectivity.

In [GM03], an overlay multicast protocol builds a virtual mesh spanning all member nodes of a multicast group. It employs standard unicast routing. The advantage of this approach is

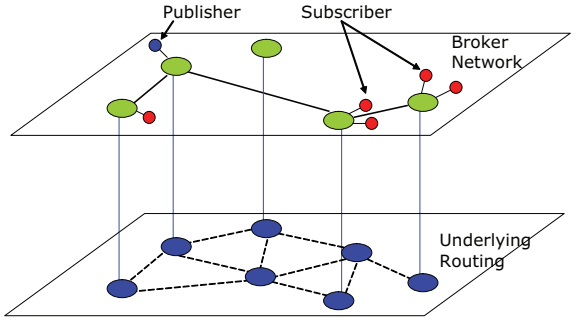


Figure 6.3: Publish/Subscribe with an Overlay Network

robustness and low overhead.

DHT based Structured Overlay
In [BMVV05], a structured P2P overlay network is used for a publish/subscribe system. Subscriptions are mapped to keys and set to a rendezvous node. There is some optimisation such as bundled notification dissemination. The performance of this approach depends on the real mapping between the overlay network and the underlying network topology (Fig. 6.4).

CrossROAD [Del05] is a cross-layer optimised P2P substrate for MANETs, which provides the same function as Pastry in wired networks through the P2P common API [DZD+03]. CrossROAD reduces the overlay management traffic by exploiting cross-layer interactions with a proactive routing protocol (e.g., OLSR). It is essentially Scribe in MANETs. Experiments described in [Del05] compare CrossROAD to Scribe with reactive MANET protocols for routing. It shows that CrossROAD over-performs.

Overlay with Multicast
Applying flooding (or random walk) over the physical topology graph is one way to find routes to an object with a target key. DSR or AODV can be used for this approach. Creating replications within n-hop nodes, and finding routes by DSR or AODV is another way.

Another possibility is to create topology dependent identifiers for the nodes and to apply geographical routing techniques (e.g., GPSR [KK00], BVR [FRZ+05], or GEM [NS03]). An object is stored and replicated at nodes near to the node where the key is stored. This approach was proposed in the geographical hash table (GHT) [RKY+02]. GHT is built on top of GPSR, and keys are coordinates in the geographical space.

Landmark routing (e.g., LANMAR [PGH00]) uses a topology-independent structure, where nodes have identifiers independent from the topology and objects are stored at the node closest to their

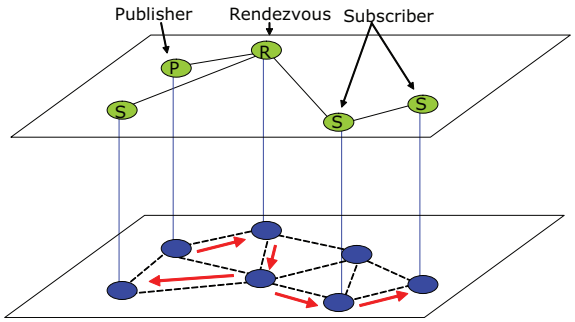


Figure 6.4: Publish/Subscribe with a DHT over P2P

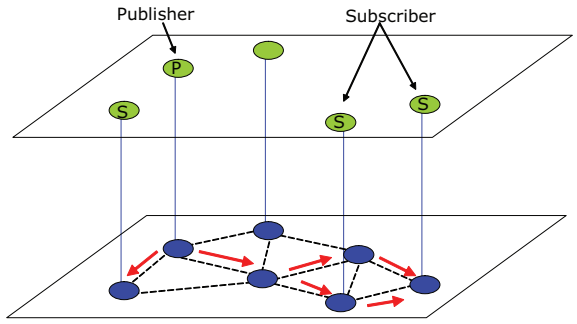


Figure 6.5: Publish/Subscribe with Broadcast

keys.

Observation
Structured overlays assign identifiers to nodes and control the identifiers of neighbours in overlay networks and the keys of the objects that they store. This is effective since lookups can be done with cost $O(logN)$; this is better than a flooding approach. However, the characteristics of MANETs require a significant amount of traffic to maintain the overlay links. Thus, strict layering may not work. Sharing information across layers may impose layering (e.g., CrossROAD).

When a topology dependent structure is used under high mobility conditions, the stored objects may need to move according to the new coordination. This may cause a significant impact with a large amount of object movement. For GHT solutions, the key location can be far away from the data source or scattered across the network, which can be a problem.

6.3.1.2 Broadcast and Gossip based approaches

Maintaining a tree topology in MANETs is challenging, as it requires high network traffic to detect and repair failed links. Thus, a structureless approach is desirable, where no global network-wide structure and no link breakage detection are required. This approach is resilient to network partition. The epidemic dissemination mechanism is a powerful form of P2P cooperation. Broadcasting delivers messages to all nodes in a network (Fig. 6.5). Unicast and multicast routing protocols often use broadcasting for the route discovery phase.

In [GT02], a theoretical foundation for epidemic approaches is described with a formal proof and simulation results. It shows that the throughput within a session increases, along with increased mobility. [WC02] gives an overview of broadcast protocols and shows simulation results with various mobility scenarios.

Variations of broadcast and gossip based approaches for delivering messages are presented below.

Simple Flooding
In simple flooding, every node re-broadcasts a received packet for the first time (i.e., using MAC-layer broadcasting) [CHTV99] to all of its neighbours. Re-broadcasts can be randomly jittered to reduce the chance of packet collisions. When the data is delivered to all nodes, the dissemination terminates, as each node keeps the history of data receipts. Flooding is robust and scalable because of the high redundancy of messages. A drawback is the high network traffic.
Controlled Flooding
To optimise simple flooding, many optimisation algorithms have been introduced.

[LGC99] uses knowledge about the network neighbours. This protocol assumes both unique identification for the nodes and knowledge of these identifications of the neighbours in the network.

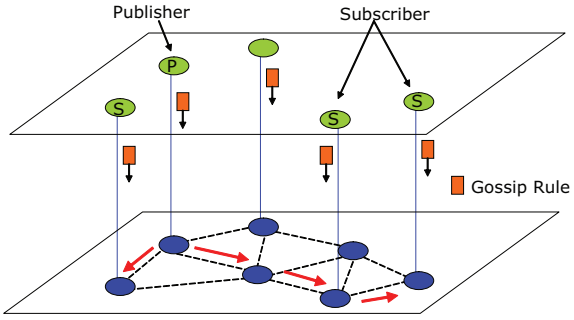


Figure 6.6: Publish/Subscribe with a Gossip Protocol

When a node receives data, it compares the neighbours of the sending node to its own neighbours. The receiving node stops resending, if the receiving node neighbours are a subset of the sending node neighbours. Otherwise, it resends the data to its neighbours.

The algorithm in [LW02] uses 2-hop neighbour knowledge that is exchanged by periodic messages. Each message contains the node's identifier and its list of known neighbours. After a node receives a message from all its neighbours, it has topology information within two hops. When node Y receives a broadcast from node X, Y knows all neighbours of X. If Y has neighbours not covered by X, it broadcasts the packet with a random delay.

[HGM01] maintains the event dissemination tree and prunes the forwarding tree. Specifically, when an event arrives at each broker in the forwarding tree, it is forwarded onto one of the broker's outgoing branches only if the event matches a subscription at a broker leading from this branch. In other words, the event broker selectively forwards an event based on the result of partial matching.

In [BBC+05] [BMVV05], content-based proximity-driven routing is provided, where a structureless network is considered without link breakdown detection. Subscriptions are kept in the local node. It uses broadcast to efficiently send a message to all neighbouring nodes based on an estimation of their distance from a potential subscriber of the message.

CBM (Content Based Multicast) [ZS00] explicitly addresses content-based publish/subscribe in wireless networks. It exploits position-based routing to constrain the propagation of matching events along a specified direction.

Probabilistic Flooding (Gossip)
Gossiping is a simple routing protocol, where the retransmission probability function is a constant value (Fig. 6.6). Gossiping exhibits a type of bimodal behaviour, where either few or most nodes receive the message. When a node receives a message, it broadcasts the message to its neighbours with probability p and discards it with probability $1 - p$. Simple flooding is a special case of gossiping with retransmission probability set to 1.

In [HHL02], this algorithm is extended, where probability 1 is given for the first k hops. This stops gossiping when only a few neighbours are near the gossip root node.

In [BCG04], topics are coordinated in a hierarchical format, and decentralised publish/subscribe without mediators is constructed without any assumption on the underlying routing protocol. Each published event has a life period, and when it expires the event is not propagated. Heartbeat messages containing a list of the event's subscriptions are sent to the neighbour nodes.

Adaptive Probabilistic Flooding
Like probabilistic flooding, the dissemination of adaptive probabilistic flooding is based on a

probability value. Unlike probabilistic flooding, the value is not fixed but adapts to the local network condition such as node density.

Autonomous Gossip [DQA04] proposes a completely stateless, bio inspired, self organising mechanism to disseminate information in a content-based fashion according to node *similarities*. Thus, this approach aims at selective casting in a self organising manner.

Parametric Probabilistic Routing (PPR) protocols apply a limited-flooding strategy [CEK03]. The key element is that the retransmission probability at each node is a function of various parameters rather than a constant, used in previous gossiping approaches such as [DGH+87]. The retransmission probability function is defined in terms of parameters, indicating when a node forwards a received message to its neighbours. The retransmission function depends on different factors such as the hop-distance of the source to the destination, the number of travelled hops, the number of times a node has forwarded the same packet, the number of neighbours, etc. PPR protocols thus perform controlled and directed flooding resulting in more than one packet copy, while the traditional single path method uses a single copy of packet.

In [BEK+05], the retransmission probability at each node is calculated, based on the distance from the source node to the destination node in hop counts. The probability depends on the number of travelled hops, where larger travelled hop counts indicate that the destination is closer.

The adaptive probabilistic flooding approach can guarantee various service levels of message delivery. Multi-path methods outperform single-path methods in the presence of redundancy. Making the retransmission probability dependent on network information will promise more robust routing protocols, especially in noisy networks such as WSNs.

In Context Aware Routing (CAR) [MHM05], a host utility is used to calculate multi-criteria decisions for forwarding data, where each context has associated attributes (e.g., host collocation, host mobility, battery level). Thus, an approach is based on hosts acting as carrier for asynchronous delivery of messages to hosts, which can be the final subscribers. Time series analysis on state space models ($Kalmanfilter$) is used to keep a history and to predict the evolution of network scenarios.

6.3.1.3 Cross-layer Integrated Approach

For performance improvement there have been efforts using cross layer design to enable breaking up traditional system partitioning by exchanging information between communication layers. Thus far cross layering focuses on between the data link layer and the network layer. It involves complex issues [KK05] and the exchange of information provides global optimisation.

In [YB04a], [YB04b], we have presented asynchronous group communication using events and subscriptions as tokens for message routing and exploit cross-layer approach between the routing layer and the publish/subscribe layer. Thus, this is a novel cross layer design between the middleware and network layers. See Section 6.7 for the details.

In [MCP05], the publish/subscribe tree overlay network is maintained by using MAODV like routing. It maintains a tree overlay on top of the dynamic topology of a MANET, where the underlying tree is kept connected and loop-free. Thus, the publish/subscribe layer integrates with the network connectivity layer. MADOV is adapted for maintaining an acyclic overlay network joining the brokers of a publish/subscribe middleware. The protocols based on a structured overlay approach also combine the cross layer integrated approach (e.g., CrossROAD).

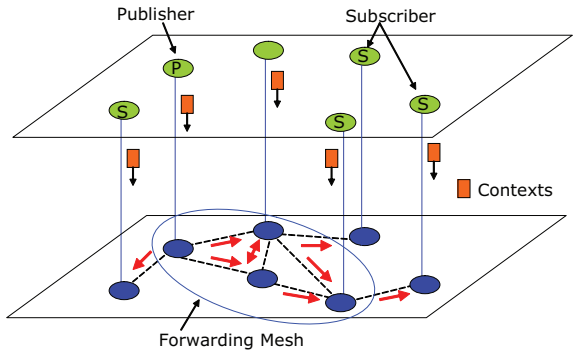


Figure 6.7: ECCO Pervasive Publish/Subscribe System

6.3.1.4 Mobile Agent Approach

In the message ferrying scheme [ZA03], a set of special nodes called message ferries act as moving access points on predefined ferry routes. Ferries collect and deliver messages during their contacts with other nodes when they move into direct communication range of each other. The message ferry exploits different epidemic routings depending on the group sizes.

Unlike traditional approaches, where node mobility creates difficulties, a new approach takes advantage of mobility to disseminate data. Smart-Tag [BLB02] and RuralWiFi [Asi04] have exploited similar ideas, where buses are equipped with Bluetooth/WiFi to *ferry* the message between fixed points (e.g., remote locations with no wired infrastructure and urban areas with *wireless mesh*).

6.3.2 Discussion

State maintenance requires control traffic, which could be expensive to operate, while a stateless approach could also be expensive if using event flooding. Stateful approaches suffer from frequent topology changes, and stateless approaches are more suitable for topology change and partitioning and isolation of nodes. Thus, dealing with mobility and partitioning of networks shows that the basis of event dissemination mechanisms should be epidemic. The basic gossip dissemination sends each message to a randomly chosen group of nodes. This approach operates in a decentralised fashion and is robust against node and network link failures.

[BEK+05] shows an interesting comparison among the three routing protocols (i.e., Gossiping, Shortest Path, MAODV). MAODV is an on-demand based multicast routing protocol. The ranking of the protocols differs based on mobility scenarios. Shortest-Path wins with regard to overall performance for low mobility in a single destination scenario. In no or low mobility conditions, MAODV wins in any-to-any delivery. Gossiping wins in any-to-any networks for high mobility values [CEK03].

ECCO-PPS deploys a controlled flooding based dissemination approach to limit the propagation of events. Similar to [HGM01], ECCO-PPS delivers events by creating multicast groups (Fig. 6.7). The difference is that ECCO-PPS uses a mesh-based approach instead of the tree-based one used in [HGM01].

ECCO-PPS takes another characteristic approach for adaptivity of contexts. While many routing protocols are studied in MANETs, more attention should be paid to the design of context-adaptive, network layer protocols. This approach simplifies forwarding algorithms based on local information, where events are labelled with publish/subscribe information, and events migrate through the nodes in the network according to their similarity with the nodes' subscription and

destination. Depending on the similarity at the broker node, messages decide to either consume, discard, or forward. Recently, geographical routing has been extended to provide multicast services (e.g., [MFWL03] and [TFW04]). Geographical routing can be seen as an instance of ECCO-PPS.

In conclusion, ECCO-PPS combines the advantage of both *Adaptive Probabilistic Flooding* and the *Cross-layer Integrated Approach* and exploits context aware routing.

6.4 Mobility

This section briefly discusses mobility-related issues in publish/subscribe systems.

On the Internet, intermittent connectivity means data can be lost. Mobile IP [JPA03] allows a device to maintain its original IP address or a unique identifier, while visiting other subnetworks. The drawback of Mobile IP is that it is based on a heavy mechanism, of triangular routing among home IP, old IP and new IP. Nevertheless within the ad hoc mode of networks, as far as identifiers are unique among them, it is not necessary to adapt to the IP scheme, where non-IP communication may be deployed. The wired network de-facto standard JMS provides durable subscription and persistent delivery mode, where store-and-forward mechanisms ensure message delivery for disconnected clients.

Mobility is probably the most significant attribute of MANETs. In MANET environments, the existence of fixed broker networks cannot be expected to manage mobility. Two distinct approaches have been attempted: [LJC+00] defines hierarchically distributed brokers, and [LH00] defines a virtual backbone as a subset of nodes. Both approaches are responsible for replicating and maintaining the location information of mobile nodes.

6.4.1 Disconnected Operation and Device Mobility

Overall, mobility indicates two situations: intentional disconnection from publish/subscribe systems and physical location change. One important aspect of physical location change has two meanings. The user may change his/her physical location to use another device to continue the work, or the device moves around continuing the operation. JMS defines a *client id* for supporting both situations. However, no publish/subscribe in MANETs addresses this issue which must be solved on a global computing scale. Note that a disconnected condition can happen without intention, due to temporary signal failure, where reconnection must be handled by a proper lower layer mechanism.

Previous research on data delivery in sparse MANETs has focused on exploiting the mobility of nodes by buffering packets during network partitions. When the network is reconnected, data packets are forwarded. There are two approaches: reactive schemes ([DFL01]), where the existing mobility of the nodes is exploited to buffer and deliver messages across network partitions, while proactive schemes ([ZAZ04], [GLM+04]) require nodes to move as dictated.

6.4.2 Mobility Support for Publish/Subscribe in Wired Networks

In wired P2P networks, several existing publish/subscribe systems support disconnected operation (see [PMR01] [HGM01]). In JXTA, the peer receives a peer-id, which is independent of the IP address, so that the peer-id can change its location. Elvin was extended to support disconnectedness using a central caching proxy [SAS01]. JEDI [CN01] [CNF98b] uses explicit *moveIn* and *moveOut* operations to relocate clients. This includes reconfiguration of the dispatching

tree, subscription information, and minimising event loss during reconfiguration. The mobility extension of SIENA [CN01] [CCW03] is similar to JEDI. Explicit sign-offs are required and stored notifications are directly requested from the old location. REBECA [ZF03] supports locations transparent publish/subscribe based on a relocation algorithm. Mobility support requires messages during the movement to be recovered disconnected period.

6.4.3 Advantages of Mobility

In Landmark Ad Hoc Routing (LANMAR) [PGH00], group mobility is used to reduce the routing update overhead. Link states are exchanged only among neighbouring nodes. Constructing hierarchical ad hoc networks, consisting of multiple layers of mobile ad hoc networks, where different strengths of nodes are used in each layer, achieves scalability by providing better connectivity. Last Encounter Routing (LER) [GV03] uses node movement and gossiping to achieve free dissemination of data. Message Ferry (MF) [PGC00] aggressively uses a set of mobile nodes to provide communication services.

6.4.4 Mobility Model and Simulation

The most common models are the Random Walk Mobility Model [CBD02] and the Random Waypoint Mobility Model [JMH02] [Bet01]. Both of them simulate node movement in a rectangular area. The Random Waypoint Mobility Model is refined in the Random Direction Mobility Model [RMSM01], which uses a random direction instead of a random waypoint. These two models are extended to create continuous movement in the Boundless Simulation Area Model [Haa97]. When nodes reach the border, they appear at the opposite side.

In the City Section Mobility Model [CBD02], nodes move on streets choosing destinations at random and follow the shortest paths to them. [JBRA+03] presents the Obstacle Mobility Model, which simulates real world topographies with objects and paths. It can design model-specific scenarios and simulate radio signals according to the topographies. The Graph-Based Mobility Model [THB+02] also maps the topology of a scenario with the node movement, by a graph.

In WSNs, most of the current protocols assume that the sensor nodes and the sink are stationary. However, there will be more situations where the sink, and possibly the sensors, need to be mobile. Sensor nodes may be attached to or carried by mobile entities.

6.4.5 Discussion

Mobile IPv6 [JPA03] is designed to support host mobility and is being standardised. The network mobility protocol, known as NEMO [IET05], conceals movements for a network that moves in its entirety. Several routing protocols, such as DSR [JMH02], AODV [P+02], and OLSR [CJ03] have mobility support by dynamically adapting topology changes. On-demand based routing protocols cope better with mobility than proactive approaches, where fresh paths are built only when needed, avoiding maintenance of route information for unused destinations, which can rapidly become stale.

Ubiquitous computing brought further complexity to mobility aspects, where network structures are more heterogeneous vertically and horizontally. Location-based computing is an important aspect in ubiquitous computing. Mobility may apply to all nodes within a network or only to a subset of nodes. The degree of mobility may also vary from occasional movement, with long periods of immobility in between, to constant travel. Mobility has a large impact on networking protocols and distributed algorithms, including the speed of movement. There is no sufficient simulation environment currently available for this level.

ECCO-PPS takes advantage of a mesh topology to deal with mobility. The change of membership is maintained by a soft state mechanism instead of any explicit mechanism for disconnected operation. Publish/subscribe needs to consider data trends and social aspects. Another aspect of mobility, based on intentional mobility related to membership, has to be considered in publish/subscribe, not only the MANETs but also for global computing.

6.5 Reliability

This section summaries reliability issues for messaging in general and for data delivery.

Publish/subscribe systems should be tolerant of their own failure and of network problems. Reliable message delivery in messaging differs from that provided in TCP. Publish/subscribe systems provide asynchronous communication, and message passing from a publisher to a broker does not guarantee that the message is delivered to the target subscribers (see our paper [Yon03] for reliability in messaging). In a centralised model, the server persists the messages in a central database. Since the server manages communication between all clients in common, this implements guaranteed message delivery. A central database is not available in a MANET or in P2P environments, as the constant availability of any node cannot be relied on. This increases the complexity in achieving guaranteed message delivery. Given network dynamics, more efficient and flexible alternatives for reliable communication must be provided.

6.5.1 Reliability in Publish/Subscribe Systems

In wired P2P environments, reliability in Scribe includes two aspects: using TCP for message delivery and flow control, and using Pastry to repair the tree when nodes fails. Here, forwarder failure is maintained by heart-beat messages between parent and child nodes, and the root state is replicated across the k closest nodes to the root node for the root failure. The reliability of transport may be taken care of by these approaches, but not the messaging level, where the messages are delayed by the store-and-forward mechanism.

One approach to provide reliability is by probabilistic, semantically reliable multicast [EG01]. Hierarchical Probabilistic Multicast (hpmcast) [EG01] is a gossip-based algorithm which deals with the more complex case of multicasting an event to a subset of the system only. Until this work, most research considered gossip-based algorithms based on broadcasting. Content-based publish/subscribe systems brought additional requirements such as dynamic dissemination. In general, traditional IP multicast lacks reliability guarantees, and *reliable protocols* do not scale well. For example, Reliable Multicast Transport Protocol (RMTP) [TS97] generates a flood of positive acknowledgements from receivers. Moreover, such protocols hide any form of membership [ADKM92] [MSMA91], making them difficult to exploit with more dynamic dissemination. Decentralisation is a key concept underlying the scalability properties of gossip-based broadcast algorithms. For example, the overall load of retransmissions is reduced by decentralising the effort. Participants have symmetric roles, and they are equally eligible to forward information. Thus, gossip-based algorithms fit well with systems with an underlying P2P model.

6.5.2 Reliable Data Delivery by Redundant Paths

The use of redundant routing paths and duplication of data are efficient approaches as described below:

Multiple trees

Overlay networks attempt to use redundant paths to achieve better performance and reliability. Resilient Overlay Network (RON) [BGS00] allows applications to select an overlay path, detect path failures, and recover data through other overlay paths in application-specific ways. [LKGH03] proposes a system for performing multi-point transfers across richly connected overlays by coordinating delivery of subsets of data in a distributed fashion. To obtain good performance, many systems use overlay paths corresponding to disjoint sets of physical links.

Mesh

Mesh can be built for forwarding multicast data and can address robustness and reliability requirements with path redundancy inherent to meshes. CAMP [GLAM99], ODMRP [LGC99] [LSG99], and FGMP [CGZ98] are multicast protocols in this category.

Redundant and Split Data

Splitstream [MFH+03] creates multiple multicast trees. At the same time, multicast streams are split into multiple stripes. This helps to balance the forwarding load among the participating nodes. Erasure codes have a reputation for achieving efficient distribution of bulk data in overlay networks [JMH02] and P2P networks [SRJB03]. This may be a good solution for coping with unreliable transmissions in wireless sensor networks, and for achieving reliability in large-scale distributed storage systems [RT99], but it is not clear whether erasure codes can perform better than simpler replications.

6.5.3 Reliable Routing Protocol

Most protocols maintain the global state of the network using either a soft or a hard state. The soft state operates periodic control packet flooding, and the hard state controls every link failure for consistency. Soft state causes a higher control overhead and achieves higher reliability, while hard state is more efficient with lower overhead and lower reliability.

Broadcast based Clustering

Clustering protocols create a forwarding tree among cluster head nodes. The cluster structure is maintained proactively. Thus, it is expensive, and the efficiency depends on how up-to-date the forwarding tree is under dynamic network conditions.

Reliable Broadcast

A broadcast medium window (BMW) mechanism in the MAC protocol of MANET ensures reliable transmission to neighbours using round robin transmission. This approach is based on unicasting without taking advantage of the wireless signal's broadcast nature.

Reliable Multicast

Deterministic protocols and probabilistic protocols are two approaches. Probabilistic protocols guarantee delivery with a certain probability. Although not as safe as guaranteed protocols, probabilistic protocols have less restricted constraints and overhead, making them more suitable for dynamic network environments.

Examples of deterministic protocols include Reliable Broadcast (RB) [PR97], Adaptive Reliable Multicast (ARMP) [GS99], and Reliable Multicast Algorithm (RMA).

Examples of probabilistic protocols are Bimodal Multicast [BHO+98], Anonymous Gossip (AG) [CRB01], and Route Driven Gossip (RDG) [LEH02] [EGH+01]. Bimodal multicast has two phases:

in the first phase, regular multicast is used to disseminate the multicast message, and in the second phase, each member randomly selects a subset of members to gossip the received messages. A member noticing any message loss will ask for retransmission according to the gossip message it receives. Membership is maintained independently for each phase. AG is an extension of Bimodal multicast for ad hoc networks. RDG has only one phase: each member builds a partial view on the group, depending on its routing information. The view is updated with information obtained from other members. A new received message is gossiped several times by a member to the other members, selected at random within the view, along with the view of the member. This approach retains the feature of predictable behaviour. It is independent of any underlying multicast support (see [Z+03]).

6.5.4 Discussion

I have discussed reliability issues from the point of view of event delivery. More precise semantics of event delivery modes such as *once and only once delivery* must be defined for wireless ad hoc network environments in a standard manner (e.g., JMS). Furthermore, involvement of WSNs at the edge of networks will raise more complex issues, where reliability tends to address more data quality perspectives.

The role of event broker nodes in enhancing reliability can be important [ABKM02]. Broker nodes can provide storage accessible by network components and applications, which will be resilient to the environment. Where disconnection or disruption is common, the ability to buffer data within the network becomes one of the few critical functions to provide high reliability.

6.6 Membership

This section discusses group membership in messaging and group communication. Defining and maintaining group membership is a fundamental function of distributed systems. Network addressing and routing schemes contain part of membership functionality, which can be categorised into the following three types:

- *Unicast* is a one-to-one relationship between the network address and endpoint. A single receiver endpoint is identified by each destination address.
- *Broadcast* and *Multicast* have a one-to-many relationship between the network addresses and network endpoints. A set of receiver endpoints is identified by the destination address, and all information is replicated. *Multicast* can model this with a specific topic, and broadcast is the case where all the nodes in the network are the subscribers.
- *Anycast* is another one-to-many relationship between the network addresses and network endpoints. The difference from *Broadcast and Multicast* is that only one of the endpoints is selected at any given time to receive messages. It is a network routing and addressing scheme, where data is routed to the *nearest* or *best* destination as viewed by the routing topology.

Publish/subscribe can conceptually contain all of the above schemes, and provides the filtering function (i.e., content-based subscription). For example, *unicast* can be modelled as a topic-based publish/subscribe with restricted access by only two clients. *Anycast* has exactly the same semantics as the *Queue* paradigm, that JMS defines.

In connected wired or many wireless networks, the transmission latency between sources and potential group members is typically negligible for most applications. As a result, if a node is a group member at the instant of message generation, the node is highly likely to be a group

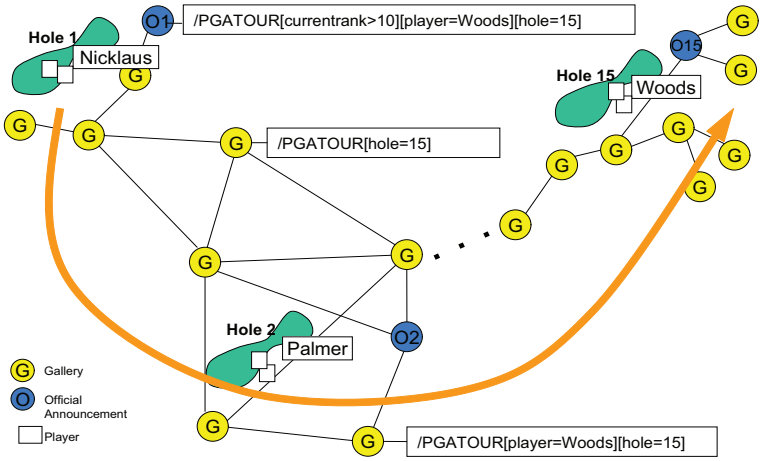


Figure 6.8: Golf Forum

member at the time when the message arrives. In sparse MANETs, however, the transmission latency may be of considerably longer duration. As a result, a node that is a member at the time a message is sent may not be a member when the message is delivered to this node. Defining membership involves the following notion of time:

- Time of message generation
- Time of subscription (joining the group and leaving the group)
- Time of message arrival to a subscriber

In *Multicast*, this problem is addressed in [JFP04], and the membership is generally determined by a specified time window. However, the emergence of WSNs brings further problems. For example, data produced from WSNs need to be delivered to subscribers over the Internet. Membership can be determined, whether based on the data production time, or data reception time at the Internet edge nodes. Time synchronisation over different network environments and the meaning of *timestamp* has to be clearly defined (see Section 7.7 for time synchronisation issues).

In ECCO-PPS, the publisher edge broker (the originator of messages) deploys controlled flooding to reach the destinations. The intermediate brokers use their knowledge of local environments to decide whether the message matches the intended destination (i.e., membership) of the message and membership is controlled in a symmetric form. ECCO-PPS currently uses simple semantics for membership time: a message is delivered if the node belongs to the group when the message is delivered.

6.7 ECCO Pervasive Publish/Subscribe

In this section, a system context adaptive Pervasive Publish/Subscribe (ECCO-PPS) is described. ECCO-PPS is a publish/subscribe middleware for MANET environments. ECCO-PPS offers selective content-based publish/subscribe group communication, where communication channels among applications are to be instantaneous and can be terminated due to changing contexts, node mobility and resource availability. Management of dynamically ephemeral groups enables the establishment of device communities.

Potential applications for ECCO-PPS include directed advertisement for mobile commerce, or dissemination of meta-data about services throughout the network. Meta-data itself can be

disseminated as advertisements, which can then be used by the subscribers to request further services. More critical applications include reporting traffic accident data, or potential crime observation to a nearby police station.

To demonstrate ECCO-PPS, I developed a prototype application *Golf Forum*, which enables real-time content-based publish/subscribe among a gallery during a golf tournament (Fig. 6.8). In this scenario, ECCO-PPS is used as an event dissemination mechanism (see Section 6.10). The access point range is about 200 metres based on 802.11*b* communication, and ad hoc communication expands the communication capability to cover the entire course. In the near future, caddies may wear smart devices, using agents to analyse other players' play such as putting direction and curving statistics, contributing towards strategies for their players.

6.7.1 Architecture

There have been efforts for designing group communication, multicast and publish/subscribe systems in MANETs. Selective message dissemination mechanisms have not been deeply explored yet. Mechanisms exist to broadcast information in a complete network [SCS03] or in a specific geographic area (Geocast) [NI97] [KV02]. However, a selective dissemination mechanism based on the local information held by nodes is still a relatively untouched area.

In dynamic network environments (e.g., with node mobility, capability, and energy constraints), proactive multicast group establishment, maintenance and usage may not be efficient, especially given that membership varies over time, and content requirements can be diverse so that multiple multicast groups are needed.

A popular approach to realise content-based publish/subscribe is to extend on-demand routing of flood messages to all nodes; then, the edge nodes filter out the received messages based on the subscriptions. This approach has the overheads of broadcasting and does not scale well if contents cover wide ranges of attributes or participating members vary over time. Alternative approaches where the source has global knowledge of destination nodes [JC01] can work only in small groups.

Below is an intuitive introduction to the concept of ECCO-PPS and its distinct features.

Data Centric Approach

ECCO-PPS exploits a *data centric approach*, using content addressing rather than using explicit host addressing. This decouples the application level of communication from the underlying transport mechanism. The data model, encoding mechanism, and matching mechanism are key issues in a data centric approach. Furthermore, the routing algorithm in ECCO-PPS controls data traffic based on data contents. This requires data filtering in a timely manner at appropriate nodes to minimise network traffic. ECCO-PPS applies a publish/subscribe communication model with content-based routing in hybrid ad hoc networks, where network environments are highly dynamic.

Combination of Stateful and Stateless Routing

ECCO-PPS follows the basic model of an event brokering system. It includes self-organising components distributed in applications, brokers, and network components. In MANET environments, it is best to create a routing table on-demand according to simulation studies (see [CM99]), and it is crucial to construct the event dissemination structure dynamically for publish/subscribe. Thus, I have chosen the routing algorithm of On-Demand Multicast Routing Protocol

(ODMRP) [LGC99] as a base and extended it in a cross layer form. The basic idea of ECCO-PPS can be deployed with a different on-demand style of multicast protocols.

The routing algorithm in ECCO-PPS is adaptive, combining stateful and stateless approaches to establish publish/subscribe. When no information for global routing is available, the dissemination mechanism is purely stateless, where no global information such as destination node information is required. On the other hand, established routing information is maintained, depending on a defined time period. Thus, a best effort mechanism for dynamic environments is provided, where construction and maintenance of the dissemination structure is minimal, while retaining selectivity instead of broadcasting to the whole network.

In ECCO-PPS, messages are spread similar to a gossip algorithm. It distributes to immediate neighbours that are interested in particular content (susceptible to the gossip), while excluding those that have no match with the message (resistant to the gossip). In this approach, node mobility may help to interact with newer nodes. Mobility causes problems in maintaining route information in the stateful model, while mobility could be helpful for the stateless mechanism. ECCO-PPS resides somewhere between, to combine both advantages.

Context Adaptation to Routing

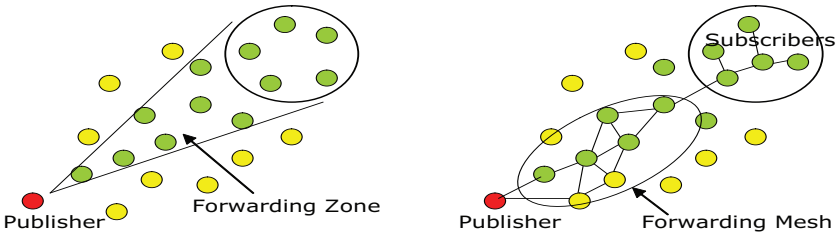
ECCO-PPS supports a content-based publish/subscribe paradigm and provides a proactive dissemination scheme to automatically deliver events in which subscribers are interested. Published events will look for matching subscriptions while travelling over networks. Routing decisions are purely based on locally available information (subscriptions, location, or any other information) and autonomous decisions. ECCO-PPS supports symmetric publish/subscribe, and events may therefore be withheld from delivery, because certain criteria required by the publisher are not found in the local information. Autonomous decisions can be extended to more general mutual consensus of event delivery. ECCO-PPS is self organising, distributed and decentralised in nature. The concept of ECCO-PPS is independent of the underlying communication protocol being used.

Conceptually, events do not need to be typed data. Local information in the subscriber edge brokers can be a series of keywords. Currently ECCO-PPS defines typed events and leaves less structured event manipulation as future work.

[TV04] presents improvements of the flooding approach by applying geographic information and advertisements. Location and context-awareness allow applications to exploit information on the underlying network context for better performance and better group organisation [GM01] [SAW94]. The MANET multi-hop routing protocols consider many contexts from physical constraints, location, mobility, and relative distances for reducing the routing overhead. Many contexts belong to the network, which are outside the scope of middleware. On the other hand, the semantic contexts from upper layers should be used for building an efficient communication for the network layer component. Here, contexts must be exchanged among applications, the middleware tier, and the network layer component to build an optimised data dissemination structure. Thus, I further extend the definition of context to conditions set by the middleware. The current context consists of subscriptions and message advertisements. However, any contexts can be exploited in this scheme.

Cross Layering

ECCO-PPS defines a generic interface to adapt the contexts from the upper layer to the network components. This is to supply data to be attached to message packets and to indicate how to process them. The proposed approach is conceptually similar to Active Networks, which allow



(a) Forwarding Zone in Geocast (b) Forwarding Mesh in ECCO-PPS

Figure 6.9: Location Based Multicast

users to inject customised programs into the network nodes. This model can be abstracted as symmetric publish/subscribe, where message forwarding decisions on the nodes are based on the consensus between the flowing message and local node information.

Summary Based Routing and Dynamic Channelisation

In ECCO-PPS, content-based subscriptions at a subscriber edge broker node can be aggregated and summarised into a compact data format in Bloom Filters [Blo70]. Bloom Filters are used to digest published events and event advertisements.

In publisher edge broker, the dynamic multicast group formation is exploited, based on a snapshot of a global view of subscriptions over the network. Clustering of subscriptions by K-means and other methods defines the multicast groups at the publisher edge broker node.

Efficient Event Structure

It is necessary to design data structures and algorithms for efficient subscription propagation and event matching/filtering. Maximising the expressiveness in publish/subscribe requires a powerful data model capable of capturing information about the events, and of expressing filters and patterns on notifications of interest. Naturally XML is a good candidate, although it lacks typing. Thus, events are defined in XML format with XML schema, and XPath is used as a subscription language. The subscriptions are tightly linked to the corresponding event data structure in the current ECCO-PPS.

6.7.2 Symmetric Publish/Subscribe

ECCO-PPS includes a similar algorithm to *Parametric Probabilistic Routing* (PPR) protocols [CEK03]. PPR applies a limited-flooding strategy, where the message forwarding probability at a node is a function of various parameters rather than a constant such as the normal gossiping algorithm. ECCO-PPS exploits *Symmetric Publish/Subscribe* to provide a dynamic function deciding on message forwarding. This abstraction is novel among publish/subscribe systems in mobile ad hoc networks.

The algorithm for setting the forwarding zone in geocast exploits a similar approach. To avoid blind flooding, almost all the geocast protocols define a forwarding zone. A node that is in a forwarding zone but not in the geocast region, only forwards the packet. A node that is not in a forwarding zone discards the packet. Thus, the whole operation results in the messages that originate from the source node reach all the group members. Simulation of geocast shows a lower message delivery overhead compared to simple flooding. It also shows more accurate delivery, compared to simple flooding, besides reducing the message overhead significantly. Fig. 6.9(a) depicts the forwarding zoned setting in geocast, and Fig. 6.9(b) shows the forwarding mesh in ECCO-PPS.

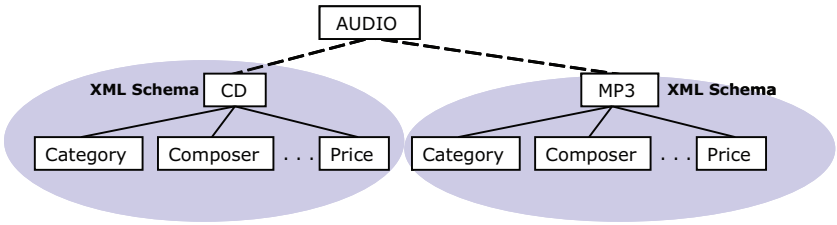


Figure 6.10: Event Type

6.7.3 Publish/Subscribe Model

Designing a subscription model is vital for a publish/subscribe system. Events flow in heterogeneous wireless networks, and providing a minimal level of safety for data interpretation and manipulation is therefore desirable. This can be realised by integrating typed events. It is important to serve high quality primitive events so that composite events will inherit good quality. I use typed events as a base and adapt content-based filtering over the defined type. Type can be defined as a universal type, and pure content-based publish/subscribe is then automatically realised.

Type-based subscriptions work well with typed languages, but it is complex to deploy type-based subscription with the serialisation process of objects in mobile devices. Moreover, mobile applications may not have the concept of objects or typing. The combination of hierarchical

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema attributeFormDefault="unqualified"
    elementFormDefault="qualified"
    xmlns="http://www.cl.cam.ac.uk/~ey204/lib/"
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:simpleType name="categoryType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="jazz"/>
            <xs:enumeration value="classic"/>
            <xs:enumeration value="pop"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="priceType">
        <xs:restriction base="xs:float">
            <xs:maxInclusive value="25.00"/>
            <xs:minInclusive value="10.00"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:element name="category" type="categoryType"/>
    <xs:element name="composer" type="xs:string"/>
    <xs:element name="price" type="priceType"/>
    <xs:attribute name="id" type="xs:int"/>
    <xs:attribute name="timestamp" type="xs:date"/>
    <xs:element name="CD">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="category"/>
                <xs:element ref="composer"/>
                <xs:element ref="price"/>
            </xs:sequence>
            <xs:attribute ref="id"/>
            <xs:attribute ref="timestamp"/>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

Figure 6.11: Event Type in XML

```
<?xml version="1.0" encoding="UTF-8"?>
  <CD id="001" timestamp="1999-02-27T12:00:00.000-00:00"
          xmlns="http://www.cl.cam.ac.uk/~ey204/lib/">
      <category>jazz</category>
      <composer>Evans</composer>
      <price>18.00</price>
  </CD>
  ----------
  <CD id="002" timestamp="1003-02-27T12:00:00.000-00:00"..>
      <category>jazz</category>
      <composer>Davis</composer>
      <price>10.00</price>
  </CD>
  ----------
  <CD id="003" timestamp="1055-02-27T12:00:00.000-00:00"..>
      <category>classic</category>
      <composer>Bach</composer>
      <price>22.00</price>
  </CD>
```

Figure 6.12: Event Instance Examples

topics and high speed content filtering could therefore be a more flexible approach for mobile applications.

XML-based Typed Event

ECCO-PPS uses XML to represent events, which gives rich expressiveness. XML does not have any clear subsumption mechanisms though. Events are defined in XML format with XML schema, and the root element name identifies the event type. The XML schema for the event consists of a set of typed elements. Each element contains a type and a name. The element's name is a simple string, while the value is in any range defined by the corresponding type. Fig. 6.11 shows an example of schema named *CD*, and Fig. 6.12 shows example messages. The W3C recommendation for XML schemata [CFGR02] specifies a set of built-in data types, plus mechanisms for defining types from other types by composition, extension, or restriction. I expect that many pervasive networked data sources will provide data in XML, in a form defined by an XML schema (either a schema written specifically for that data source or a widely used network data type). XML schemata are identified by URLs, which serve as globally unique identifiers.

I exploit *Hypercube* for event/subscription indexing as described in Chapter 4. The current implementation of *Hypercube* is too heavy to incorporate into resource-constrained mobile devices. Thus, a lightweight *Hypercube* for resource constrained devices for wireless networks is desirable and it will be a good future work. Applications are now dependent on XML and RDF for data representation, and expressive query support and indexing are desirable to realise data centric routing.

Subscription Language

Most event systems support a subscription language that allows a subscriber to express its information need. The resulting filtering expression is then used by the brokers to determine whether a particular event notification is of interest to a subscriber. If the language is expressive, efficient matching of notifications will be difficult. However, if the language does not support rich constructs, its applicability is limited.

A subset of XPath [B$^+$04] is used as a filter specification language. XPath provides a syntax for defining parts of XML-based events. With XPath, XML-based events can be seen as a tree of nodes, and a pattern expression identifies nodes. An XPath pattern is a slash-separated list of

child element names describing a path through the XML-based event. Pattern expression over the attribute fields can be defined as a subscription. This functionality gives selective information. Some XPath expressions will be transformed to simplified and unified formats, and complex expressions are limited. Examples of supported queries are shown in Fig. 6.13.

```
1. /CD[category=jazz][composer=Evans]
2. /CD[category=jazz and price<20 and price>15]
3. /CD[category!=jazz] will be transformed to
          /CD[category=pop and classic]
```

Figure 6.13: Subscription Filter for Type *CD*

6.7.4 Summary Based Routing and Compact Event Encoding

Given the constrained MANET environments, it is mandatory to aggregate the set of subscriptions to a compact set of content specifications. The summary of subscriptions and events is flowing over the networks, and a data structure with rich and efficient semantics in compact format is important. Thus, data structures and algorithms for efficient subscription propagation and event matching/filtering are essential. Summarising subscriptions would save network bandwidth and processing cycles for matching. It is a tradeoff between routing efficiency and event traffic. Summaries with lower precision allow more efficient routing, while they produce more false positive event traffic.

In ECCO-PPS, subscriptions are aggregated at the attribute level in the subscriber edge broker nodes and represented in compact data format. The event advertisement and notification are also transformed to a compact data format, which uses XPath as an intermediate expression during the transformation. For encoding data structures, Bloom Filters [Blo70] are used.

Bloom Filters

Bloom Filters are compact data structures for probabilistic representation of a set to support membership queries. Each filter provides a constant-space approximate representation of the set. Errors between the actual set and the Bloom Filter representation always take the form of false positives to inclusion tests. The false positive probability decreases exponentially with linear increase in the number of hash functions and vector size [Blo70],[BM02]. A Bloom Filter is a bit vector, and elements are inserted by setting the corresponding bits corresponding to hash functions. The intersection (AND) and union (OR) operations can be performed directly between these vectors. The inclusion operation checks that all the bits from the hash functions are set. Thus, as the number of elements increases, false positives also increase in the set.

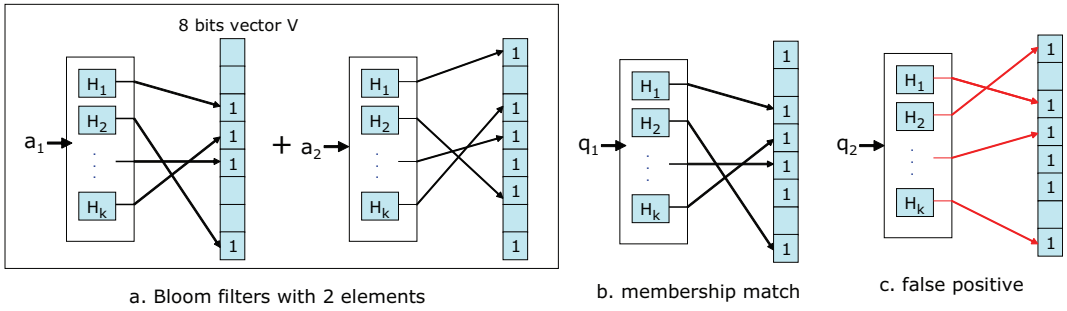


Figure 6.14: Bloom Filters Basic

As an example, for constructing and updating Bloom Filters, let a set A = $a_1, a_2, ...a_n$ of n elements convert to Bloom Filters. Bloom Filters describe membership information of A using a bit vector V of length m. k hash functions, $H_1, H_2, ...H_k$ can be used for each item of A calculating k values, each output results in from 1 to m and the corresponding bit in vector V is set. In Fig. 6.14a, for example, two elements of A, a_1 and a_2, are set in the vector V. When the membership enquiry of q_1 is operated it results in matching, where q_1 has the same value as a_1 (Fig. 6.14b). However, the q_2 membership enquiry also results in matching, even if it does not match either a_1 or a_2 (Fig. 6.14c). This results in a false positive. Typically, the size of the input of the hash function is much larger than the size of the output of the hash function. In ECCO-PPS, for *String Constraint MD5* is used to create the digests of the string constraints for the hash functions fixed at 4 (Fig. 6.15).

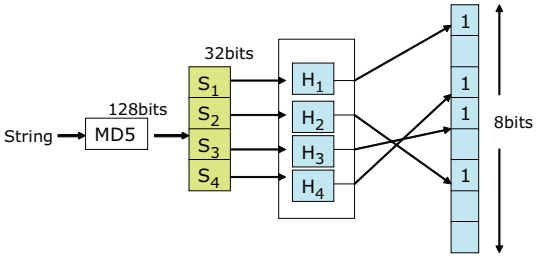


Figure 6.15: Bloom Filters for String constraint

False Positive Probability

The false positive rate depends on the length of the bit vector and the number of keys in the filter. The larger the bit vector, the smaller the probability. The relationship between the number of hash functions and the false positive rate is more complex. Too few hash functions may not provide sufficient discrimination between keys, while too many hash functions make the filter dense, increasing the probability of collisions. It has been shown in [Blo70] that the probability of a false positive is equal to:

$$(1 - e^{-kn/m})^k$$

where n is the number of elements to be stored, k is the number of hash functions, and m is the size of the bit-array.

Subscription Summary

Subscription summary structures are defined in Bloom Filters, using an approach similar to the one described in [TE04], extended to take advantage of XML typed events by XML schema and the XPath subscription language. Currently, no hierarchical encoding of Bloom Filters is used. A subscription summary consists of five data structures described below, and a subscriber edge broker's subscription summary is an array of these data to keep the summarised per-broker subscription information. Examples are shown in Fig. 6.17 and Fig. 6.16.

```
Composer                          ID       EL       SC
-------------------------------------------------------------
domenico scarlatti:               00000001 00010000 11010000
philipp meier, christian zaugg:   00000010 00010000 01001001
el gata's rhythm orchstra:        00000011 00010000 00110100
camerata europeana:               00000100 00010000 10101000
stravinsky:                       00000101 00010000 11000000
gf handel - js bach:              00000110 00010000 01000101
charlotte church:                 00000111 00010000 00110001
```

Figure 6.16: Subscriptions in Bloom Filters

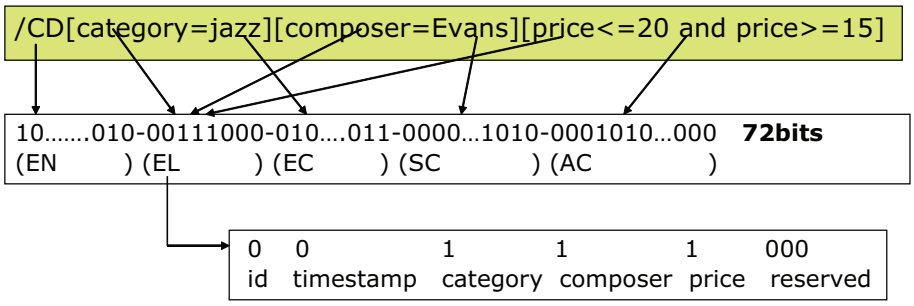


Figure 6.17: Encoding Subscriptions in Bloom Filters

1. **Event Type Name (EN):** contains the hashed value of the root element name in the XML schema that is an identifier of the event type.
2. **Element Association List (EL):** stores information about the elements and attributes in XML schema and actual values that belong together in a subscription. An EL consists of an array of bits with a constant number of columns and a variable number of rows for subscriptions. Columns represent the ordered set of supported elements and attributes defined in XML schema, and the rows represent the unique set of subscriptions. Redundant ELs are eliminated. Each row in an EL includes associated EC (Enumeration Constraint), AC (Arithmetic Constraint), and SC (String Constraint).
3. **Enumeration Constraint (EC):** holds the constraints of each enumerated string attribute of a subscription.
4. **Arithmetic Constraint (AC):** holds the constraints of each arithmetic attribute of a subscription. It consists of two arrays. The first array consists of two columns and a variable number of rows. Each row represents non-overlapping ranges of values specified in subscriptions for the corresponding attribute. The second array is used when an arithmetic constraint has an equality operator for a value that is out of the existing sub-ranges.
5. **String Constraint (SC):** contains information about the constraints in the string type. For each different string type element/attribute, a broker implements a SC structure using three bit vectors (SC_L, SC_R, SC_X) as Bloom Filters. For containment operations, the specified string value is divided into two substrings, 'left' and 'right', defined relative to the position of the operator '*'. After the string value is divided into the two substrings, the left (right) substring is hashed and placed in the $SC_L(SC_R)$ filter for the specific string. If the constraint specified a prefix or suffix operation, the specified string value is hashed, and the result is inserted in the SC_X filter. In case of equality without a containment operation, the whole string is considered as if a suffix operation had been performed, and the hashed result is placed in SC_X. See an example in Fig. 6.18.

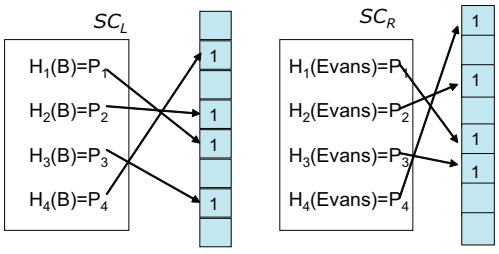


Figure 6.18: String Constraint for $B * Evans$

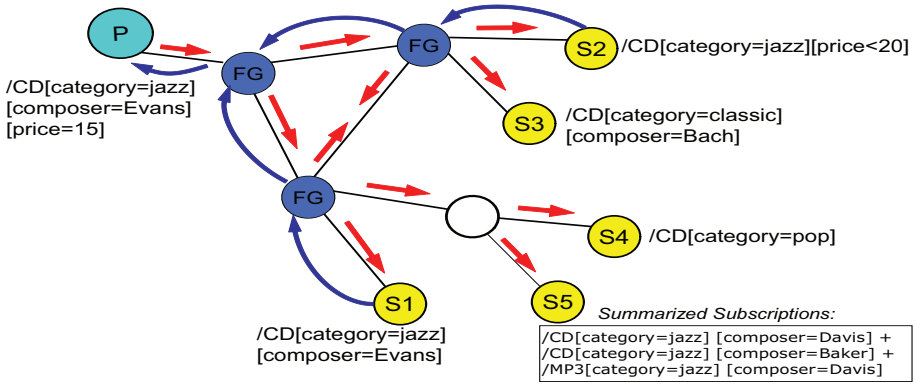


Figure 6.19: ECCO-PPS Routing Overview

6.7.5 ECCO-PPS Routing

This section describes the routing scheme of ECCO-PPS. The topology of a mobile P2P system has to constantly adjust itself by discovering new communication links. It also needs to be fully decentralised due to the lack of a central access point. In such environments, it is best to create a routing table on-demand. ODMRP performs well with regard to throughput and control packet overhead. ODMRP is simple and scalable, by avoiding the drawbacks of multicast trees such as intermittent connectivity and frequent tree reconfiguration. The routing algorithm of ODMRP is therefore selected as a base and extended it for ECCO-PPS.

Routing Strategy

A broker can reside in an independent node or in the same node where publishers/subscribers reside. The routing between a broker and publishers/subscribers can be achieved by unicast protocols that are out of scope of this dissertation. The focus of routing is between the publisher edge broker and the subscriber edge broker.

In general, there are two approaches for disseminating events to the corresponding subscribers. The first is flooding the message by broadcast, followed by filtering at the broker. The second is match-first and requires a precomputed destination list that is broadcast to all brokers followed by routing using the list. The flooding protocol is simple but may lead to network congestion. Match-first is not scalable, because the routing table must be shared by all brokers, and pre-processing may not work well in MANET environments. A MANET environment's dynamic network condition may cause frequent reconfiguration of routing tables. A further possibility is a distributed approach by the brokers, where the brokers examine the message content and forward messages using their routing table. This is bandwidth and space efficient, but establishing and maintaining routing tables can be complex. ECCO-PPS's approach is a combination of distributed matching and scoped flooding.

An overview of the protocol operation of ECCO-PPS is depicted in Fig. 6.19. First, a publisher edge broker node P broadcasts a Join Request (JR) packet (straight arrows) over the network. The event is embedded in the JR packet. The intermediate nodes forward the JR packet.

The subscriber edge broker nodes ($S1$ to $S5$, in which $S1$ and $S2$ match the event notification) decide to join the group based on the matching subscription. After joining, a Join Table (JT) packet (curved arrows) is broadcast. An intermediate node, receiving the JT packet, forwards the packet if it is located between the publisher edge broker node and the subscriber edge broker node. It then becomes a member of the forwarding group (FG).

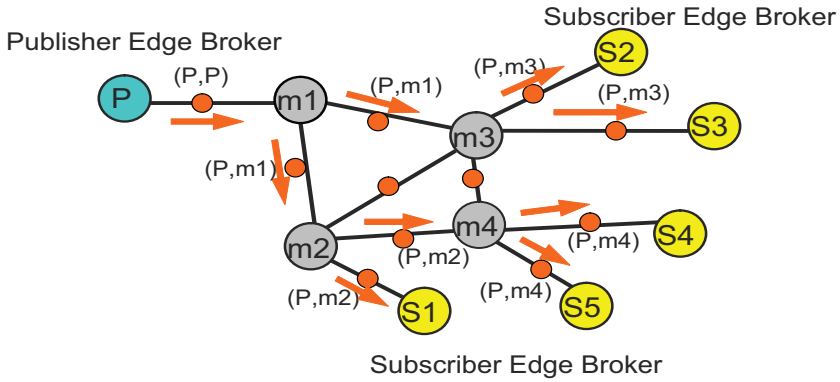


Figure 6.20: Join Request Operation

The JT packet is propagated until it reaches the publisher edge broker via the shortest delay path. A forwarding group is created through the propagation. The publisher broker node operates global grouping from aggregated subscriptions. Afterwards, the publisher edge broker can transmit *Data packets* to subscriber edge brokers via selected routes. The forwarding group creates a mesh for multiple paths. The detailed operations of the routing are described below.

Join Request (JR) Operation

- A publisher broker node (P in Fig. 6.20) wishing to send events periodically broadcasts a Join Request (JR) packet over the network.
- The digest of the event notification to be sent in the Bloom Filter expression (small circle) is attached to the JR packet.
- A node receiving a non-duplicate JR packet stores the upstream node ID and rebroadcasts the packet.
- Optionally, a JR operation provides *advertisement* mode, which sends out a special event to establish the route before sending out the whole data. This can be deployed when the network is more stable and pre-setting routing is beneficial.
- Entries for the routing table are depicted in Fig. 6.20

Join Table (JT) Operation

- The subscriber edge broker nodes ($S1$ to $S5$ in Fig. 6.21) that keep the subscriptions in Bloom Filters have the correct bits set for subscriptions to be recognised as receiver nodes

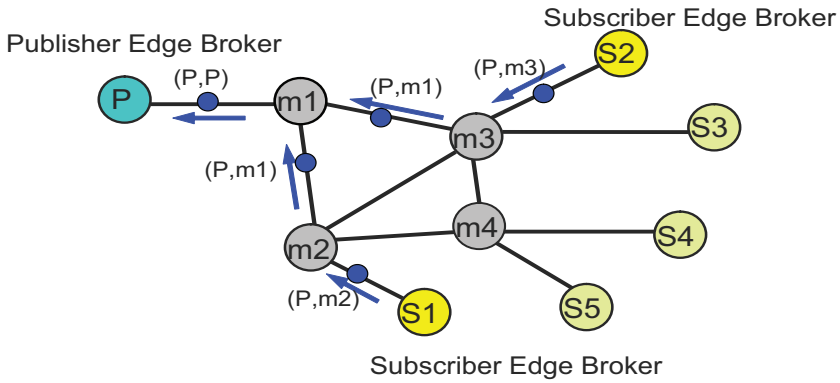


Figure 6.21: Join Table Operation

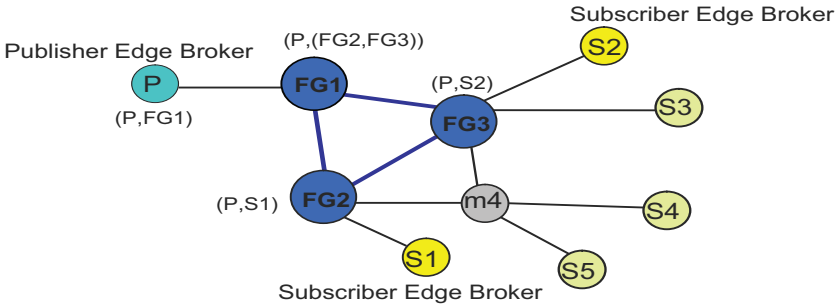


Figure 6.22: Forwarding Group

over the network.

- Once a subscriber edge broker node decides to join the group (i.e., $S1$ and $S2$), it creates or updates the publisher entry in its member table. During this operation, the subscriber edge broker detects only false positive events by matching the local subscriber's information.
- The Join Table (JT) packet is broadcast periodically. The subscription information is attached to the JT packet.
- An intermediate node (router node), receiving the JT packet, compares its node ID with the entries of the forwarding group table. If there is a match, it becomes a member of the forwarding group (FG in Fig. 6.22). It sets a *forwarding flag* (FG_Flag) and broadcasts its JT packet. Optionally, the subscription information is kept alongside the routing information, which is used as a filter for data forwarding. Otherwise, only broker nodes perform the matching operation. The normal gossip parameters can be used at the *router node* instead of filtering.
- The JT packet is propagated by each forwarding group member until it reaches the publisher broker node via the shortest delay path. This process creates a mesh among all forwarding group members.
- The publisher broker node aggregates the subscription information for the group according to the received subscriptions in the JT packet.

Data Forwarding

After the group establishment and route construction process, a publisher edge broker can transmit *Data packets* to the subscriber edge broker via selected routes and forwarding groups. Intermediate nodes relay a *Data packet* only when it is not a duplicate, subscription information for routing destination matches, and forwarding group membership has not expired. When receiving a *Data packet*, a node forwards it ONLY IF it is not a duplicate and the setting of the FG_Flag for the multicast group has not expired. This whole operation minimises traffic overhead and avoids sending packets via stale routes. Also flooding redundancy helps overcome node displacements.

Route and Subscription Maintenance

No explicit control message is required to join or leave the group. When the publisher edge broker node leaves the group, it stops sending a JR packet. The subscriber edge broker node avoids sending back a JT packet in order to leave the group. Forwarding group nodes demote to non-forwarding group nodes if no JR is received within the time-out period.

For maintaining the group, periodic flooding of JR is used to refresh routes and group memberships. Subscriptions are also up-to-date using this mechanism. Subscriber Edge Brokers broadcast JT as a reply to JR or subscription change on the node. This flooding of packets often risks causing *broadcast storm* problems. Optimising the refresh interval is critical for performance and reliability.

When small route refresh interval values are used, route information including membership information is maintained at the cost of introducing more network traffic, leading to congestion. On the other hand, if large route-refresh values are selected, nodes may not know the up-to-date route and multicast membership. Thus in highly mobile networks, large values will result in poor protocol performance. Similarly, the forwarding group timeout interval should be carefully set. Small values should be used for the networks with heavy traffic so that unnecessary nodes can timeout quickly without creating redundancy. Large values should be chosen for a high mobility scenario, that provides alternative routing paths. The timeout value of forwarding groups must be larger than the route refresh interval.

Mobility prediction by ODMRP can be used to adapt the route refresh interval. This scheme uses the location and mobility information provided by a system such as GPS to determine how long routes will remain valid.

Routing Tables

There are four data structures that keep the state of routing:

- **Member Table** (*group ID, source ID, lastJRtime, subscription*): Each subscriber edge broker node stores the publisher edge broker information. Summarised subscriptions are part of this table.
- **Routing Table** (*sourceID, nextHop*): This is maintained by each node. When a non duplicate JR packet is received, an entry is updated.
- **Forwarding Group Table** (*groupID, lastRefreshtime, subscription*): A forwarding group member, which is a routing node between the publisher edge broker and the subscriber edge broker forming forwarding mesh) maintains the group information in this table.
- **Message Cache:** This is maintained by each node to detect duplicates.

Channel Setting

The publisher edge broker node defines the group from the propagated subscriptions. When a new message is published by the publisher, the publisher edge broker looks for any matching group. If the group covers a coarse-grain subscription, more noise would be delivered to the broker, while many groups would be created, if it covers a fine-grain subscription. On the other hand, when a node has high mobility, setting a fine-grain subscription may prevent the overhead of group membership maintenance. Because of the dynamic MANET environment, a crisp and deterministic setting may not produce better performance than a probabilistic approach. It is challenging to define the balance between multicast groups and subscriptions.

The naive solution of the channel setting is using the least constrained subscription information from the propagated subscriptions. This approach may cause inefficient data dissemination under certain subscription conditions (e.g., a subscription subscribes to all events). Setting hierarchical groups on subscriptions and moderating subscriptions to balance network traffic is desirable.

Thus, a dynamic channelisation scheme for aggregating, merging and updating channels for propagated subscriptions is investigated. The results of an experimental study on dynamic

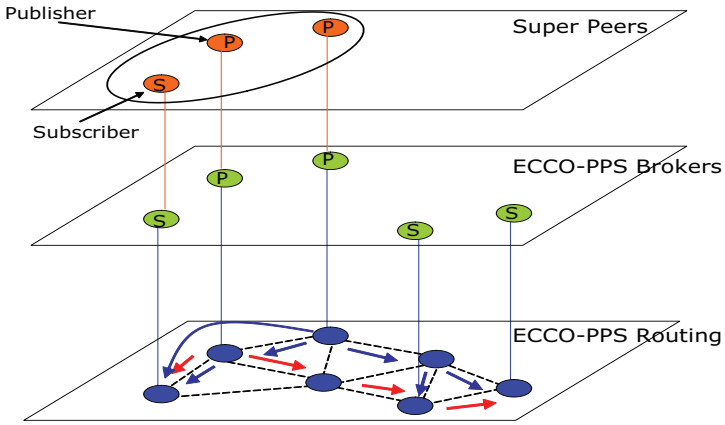


Figure 6.23: Creating Super-Peers over ECCO-PPS

channelisation by clustering subscriptions is shown in Section 6.9.

6.7.6 Mobility and Reliability

ECCO-PPS uses the forwarding mesh inherited from ODMRP, and it provides best effort mesh-based routing. In general, the increase of multicast group size decreases the average loss ratio of packet delivery, so more dense multicast groups will help ECCO-PPS.

The following are ideas for increasing the reliability of ECCO-PPS, but they are for future work. To improve the reliability and fault tolerance addition of a caching (store-and-forward) function at the broker node will be especially helpful. Moreover, applying a pure gossip algorithm to repair lost messages will increase reliability. Thus operational steps are:

- Step 1: ECCO-PPS is used to disseminate the messages.
- Step 2: Node X randomly selects another node Y to gossip the messages it has received.
- Step 3: X and Y exchange missing messages.

This approach does not require membership information. If a non member receives the message, it forwards it to another node. If the member receives the gossip, it decides to accept or forward.

6.7.7 Disconnected Operation and Storage

The current architecture uses *best effort* to disseminate events and the repair of network partition is maintained by the soft state operation without an explicit event-caching operation. To solve this issue, both epidemic dissemination and constructing store-and-forward type of overlay function should be applied, which can be integrated within the super-peers. Consideration of disconnected operation is not currently part of this system, and it should be a future extension.

6.7.8 Super-Peers

Nodes in MANETs are heterogeneous with regard to resource configuration and communication ability. Constructing an overlay exploiting the different capabilities in peer nodes can lead to an efficient network architecture, where a subset of peers, called super-peers, takes over specific responsibilities for peer aggregation. Super-peer candidate nodes may be a gateway, sink node, publisher edge broker node, or resource rich node, possibly connected for long periods. Among super nodes, different media-based communication or different communication channels

are possible, to exchange more control information. Super-peers can be organised based on DHT protocols or any other logical topology. Fig. 6.23 depicts a super-peer overlay, where ECCO-PPS routing, a broker overlay, and a super-peer overlay form a 3-tier architecture.

In ECCO-PPS, the purpose of super-peers is to aid underlying publish/subscribe communication without involving routing mechanisms, while the normal super-peer infrastructures exploit a two-phase routing [NT04]. The super-peer backbone in ECCO-PPS will be useful for caching, supporting disconnected operations, or supporting location information. Super-peers can be semi static nodes, which may provide geographic information to the regular nodes. The super-peer backbone can also operate gossipping mechanisms over the underlying publish/subscribe to increase reliability. The super-peer backbone in ECCO-PPS is essentially an application using ECCO-PPS.

6.7.9 Summary

I have introduced ECCO-PPS, a structureless asynchronous group communication system over wireless ad hoc networks. ECCO-PPS provides context adaptive casting using restricted flooding with a cross layer approach between middleware and the network layer. This realises the symmetric publish/subscribe paradigm. Controlled flooding, policy based routing, and parameterised gossip-based routing address similar issues; these are key approaches to implement more efficient routing based on various contexts. Thus, the main idea can apply over different types of networks. Data centric communication abstraction over heterogeneous wireless networks will have substantial impact for constructing reactive distributed applications. I demonstrated controlled flooding in an early trial of publish/subscribe systems in wireless network environments [YB04a], [YB04b]. DHT may not work well in dynamic MANET environments, but the use of super-peers creating an overlay network on top of dynamic epidemic based networks (e.g., ECCO-PPS) enables the addition of multiple functions (e.g., storage, directory).

6.8 Experiments

A Java-based prototype of ECCO-PPS have been developed over a Java ns-2 simulator (JNS) [JNS02]. This section first discusses simulation environments and then shows experimental results.

6.8.1 Simulator

Integrating ECCO-PPS with network layer components in mobile devices is challenging. Thus, simulation environments are looked into, where the simulator can satisfy the following criteria:

- **Group based mobility support:** In the majority of existing network simulators, node mobility and network topology are based on uniform random movement, which does not reflect the real world. For the experiments a mobility pattern like gallery movement of the golf tournament is desired, where constant group members are partitioned but mostly connected. Depending on the movement of member nodes, connection topology changes within a group, but more or less the same group members form the group throughout the tournament. Each group may be partitioned and any node may join the partition and go away from the partition. Ad hoc vehicle communication over the highway can be modelled in a similar way.
- **Discrete event generator:** Verification of the network routing protocol is part of the experiment, but flexible event generation based on publish/subscribe characteristics is a more important part of the experiment. This requires the specific event generation on each

publisher node, tightly related to the network topology. The data trend is an important factor for the simulation, but without writing our own data generator for the simulation environment, there is no standardised input data for the simulation. The uniform generation of data may not reflect a real world situation. It is important to enable repeatable and comparable simulation results. The data trend includes a query pattern (e.g., the number of queries issued, interval, distribution of query popularity).

- **Java Based:** Ease of event generation of publish/subscribe events can be done in Java. ECCO-PPS uses XML for event structures as base.

MANET protocol simulation environments vary and there are no standardised methods. In [KCC05], the current simulation environments and credibility are discussed in detail. I did not find any simulator satisfying the above requirements when first checked in 2003.

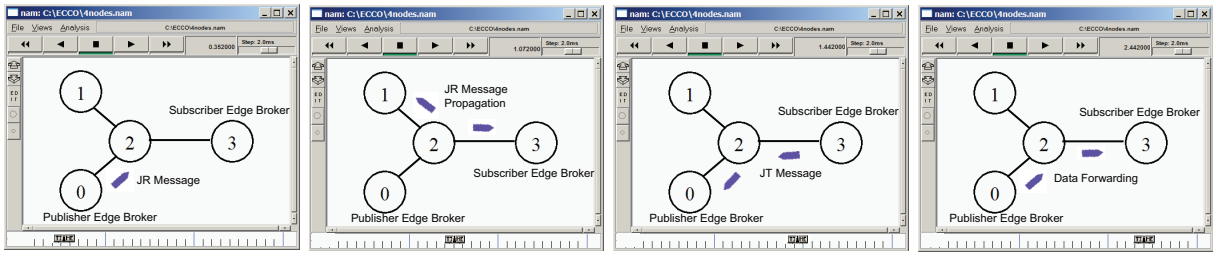
It is not intuitive to implement wireless routing protocols within current mobile devices for experiments, and a reasonable size of system cannot be tested with the available resources. Thus, testing has to rely on simulation. After consideration, I decided to use JNS to carry out experiments using simple methods to highlight the characteristics of the approach taken. JNS has the capability for real world applications using the simulated network protocol, and it is Java-based. I created a small testbed that reflects a realistic scenario. However, JNS does not support mobility, and a dynamic mobility scheduler is therefore added.

JNS

JNS is a Java implementation of the original ns-2 simulator. It is not as complete as ns-2 but is a lot more accessible without using Object Tcl. In JNS, new protocols can be simulated in a controlled environment. JNS produces a trace file (with format as NAM trace files) that can be viewed in a network animator (e.g., Network Animator [Ani]). JNS supports a simulation of a real world implementation. The simulator can handle IP packets. If a packet destination address is within a range of multicast addresses, the packet will be copied and sent to all interfaces of the node. Thus, real world applications are able to run over the simulator. JNS allows dynamic scheduling of events, and the dynamic scheduler that interfaces Java Remote Method Invocation (RMI) to a virtual multicast socket can behave like a normal Java unicast/multicast socket.

Simulator Validation

This section shows the base case of ECCO-PPS simulation with JNS to validate the basic operation of the simulator. The network consists of four nodes: a publisher edge broker node, two subscriber edge broker nodes, and a router node. The topology is depicted in Fig. 6.24. No mobility is scheduled. The configuration values for soft state maintenance are set to 0, thus no dynamic topology reconfiguration is operated. The ECCO-PPS operation follows the steps below:



(a) Broadcast JR message (b) Propagate JR message (c) Subscribe with JT message (d) Data Forward via Route Table

Figure 6.24: Base Case of ECCO-PPS Simulation with JNS

- Join Request (JR) message is dispatched from the publisher edge broker node 0 (see Fig. 6.24(a)).
- JR message is propagated to all the nodes $1, 2, 3$ (see Fig. 6.24(b)).
- Nodes $1, 3$ are subscriber edge broker nodes. Node 3 contains matching subscription, and returns Join Table (JT) message (see Fig. 6.24(c)).
- Data forwarding to the subscriber edge broker node 3 is operated via the constructed route (see Fig. 6.24(d)).

The NAM trace of the above operation is visualized in Fig. 6.24 and it validates the base case of simulation correctness.

Mobility Enhancement

To support mobility, a special dynamic scheduler that takes node link up/down operations was added to JNS. Link up/down operations are operated according to the configuration, which is provided as an input to the mobility scheduler. The configuration includes the attributes on each movement of each node: *Node ID, Current Link Node*, and *Next Link Node*.

Multiple configuration files can be created for individual nodes, and separate schedulers can be assigned to schedule parallel operations (see Fig. 6.25). Configuration files can be generated from any mobility patterns. No major mobility patterns such as Random Waypoint [JMH02] are currently implemented. One mobility pattern is supported that includes the following characteristics:

- The ad hoc group moves together in a loose fashion, and the formation of the group constantly changes. There are no node partitions.
- Mobility occurs on the receiver nodes, and they are always connected to at least one other node.

Fig. 6.26 depicts the group mobility outline, where the node G moves around but always connects to one of the other member nodes in a group and sometimes connects to the access point that publishes messages. The location state information of each node is kept from the initial node connection topology, which is kept in a matrix. The given input *Node ID* and *Speed* determine the next position of the node within the topology, with a random choice of next link node. The region for the node movement is restricted within the original matrix. The mobility speed feeds the interval between *Link Down* and *Link Up*. This operation generates the configuration file for the mobility scheduler.

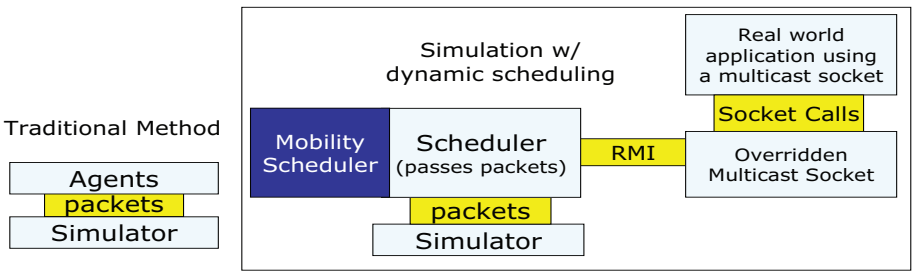


Figure 6.25: Components Structure of Simulator

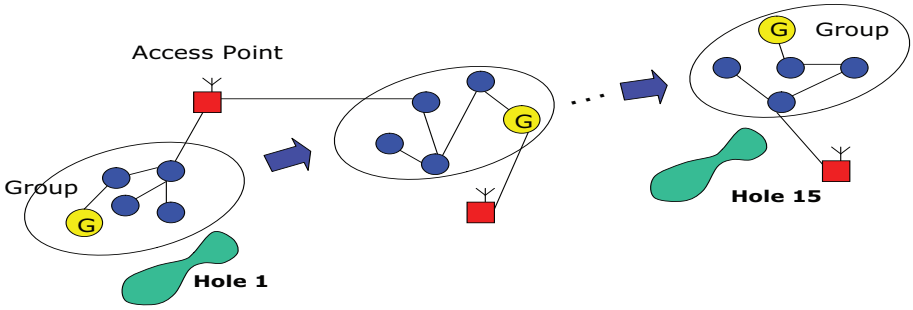


Figure 6.26: Group Mobility

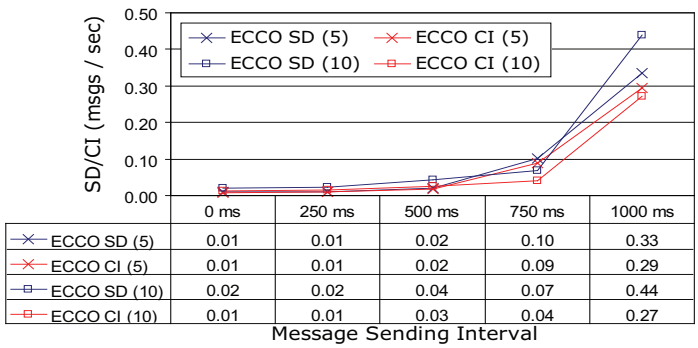
Thus, it appears that subscriber nodes move around to connect to different forwarding nodes. The current mobility support is not optimal, but the goal of the mobility support is to obtain useful experimental results on the relation between mobility and subscription patterns, which will provide useful information for optimising the balance between group formation and subscriptions.

6.8.2 Simulation Setup

A dedicated PC (Intel Pentium M 760 - 2GHz - RAM 1GB) is used for the experiments. The experiments are done through real applications running on top of JNS, thus the limit of the number of applications running on the single machine restricts the scale of the experiments. On the other hand, extracting the characteristics of ECCO-PPS can be done within the given environment. The basic topology as described in Fig. 6.19 has been used unless described otherwise. The following values are used for the configuration of ECCO-PPS.

- Join Request Refresh Interval: 5 seconds
- Forwarding Group Timeout: 25 seconds
- Route Timeout: 5 seconds
- Data Rebroadcast Interval: 25 milliseconds

A publisher edge broker sends 100 messages containing five different types of messages that uniformly match 60% of subscriptions with an interval of 500 milliseconds unless stated. Each experiment is executed between 5 and 20 times. Standard deviation (SD) and Confidence interval (CI) values are calculated for each experiment. Fig. 6.27 depicts SD and CI values for ECCO-PPS in the experiment described in Section 6.8.3 (see Fig. 6.28). Fig. 6.27 shows that the CI from 10 executions is similar or lowers than the CI from 5 executions, and values within the range of CI does not give any impact to the outcome of the experiments. 5 executions are therefore



	0 ms	250 ms	500 ms	750 ms	1000 ms
ECCO SD (5)	0.01	0.01	0.02	0.10	0.33
ECCO CI (5)	0.01	0.01	0.02	0.09	0.29
ECCO SD (10)	0.02	0.02	0.04	0.07	0.44
ECCO CI (10)	0.01	0.01	0.03	0.04	0.27

Figure 6.27: Standard Deviation and Confidence Intervals for ECCO

sufficient (see Section 4.7.4.3). Each simulation is executed minimum 5 times, and the data are the average over those runs. Because this experiment is executed with real network functions, the overhead of any substantial matter (e.g., CPU, network resources) may impact the experimental results. However, the results should still show the essential characteristics of the system. A real world experiment with resource constrained mobile devices in an 802.11b ad hoc network has been partially done with a few devices and this is out of scope of this dissertation.

6.8.3 ECCO-PPS vs. Multicast with Predefined Channels

Using the original ODMRP, there are two ways to implement a publish/subscribe system:

- ODMRP with one channel: uses a well known channel, and every subscriber edge broker node operates the subscription matching process.
- ODMRP with N channels: predefines channels for each subscription. In this case, five channels are defined and this approach is not a content-based subscription.

Fig. 6.28 shows the throughput of messaging systems with the above two multicast approaches and ECCO-PPS. The X axis indicates the message interval. The throughput is measured when all messages are delivered to the target subscribers. The conversion time from XPath to Bloom Filters is included. The filtering operation to remove the false positives is also included in the experiment. ECCO-PPS improves the performance when the message interval cancels out the conversion overhead. Single channel ODMRP suffers from the overhead of subscription matching.

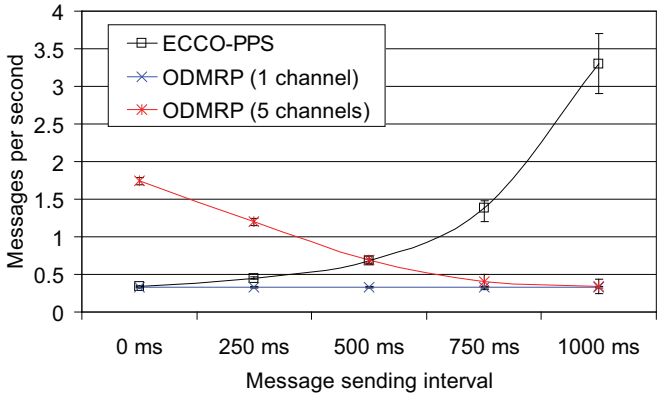


Figure 6.28: Throughput Comparison

Fig. 6.29 shows the network traffic over three systems: the number of bytes of total incoming packets. Brokers 1, 2 and 4 have matching subscriptions. Because of the simple topology, the unmatching brokers receive high volumes of packets from the flooding of control packets. ECCO-PPS shows comparable throughput to the predefined channel-based multicast. The single channel ODMRP gets twice as much network traffic as the others.

6.8.4 ECCO-PPS Conversion Overhead

ECCO-PPS requires a conversion process from message to XPath expression. Ultimately, both subscriptions and messages in XPath should be transformed to Bloom Filters. The conversion

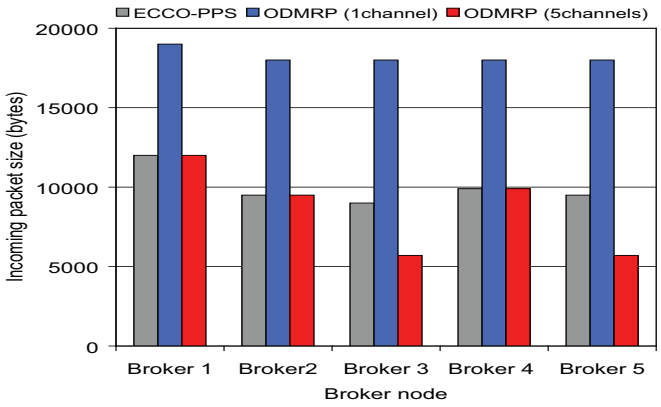


Figure 6.29: Network Traffic Comparison

overhead depending on the complexity of the XML schema is measured. This was done separately from the simulation experiments. The result indicates that the conversion overhead stays constant within the range of 200 nesting elements in the XML schema. The comparison between the message notification and the aggregated subscriptions takes only a fraction of time, since it compares simple bit sets.

Fig. 6.30 shows the average number of delivered messages among subscriber edge broker nodes. It highlights the turning point, at which the conversion overhead gets negligible.

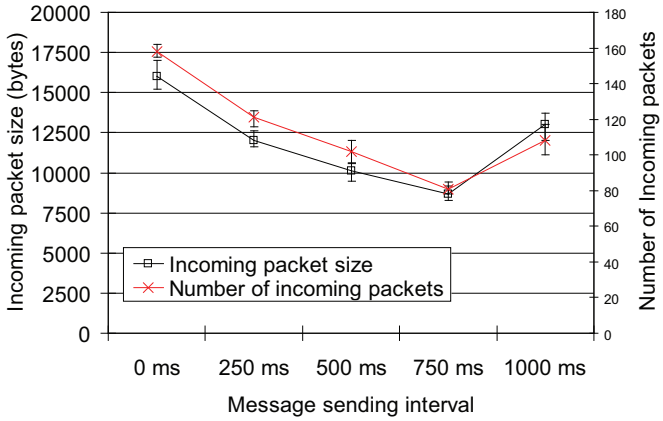


Figure 6.30: Number of Messages vs. Sending Interval

6.8.5 ECCO-PPS Scalability

Fig. 6.31 shows the throughput for the different numbers of subscribers. The throughput stays the same in spite of increased network traffic. If an intermediate node is in a resource constrained device, the node may cause congestion for the whole system throughput. Thus, this experiment may show different results in real world deployment with mobile devices. Error bars on the line of message number is too small, therefore it is not shown.

The scalability of the group size is not optimal in ECCO-PPS, because it inherits the characteristics of ODMRP. The packet delivery ratio significantly decreases as the multicast group

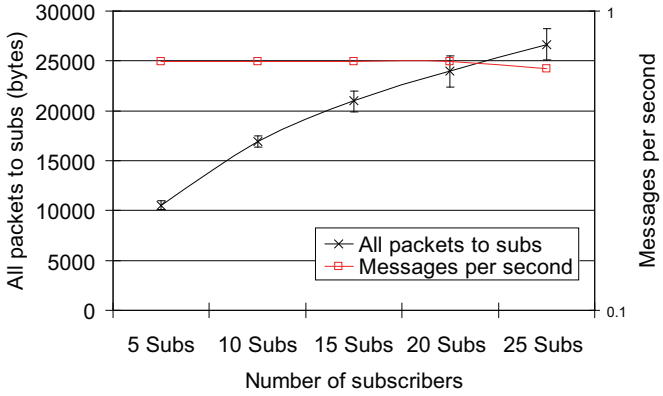


Figure 6.31: Number of Subscribers over ECCO-PPS

increases to fifty members (see [L+00] [WC02]). This can be attributed to collisions that occur from frequent broadcasts through the network.

6.8.6 ECCO-PPS: Mobility and Reliability

The proportion of messages delivered correctly is related to node mobility in Fig. 6.32. Note that the delivery ratio is high at human walking speed. High mobility gives more impact to ECCO-PPS than ODMRP with a single channel. The figure shows that overhead from complex data management is affected when nodes move fast.

The message delivery ratio with various message sending intervals over different mobility speeds is shown in Fig. 6.33. Note that the X axis indicates the mobility speed; it is not linearly scaled over the entire range. The range of speed is selected between 7.5 km/h and 75 km/h on the X axis. This also applies for the next experiment with the range between 7.5 km/h and 150 km/h. When the mobility speed is high, message loss is unavoidable. Note that the speed of *7.5km per hour* is about twice as fast as human walking speed.

The second experiment focuses on the reliability of mesh topology. The subscriber edge broker node connects to one or more nodes in the forwarding group, where the forwarding group builds a mesh. When messages are delivered to the subscriber edge broker through the forwarding group, the shortest path is used if available as shown as the *Primary Route* in Fig. 6.34. If the

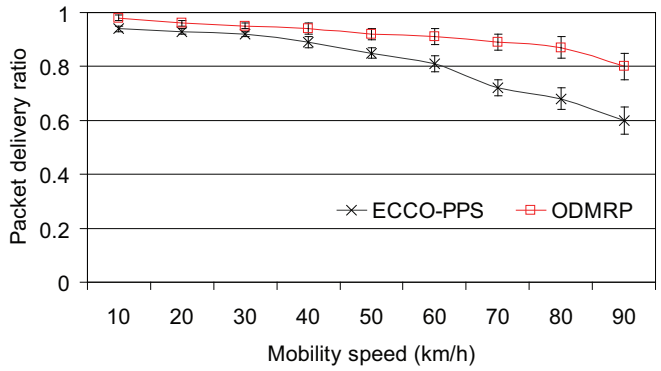


Figure 6.32: Message Delivery Ratio over Mobility 1

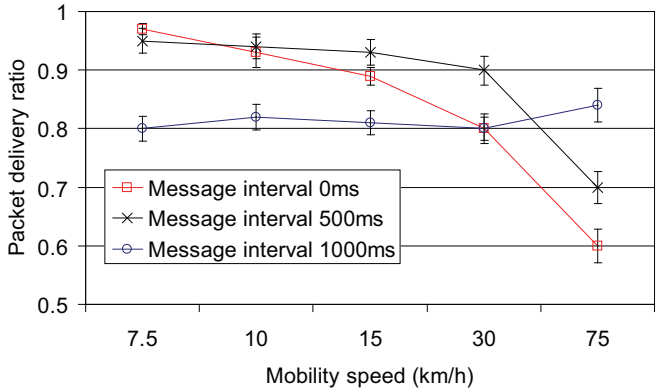


Figure 6.33: Message Delivery Ratio over Mobility 2

shortest path is not available, an alternative path is used, shown as the *Alternative Route* in Fig. 6.34. This figure shows that the mesh topology gives 100% message delivery rate and 100% mesh usage. High mobility speed causes frequent unavailability of the primary link, and the alternative route is used on that occasion. The throughput, incoming packet size, and incoming packet number show constant values in this experiment.

In the third experiment, the receiver broker node moves repeatedly around from node 1 to 9, and the link with these nodes changes accordingly. Fig. 6.35 shows that when the subscribing broker nodes move faster, messages are carried by the JR packets rather than Data packets, indicating

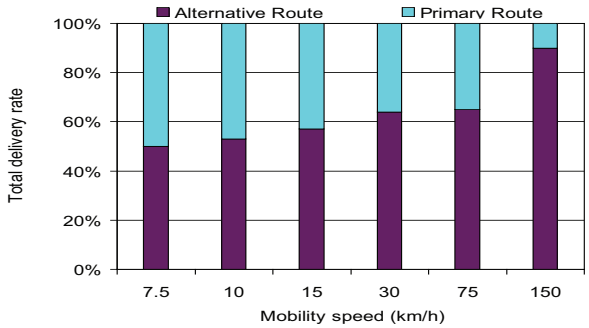


Figure 6.34: Reliability Increase by Mesh Topology

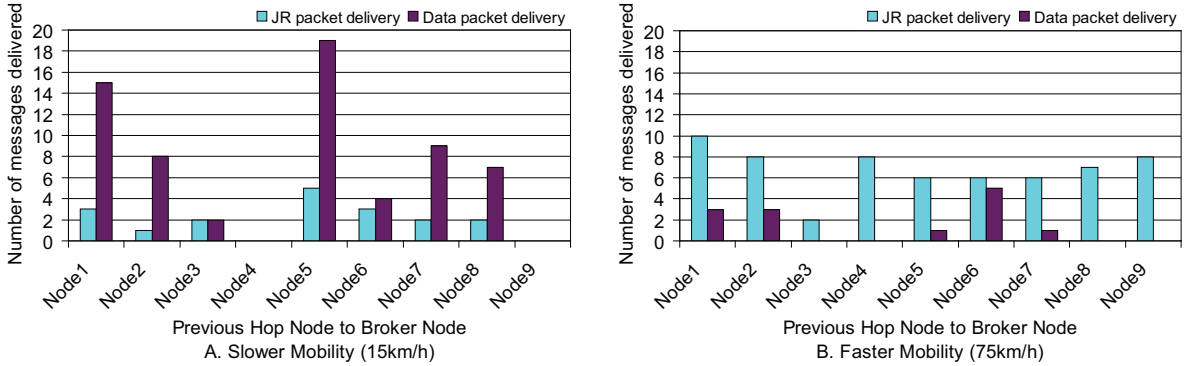


Figure 6.35: Message Delivery by Join Request Packet and Data Packet

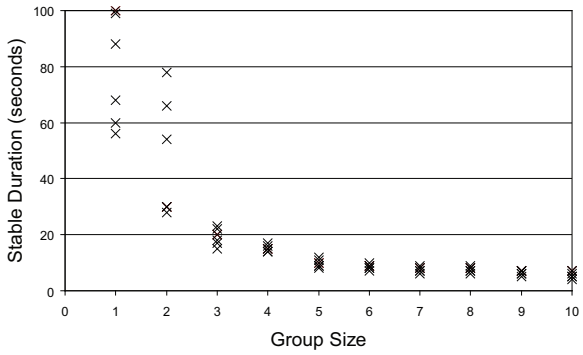


Figure 6.36: Group Stability Clustering

that the established route cannot be used because of node mobility. The packet delivery ratio is 0.82 in Fig. 6.35A, and 0.93 in Fig. 6.35B.

6.8.7 Group Stability

Group stability indicates the time for a group to stabilise a group (i.e., no subscription is propagated to the publisher edge broker). If it is too short, there is not enough time for a potential channel split/merge operation. Thus, the duration impacts the finalisation of group formation. Group stability is also affected by mobility: the faster the mobile devices move, the less stable the group is. Fig. 6.36 shows the time a group becomes stable, where the X axis indicates the number of group members. Stability decreases rapidly with increasing numbers of group members. In this experiment, the average mobility speed is set to $2.5km/hour$, which is the speed of slow pedestrians. The radio transmission range of each device is 50 metres. A group with 2 people exists on average 30 seconds, whereas a group with 5 people is only stable for 9 seconds. Even a group only stable for 9 seconds still allows ECCO-PPS to operate channelisation.

6.8.8 Routing Characteristics

There are several comparison studies of multicast in MANETs as well as a comparison with flooding [L+00] [BEK+05] [BMJ+98] [DPZ04] [DPRM01] [WC02]. According to these studies, mesh-based protocols seem to outperform tree-based protocols in high mobility situations. Flooding gives the best reliability all the time, but with a high overhead of network traffic. On-demand based protocols suffer from scalability issues as the size of the multicast group increases. Maintaining a mesh costs more with more senders. Self-pruning of on-demand protocols should decrease the control overhead.

ECCO-PPS inherits the routing characteristics of ODMRP, and it provides various ways of pruning for the flooding of packets. Reference [V+06] provides an informative comparison of flooding based routing with scoped flooding such as ODMRP and MAODV.

Table 6.1 shows a comparison between flooding, ODMRP, and DHT-based approaches. ECCO-PPS is conceptually similar to gossip algorithms. When the context of ECCO-PPS is geographical information, the function of ECCO-PPS is similar to GHT based routing or geocast.

6.8.9 ECCO-PPS with Geographic Context

Fig. 6.37 depicts ECCO-PPS with the geographic context as an input for routing. The publisher

Protocol	Route Construction Overhead	Route Maintenance OverHead	Redundant Packet Forwarding	Reliability	Scalability	Mobility Resistance
Flooding	Low	Low	High	High	Low	High
ODMRP	Medium	Medium	Medium	High	Low	Medium
DHT-Based	High	Low	Low	Medium	High	Low

Table 6.1: Routing Strategy Comparison

edge broker has an advertisement agent as a publisher to deliver advertisements based on where the subscribers locate and their subjects of interest. The agent can use the *advertisement* for grouping subscribers, and based on the groups the agent disseminates the advertisements. For example, subscribers who are interested in the subject *Computer* and happen to be in the area code *CB3* near the computer laboratory can be grouped in the circle depicted. The functionality is the same as for the *geocast*, where multicast delivery is targeted to the receivers residing in a specific geographic location.

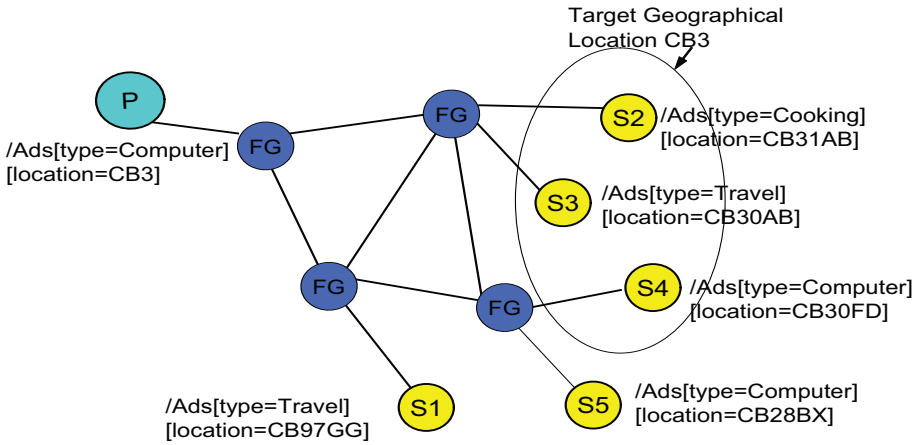


Figure 6.37: ECCO-PPS with Geographic Context

6.8.10 Bloom Filter Effect

This experiment is done separately from the network routing experiments to examine the feasibility of Bloom Filters. A subset of *FreeCD Database* (http://www.freecddb.org/) is used. The database contains CD/DVD information including the title, name of composer, release year, price, etc. I experiment with various dimensions of subscriptions with the FreeCD database using Bloom Filters to explore their effectiveness for real data. The number of hash functions for the Bloom Filters is 10. To simplify the experiment, a single attribute is set to an event/-subscription. The experiments produce deterministic results with identical inputs for repetitive execution. Thus no error bars are shown in figures in this section. Fig. 6.38 shows the string sizes of composer names, and Fig. 6.39 shows the numbers of collisions and false positives over three different input string sizes: 36 bytes, 48 bytes, and 72 bytes. The bit vector size is set to 8. The X axis indicates the size of the event set.

In Fig. 6.39(b), the numbers of false positives is the sum of the false positives on all the events over the whole event set. Thus, the false positives increase more than the size of the event set. The majority of sizes for composer names is around 10 to 20 bytes. Different input string sizes for Bloom Filters do not show any difference.

Fig. 6.40 shows the collisions and false positives on various Bloom Filter vector sizes with 36

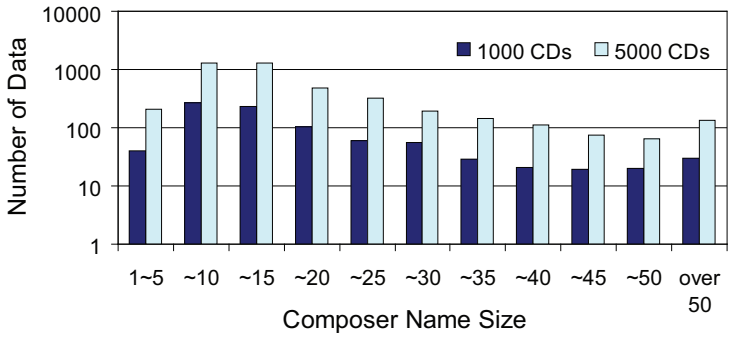
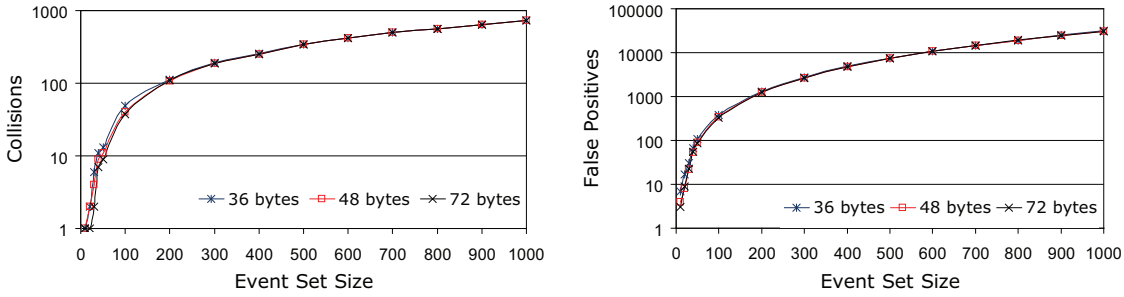


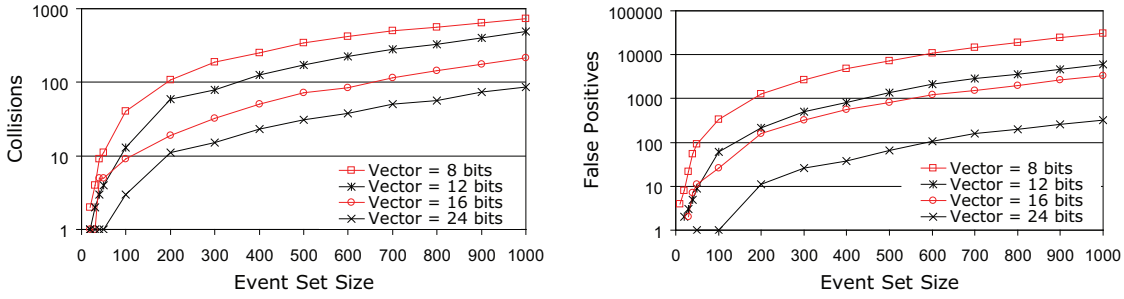
Figure 6.38: Size of Composer Name for Bloom Filter Input



(a) Collisions (b) False Positives

Figure 6.39: False Positives vs. String Size

bytes of composer name. The changes of Bloom Filter vector sizes show a strong impact on false positive rates. Overall, false positive rates increase with increasing size of data and decrease exponentially with linear increase in vector size.



(a) Collisions (b) False Positives

Figure 6.40: False Positives vs. Bloom Filter Size

Fig. 6.41 shows the false positive rate, instead of numbers from the same experiment as in Fig. 6.40(b). The Y axis shows the false positive rate, where the value 1 is 100%. Thus, with 24 bits in the vector, it shows 10% of false positives over a 500 event set and with 8 bits in the vector, an extreme rate (1000%) of false positives over a 500 event set. The false positives are the sum of false positives of each event over all events, and therefore the rate can go up more

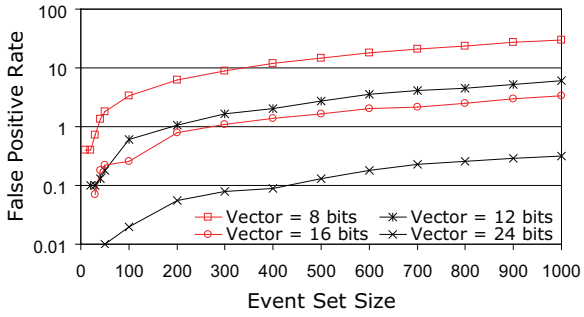


Figure 6.41: False Positive Rate vs. Bloom Filter Size

than 100%.

In a mobile ad hoc environments, the upper threshold of the false positive rate could be less than 50%, depending on the applications. The experiments show high Bloom Filter false positive rates over a short bit vector. This made us consider the use of RTree that preserves better aggregation of subscriptions, if it can realise a lightweight data structure for the pervasive version. With RTree, range aggregation is possible while Bloom Filters have no means to aggregate range subscriptions.

6.9 Dynamic Channelisation

To maintain the subscription semantics for multicast groups, the publisher edge broker uses the propagated subscription information carried by the Join Table (JT) packet. Thus, the publisher edge broker node has knowledge of all subscriptions along the line of its published events and maintains the event dissemination channel with that. Ideally it is desirable to optimise channelisation, using global knowledge over the network and among the publisher edge broker nodes. When the system evolves (e.g., joining and leaving of the publisher or subscriber and/or node failure), multicast groups are reconfigured to maintain the desired level of system efficiency. The set of overall subscriptions changes over time and an overall view of subscriptions is necessary, in order to compute at run time the multicast groups that better represent the group communication. This includes split/merge of channels.

Imagine that you are moving in a crowd and look for people to share a taxi ride, or you are a special advertisement agent and look for target devices to send advertisements related to nearby stores. You advertise events, and, depending on the response, several groups can be formed (e.g., taxi ride to Cambridge North, to Cambridge East). Dynamic channelisation will automatically create multiple channels, based on the similarity of interests (i.e., subscriptions).

The publisher edge broker can introduce summarisation of multicast channels, where imprecise summarised subscriptions are assigned to a channel, or channels are very fine-grained such as mapping to each subscriber. This is a tradeoff between routing efficiency and network traffic: summaries with lower precision would give more efficient routing, with a higher amount of false positive event traffic.

This section discusses tradeoff solutions between expressiveness and efficient dissemination. To make summaries more compact and effective, subscription clustering is attempted. This leads to dynamic creation of multicast channels for optimisation of system throughput.

6.9.1 Grouping Subscriptions

In general, designing content-based publish/subscribe needs to consider issues to balance channelling and filtering: maximising expressiveness of subscriptions and scalability to the number of subscribers based on event space size. It is challenging to group common subscriptions and adapt them to the network topology.

In content-based publish/subscribe, a naive approach maps subscriptions to 2^N multicast groups (N = number of consumers). Reducing the number of required groups can be done by reducing the precision using filtering on subscribers, dividing subscriptions into groups, or creating subsets of neighbouring subscribers. In Gryphon [OAA+00], the following algorithms have been described and experimented with:

- Optimal Algorithm: events are mapped to the exact group of subscribers, which increases the number of multicast groups.
- Flooding Algorithm: events are propagated to all subscribers and filtering is performed at the subscriber end.
- Clustered Group Multicast Algorithm (CGM): divides subscribers into exclusive subsets, and each subset has its own multicast group.
- Threshold Clustered Group Multicast Algorithm (TGCM): combination of flooding and CGM, which behaves like CGM unless the number of destinations in a cluster exceeds a threshold, in which case the cluster is flooded.
- Neighbour Matching Algorithm: each node performs tests on an event to determine which subset of its neighbours is on the next hop to a final destination. This requires extra bandwidth and processing, which increases latency.
- Group Approximation Algorithm: combines actual groups with approximate groups, which minimises wasted events (i.e., observing groups which have a relatively low probability of receiving events, and sorting groups according to their probability of receiving events).

Gryphon's main focus is on mapping matching subscriptions to the network level of multicast groups in wide-area network environments. It evaluates network issues and the economics of equipment deployment. For multi-party applications, [WKM00] proposed using K-means to group subscribers with sets of publishers they are interested in, to minimise wasted event traffic.

Clustering in ad hoc networks usually indicates clustering the nodes so that hierarchical routing schemes can control efficient broadcast. Thus, clustering schemes are based on the location or the topology. The proposed subscription clustering has much in common with subscription partitioning and routing in wide area networks. In this section, clustering the semantics of subscriptions to classify the subscriber nodes according to common subscriptions is attempted.

In [WQA+02], two approaches to partitioning the overall publish/subscribe operations among multiple brokers are discussed:

- Regular partitioning of the space: Event Space Partitioning (ESP)
- Clustering of the subscriptions: Filter Set Partitioning (FSP)

In the Event Space Partitioning (ESP) approach, the d-dimensional space is partitioned into N_s disjoint subspaces. Each space is mapped to a different server. If a subscription intersects the subspace owned by the server, the subscription is kept in it. When the server's subspace contains an event, the event is forwarded to the server, which is responsible for the subscription.

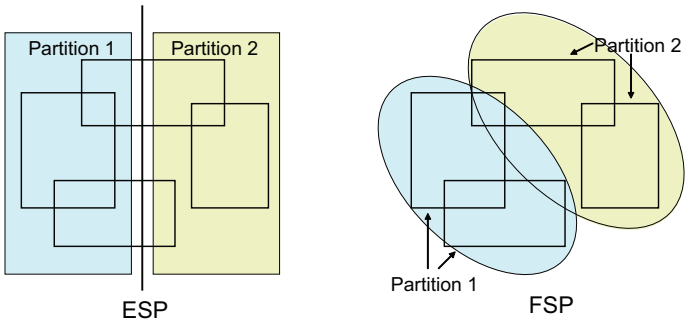


Figure 6.42: Filter Set Partitioning and Event Space Partitioning

To eliminate event traffic to a subspace, summary filters can be used. The ESP approach minimises event traffic by propagating events to the responsible server for the events. However, if more than one subspace contains subscriptions, replicated subscriptions are kept in the multiple servers. This causes complex maintenance task of subscription management.

Filter Set Partitioning (FSP) takes a dual partitioning approach assigning each subscription to a single server. Subscriptions can be grouped with similar subscriptions and assigned to the same server so that compact and effective summary filters can be constructed. An event is forwarded to all servers containing matching summary filters. Figure 6.42 depicts the difference between these two approaches for 2-dimensional subscription filters.

In [RLW+02], efficient communication schemes based on multicast techniques for content-based publish/subscribe systems are shown by applying clustering algorithms to form multicast groups. These algorithms perform and scale well in the context of highly heterogeneous subscriptions. Iterative clustering algorithms (K-means) seem to be better suited to subscription dynamics. Hierarchical clustering algorithms have poorer performance than iterative clustering but improve performance by subdividing the existing groups when more multicast groups become available.

My goal is similar to the FSP approach. For content-based publish/subscribe, the FSP approach has shown good load balancing in both static and dynamic environments, while significantly reducing network traffic. The purpose of clustering for multicast grouping is to reduce the network load, and it is not necessary to create a hierarchical cluster even if the propagated subscription is in the hierarchical formation. Another aspect is that subscriptions may reach the publisher edge broker nodes in a time series, and the algorithm needs to accommodate periodical update.

Aggregation of subscriptions takes place over time and space. Responses from the subscriber brokers arrive at the publisher broker node one by one, which makes multicast group construction complex. If the group covers a coarse-grain subscriptions, more noise will be delivered to the broker, while many groups will be created if it covers fine-grain subscription. However, if a node has high mobility, setting a fine-grain subscription may prevent the overhead of group membership maintenance. The goal is to define a channel per entity, but there is no obvious one-size-fits-all solution. It is challenging to define the balance between multicast groups and content-based subscriptions, and the optimised method will depend on the application characteristics.

6.9.2 Global Subscription State

At the publisher edge broker, subscription semantics for the multicast has to be identified. The key issues are the number of channels and grouping strategy for multiple channels. Design

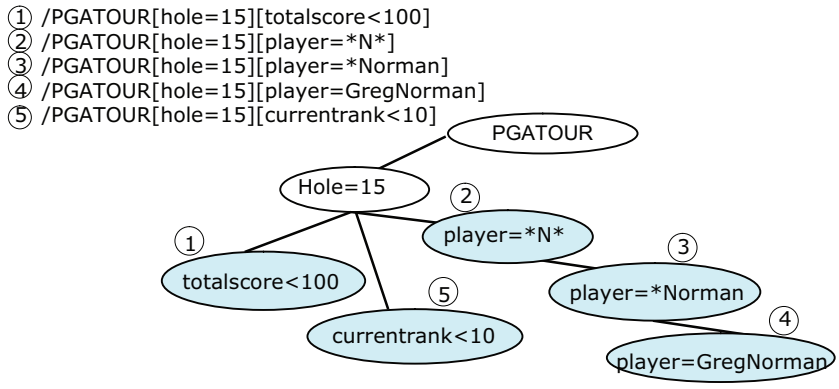


Figure 6.43: Content Subscription Graph for Multicast Group

of a data structure to keep the aggregated subscriptions at the publisher edge broker node, which maintains the state of subscriptions, is also important. The following approaches are experimented:

- Maximum coverage subscription
- Aggregated subscriptions in a graph
- Multidimensional index with RTree
- Clustering subscriptions to create multiple multicast channels

The first three methods define a single channel described below and the fourth method exploits clustering subscriptions (see Section 6.9.3).

Maximum Coverage Subscription

This approach is efficient when subscriptions are client-specific in mobile ad hoc network environments; important alert messages are an example. Thus, a subscriber who subscribes to everything causes flooding and breaks the assumption.

Aggregated subscriptions in a Graph

Aggregated subscriptions are described in a graphical representation, which retains the hierarchical patterns of content coverage among subscriptions. The construction of coverage relations is similar to filter coverage construction in the publish/subscribe systems described in Chapter 5. The graph may result in a single spanning tree when the subscriptions are not hierarchical. Fig. 6.43 shows an example of the content subscription graph. When the publisher edge broker publishes an event, it searches all subscriptions in the graph and, if it matches at least one of them, it publishes the event.

Rectangles in RTree

RTree [Gut84] (see Chapter 4) is a dynamic indexing structure for multi-dimensional data rectangles to decide an aggregated subscriptions. Thus, a new event can be determined to be either matching or not, using the intersection function of RTree.

6.9.3 Clustering Subscriptions

Cluster analysis classifies similar objects into groups where each object within the cluster shares heterogeneous features. The quality of clustering depends on the definition of similarity and the calculation of complexity. Cluster analysis is divided into hierarchical and non-hierarchical

methods. K-means [Mac67] is a heuristic non-hierarchical clustering method, which partitions data into K clusters. K needs to be determined at the onset, and the algorithm is relatively simple. The Fuzzy K-means method [Bez74] uses fuzzy classification. AutoClass [CS95] automatically determines the number of clusters and classifies the data stochastically. COBWEB [Fis87] and CLASSIT [GLF89] are known for clustering periodical data.

In hierarchical methods, the final number of clusters is not known. Two types of hierarchical clustering exist: *agglomerative* and *divisive*. Agglomerative clustering initially places an object in its own group, and combining the two nearest groups results in a hierarchy. Divisive clustering, on the other hand, initially places all objects into a single group. Two objects that are farthest apart within the group are then chosen as seed points for splitting into two groups. The other objects are classified into the new groups by their distance from the seeds. This operation continues until a threshold distance is reached. Hierarchical clustering may produce relatively good results, but it requires a maximum of $O(N^2)$ calculation time, and established clusters are not easily re-classified. Thus, hierarchical clustering methods may not be a good choice when incremental classification is expected.

Grouping by K-means Clustering

This approach is to set the effective groups, including split/combined groups, as a result of the K-means operation. K-means clustering generates a specific number of disjoint, flat (non-hierarchical) clusters, and the algorithm is used for similarity-based grouping. Let G be a set of objects representing subscriptions as an input to the K-means algorithm where the output is:

$$G_1, G_2, ..., G_k; G_i \cap G_j = \emptyset; \text{ (where G: Set of objects, and k: Number of clusters)}$$

The ultimate goal is to divide the objects into K clusters, where some metric relative to the centroids of the cluster is minimised. The distance to the centroids to minimise varies:

- Maximum distance to its centroid for any object
- Sum of average distance to centroids over all clusters
- Total distance between all objects and their centroids.

A total distance minimisation mechanism is used. The euclidean distance is used for distance calculation between object x and y (e.g., a centroid and an object):

$$d(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_j)^2}$$

The sum of squares criterion for: $G_1, G_2, ..., G_k$ is:

$$c(G_i) = \sum_{r=1}^{|G_i|}\sum_{s=1}^{|G_i|}(d(x_r^i, y_s^i))^2$$

Thus, the goal for minimisation is:

$$\sum_{i=1}^{k} c(G_i)$$

The metric to minimise and the choice of distance measure will determine the shape of the optimum clusters. The operation of K-means is:

```
PGATOUR Subscriptions:
{0.72, 0.72, 0,     1,     0,1,          0,  0.7, 0,1}
{0.72, 0.72, 0.56,  0.60,  0,1,          0,  1,   0,1}
{0.72, 0.72, 0.584, 0.586, 0,1,          0,  1,   0,1}
{0.72, 0.72, 0.584, 0.586, 0.309,0.310, 0,  1,   0,1}
{0.72, 0.72, 0,     1,     0,1,          0,  1,   0,0.1}
-------------------------------------------------------------------------------------
K=2 Iteration=1:
Cluster 1 Mean=[0.72:0.72:0.0:1.0:0.0:1.0:0.0:0.85:0.0:0.55]
                items: (1) data 0.72, 0.72, 0.0, 1.0, 0.0, 1.0, 0.0, 0.7, 0.0, 1.0
                       (2) data 0.72, 0.72, 0.0, 1.0, 0.0, 1.0, 0.0, 1.0, 0.0, 0.1
Cluster 2 Mean=[0.72:0.72:0.576:0.59:0.103:0.77:0.0:1.0:0.0:1.0]
                items: (1) data 0.72, 0.72, 0.56, 0.6, 0.0, 1.0, 0.0, 1.0, 0.0, 1.0
                       (2) data 0.72, 0.72, 0.584, 0.586, 0.0, 1.0, 0.0, 1.0, 0.0, 1.0
                       (3) data 0.72, 0.72, 0.584, 0.586, 0.309, 0.31, 0.0, 1.0, 0.0, 1.0

K=3 Iteration=1:
Cluster 1 Mean=[0.72:0.72:0.0:1.0:0.0:1.0:0.0:0.85:0.0:0.55]
                items: (1) data 0.72, 0.72, 0.0, 1.0, 0.0, 1.0, 0.0, 0.7, 0.0, 1.0
                       (2) data 0.72, 0.72, 0.0, 1.0, 0.0, 1.0, 0.0, 1.0, 0.0, 0.1
Cluster 2 Mean=[0.72:0.72:0.56:0.6:0.0:1.0:0.0:1.0:0.0:1.0]
                items: (1) data 0.72, 0.72, 0.56, 0.6, 0.0, 1.0, 0.0, 1.0, 0.0, 1.0
Cluster 3 Mean=[0.72:0.72:0.584:0.586:0.1545:0.655:0.0:1.0:0.0:1.0]
                items: (1) data 0.72, 0.72, 0.584, 0.586, 0.0, 1.0, 0.0, 1.0, 0.0, 1.0
                       (2) data 0.72, 0.72, 0.584, 0.586, 0.309, 0.31, 0.0, 1.0, 0.0, 1.0

K=4 Iteration=1:
Cluster 1 Mean=[0.72:0.72:0.0:1.0:0.0:1.0:0.0:0.85:0.0:0.55]
                items: (1) data 0.72, 0.72, 0.0, 1.0, 0.0, 1.0, 0.0, 0.7, 0.0, 1.0
                       (2) data 0.72, 0.72, 0.0, 1.0, 0.0, 1.0, 0.0, 1.0, 0.0, 0.1
Cluster 2 Mean=[0.72:0.72:0.56:0.6:0.0:1.0:0.0:1.0:0.0:1.0]
                items: (1) data 0.72, 0.72, 0.56, 0.6, 0.0, 1.0, 0.0, 1.0, 0.0, 1.0
Cluster 3 Mean=[0.72:0.72:0.584:0.586:0.0:1.0:0.0:1.0:0.0:1.0]
                items: (1) data 0.72, 0.72, 0.584, 0.586, 0.0, 1.0, 0.0, 1.0, 0.0, 1.0
Cluster 4 Mean=[0.72:0.72:0.584:0.586:0.309:0.31:0.0:1.0:0.0:1.0]
                items: (1) data 0.72, 0.72, 0.584, 0.586, 0.309, 0.31, 0.0, 1.0, 0.0, 1.0
```

Figure 6.44: K-means Clustering: Examples

1. Place K points representing the initial group centroids among subscriptions.
2. Assign each subscription to the group containing the shortest distance to the centroid.
3. When all subscriptions have been assigned, recalculate the positions of the K centroids. The centroid is calculated as the average of all objects in the cluster.
4. Repeat Steps 2 and 3 until the distance of the previous centroid and the new centroid is less than 0.01%. This results in a separation of the subscriptions into groups, from which the metric to be minimised can be calculated.

The K-means method is relatively fast, with the calculation time being O(tkn) (n: number of objects, k: number of clusters, and t: number of repetitions) and normally $t << n$. The weight can be added for associated data values.

In ECCO-PPS, the propagated subscriptions are in the form of Bloom Filters and associated values. They could be summarised when the subscriber edge broker has more than one subscriber connected. When a string value is associated with the subscription, the value itself is hashed, resulting in loss of the locality of the original string value. Thus, the dynamic channelization can be realised without string-based attributes, or an approach not using Bloom Filters (e.g., using RTree for subscription propagation, see Section 6.9.6 for more detail).

The input of K-means is n-dimensional points: $D = x_1, x_2, ..., x_n$, and $K =$ the desired number of clusters. Fig. 6.44 and Fig. 6.45 show a simple example of the subscriptions shown in Fig. 6.43

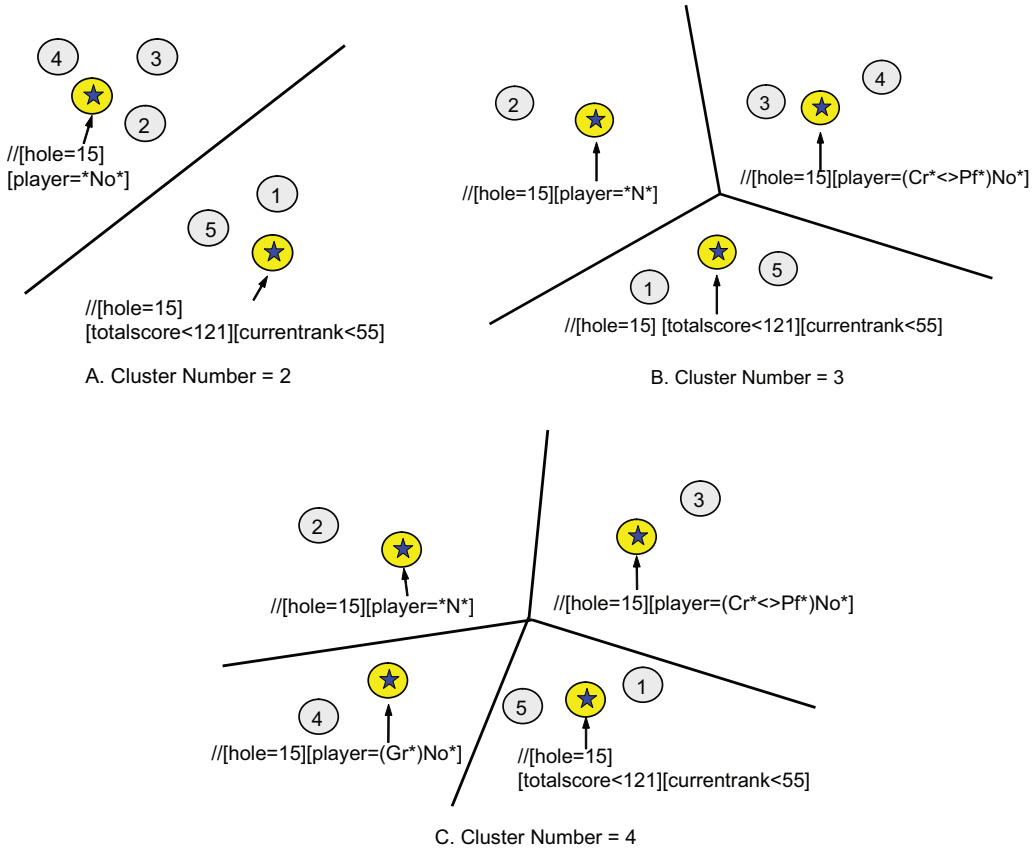


Figure 6.45: Grouping by the K-means Algorithm

divided into different numbers of groups. In this example, the attributes *hole, totalscore, currentrank* and *player* are used. The attribute *player* is split to *lastname* and *first name*. In total 5 attributes are used, and for expressing the range, each attribute represents the lower and upper band, so that the data vector feature for K-means input is 10-dimensional. Fig. 6.44 shows the input and output of the K-means clustering process. *Mean* indicates the centroid of each cluster, which is approximated in the description of subscriptions in Fig. 6.45. This is a simple example, and it has only one iteration of the centroid calculation.

Selection of K is difficult and the challenge is selection of the number of clusters (i.e., K) and measurement criteria of distance from the centre of gravity of cluster. Currently the value K is computed from the number of elements and attributes in a cluster.

K-means Comparison with Content Graph

Fig. 6.46 shows algorithmic comparisons of dynamic channelization from the experiments. The *Graph* method keeps a single group (i.e., one channel), while the K-means method creates more than one group. In this experiment, the subscriptions are randomly generated. The number of groups is influenced by the dimensions of the attributes of subscriptions and the distribution of subscription trends. This issue requires further experiments with real world data. The number of iterations is also shown, depicting the influence of the dimensions to be used for clustering.

K-means creates multiple channels, while the *Graph* approach keeps a single channel. The traffic over the networks will intuitively be reduced. K-means is usually not suitable for non-convex clusters, making it difficult to generalise the clustering algorithms. However, a purpose of us-

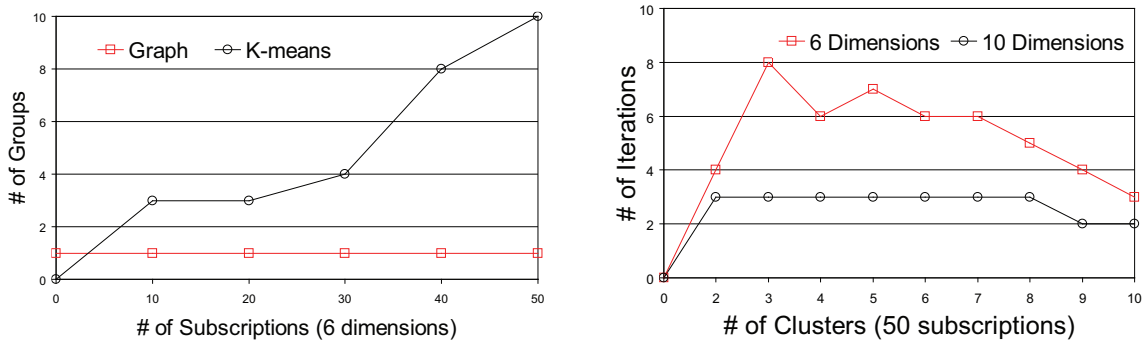


Figure 6.46: Cluster Algorithms Comparisons and Dimensions Impact in K-means

ing the K-means algorithm is to exclude subscriptions that are distant from the majority of subscriptions. This algorithm performs well for highly heterogeneous subscriptions.

A basic problem of K-means is that the number of groups has to be known. K-means algorithms converge after a number of iterations, and normally they converge quickly. The processing can be stopped after any iteration so that feasible partitions into K groups can be adapted for given environments. This provides a way to accommodate changes of grouping, by simply running a number of re-balancing iterations when new subscribers arrive or subscriptions change.

6.9.4 Hierarchical vs. Flat events vs. Mixed

Fig. 6.47 shows a comparison of subscription trends among non-hierarchical, hierarchical, and mixed instances. Error bars on the line of network traffic are too small and therefore not shown. The mixed trend indicates that clustering is not efficient, leading to more network traffic. The non-hierarchical trend shows a stronger effect on clustering than does the hierarchical trend.

The experiments are performed using a specific program written for this purpose. The number of nodes is 100, the number of publishers is 20, and the number of subscribers is up to 50. The average hop counts from the publisher edge broker to the subscriber edge broker is 3, and 3 nodes participate in a forwarding group per publisher edge broker.

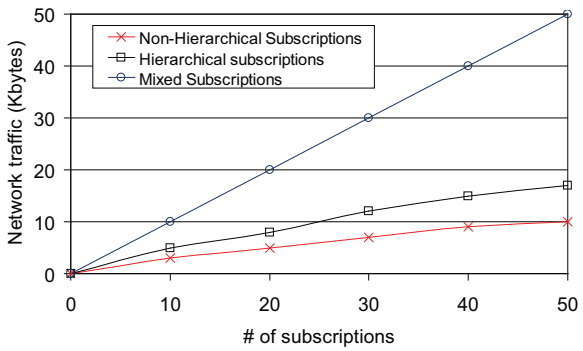


Figure 6.47: Subscription Trend on K-means Clustering

6.9.5 Channel Maintenance

The publisher edge broker is responsible for creating and maintaining multicast channels, including reorganising the multicast channels according to subscription patterns. The reorganising operations are triggered:

- When a subscriber leaves: no JT message arrival to the publisher edge broker.
- When a subscriber joins: JT message arrival to the publisher edge broker.
- When a subscriber changes subscription: this stops the old JT message arrival and starts a new JT message arrival to the publisher edge broker.

The reorganisation process includes evaluation of subscriptions (e.g., clustering) and split/merge channels. Any configuration parameters (e.g., cost factors) can be used as an input for the reorganisation of channels. This operation involves additional traffic from control packets over the networks. However, it will be necessary to provide efficient group membership and selective event dissemination. Currently, multicast channel maintenance only deals with the publisher edge broker. As there could be two identical channels over the network, it could end up constructing two identical separated routing structures. Optimisation can be done using rendezvous nodes by construction of another overlay (e.g., super-peers).

Split and Merge Channels

New subscriptions or publications will change the multicast channel setting, which causes channel split or merging. Determination of the number of clusters for K-means is difficult, and the algorithm used above has to be modified depending on the characteristics of subscriptions. Autoclass [CS95] determines K stochastically and such an approach has to be integrated for determination of K.

The multicast group reorganisation cost can be assumed to be proportional to the number n of multicast groups involved in the reconstruction. In the K-means approach the number of clusters is fixed as K, and the cost of reconfiguration of the multicast group is therefore constant.

6.9.6 Subscriptions in an RTree

The subscription information belonging to the channel can be kept in an RTree at the publisher edge broker node, where each subscription's information and centroid information are kept. RTree conserves the range queries within the data structure. Modelling each subscription as a rectangle in a d-dimensional space can maintain subscription information efficiently, where subscription attributes correspond to the dimensions of the RTree.

The clustering-based approach associates a multicast group with each cluster, and then the nodes in the multicast group are organised in the multicast tree, where the centroid is the root. The created tree maintains the events and the subscriptions within the cluster. The corresponding multicast group can be identified by the intersection operation among the hyper-rectangle in the events space. RTree also provides an integrated approach with the type definition. Expressing a type as a dimension of information in RTree may be simpler than defining separated event types.

6.9.7 Use of Super-Peers

The use of super-peers to maintain the whole system may improve subscription management and system performance. For example, when a specific subscription has a significant distance from the common subscriptions, it may create a direct channel to a super-peer, which is the nearest

subscriber edge broker to the subscriber. Super-peers can create an overlay to manage channels in cost effective ways.

Partitioning in the ESP approach can be good for geographical super-peer environments, where each region can have a dedicated super-peer to be responsible for multicast group maintenance. In this case, publications are forwarded to the appropriate super-peer brokers, before that broker will perform event dissemination. This approach contains the same mechanism as CAN multicast.

6.10 Sample Application

A prototype application *Golf Tour* is developed, which enables a real-time content-based publish/subscribe system among the gallery during a golf tournament (Fig. 6.8). Initially, the gallery obtains the software including the players' list. Subscriptions based on defined categories can be registered at anytime. Categories include *players*, *holes*, *scores*, etc., and combinations of attributes can be chosen. In Fig. 6.8, the gallery moves along with the players using smart phones or PDAs obtaining other players' information. During the game, a participant may publish messages or change subscriptions. Gallery G1, G2, and G3 have similar subscriptions and

```
<?xml version="1.0" encoding="UTF-8"?>
<PGATOUR id="001" timestamp="1003-02-27T12:00:00.000-00:00"
        xmlns="http://www.cl.cam.ac.uk/~ey204/lib/">
    <player>woods</player>
    <hole>15</hole>
    <holescore>5</holescore>
    <totalscore>52</totalscore>
    <currentrank>2</currentrank>
    <club>1,4,7,P</club>
    <put>(2489,1087)(3456,987)</put>
    <grass>
        <humidity>68%</humidity>
        <cut>medium</cut>
    <wind>NNW,0.5</wind>
    <temperature>19C</temperature>
</PGATOUR>
```

Figure 6.48: Event Instance Examples

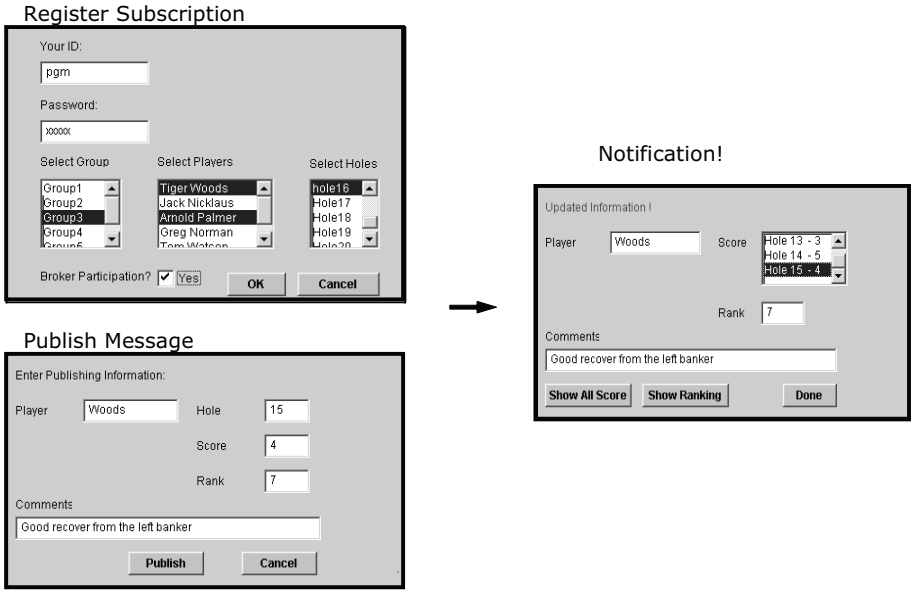


Figure 6.49: Golf Tour Client API

matching messages may be delivered on the same multicast channel depending on the topology of publishers and timing of publishing messages. Gallery nodes accordingly create a forwarding mesh. The proposed approach constructs dynamic channels from subscriptions and optimises the traffic, while maintaining changes in subscriptions and movement of devices. After the game, this mechanism may be used for taxi sharing. The access point based 802.11*b* communication range is around 200 metres, and ad hoc communication can extend the communication capability to cover the whole golf course. There would be other solutions such as use of satellite or radio, but ECCO-PPS does not require any infrastructure.

Fig. 6.49 shows the user interface of the subscription registration, and message publishing. When a message is published, the message forms an XML data shown in Fig. 6.48. The communication between brokers and clients is implemented with unicast. Node *O* indicates official announcements at each hole, which regularly publishes messages. When a message is published, ECCO-PPS establishes the routing table. Mobile peer-to-peer communication can extend the communication capability to cover an entire golf course. No experiment or performance evaluation was operated using this application.

6.11 Summary and Outlook

In this chapter, I have investigated structureless asynchronous group communication over wireless ad hoc networks. Using flooding for constructing routes reactively in mobile ad hoc networks has the advantage that no prior assumption of the network topology is required. Thus, flooding is an attractive property in dynamic mobile ad hoc networks. To optimise flooding traffic, reducing the overhead by providing some means to direct the messages towards the target destination is a powerful solution.

I present ECCO-PPS, which uses context adaptive controlled flooding, taking a cross layer approach between middleware and the network layer, and providing a content-based publish/subscribe paradigm. Application data are influential to data dissemination in ubiquitous computing scenarios. The state information of the local node may be the event forwarding trigger. Key characteristics are that publish/subscribe is becoming more symmetric, and the destination of events depend on the rules and conditions defined by the sender or event itself. The publish/subscribe system offers a data centric approach, where the destination address is not described with any explicit network address. The symmetric publish/subscribe model adds a new level of data centric paradigm, and this leads to a fundamental change in functionality, at the network level, of asynchronous communication and membership maintenance.

To optimise system throughput, an imprecise summary is used, which provides an important tradeoff between routing overhead and false positive traffic. Considering summaries for subscriptions requires careful designing of the expressiveness of subscriptions, and needs to consider the whole event space and dimensions of subscriptions. Thus, a conclusion of this work is the requirement to a lightweight RTree for content-based publish/subscribe in MANETs so that the semantics of each subscription can be well-propagated through the network. Overall, the context exchange interface provided by ECCO-PPS allows both middleware and network layer components to exploit efficient and dynamic event routing mechanisms for better performance. The experiments show conclusive evidence to prove the above capabilities.

I attempted *Dynamic Channelisation* using subscription clustering that is work in progress, with scope for future work. The use of super-peers will be important, where resource-rich peers can create another overlay network to help channelisation, storage, and other functionalities in publish/subscribe systems.

7

Event Correlation

To add an additional dimension of data processing in global computing, it is important to understand event aggregation, filtering and correlation. Event correlation services may be part of either applications or event notification services. The emergence of WSNs brings yet more complex issues to event correlation. Although composite events have been a useful modelling tool in active database research and telecommunications network monitoring, little progress has been made in event-based middleware. The semantics of operators for composite events is not defined in a uniform manner in existing middleware and applications, leading to a number of problems. Event consumption rules are mostly handled as part of an implementation without a clear semantic definition. Formal mechanisms to define complex temporal and spatial relationships among correlated events are rarely found. Unambiguous semantics over heterogeneous network environments is highly desirable.

In this chapter, I first discuss the characteristics of composite events and queries and report a comparative study of existing event correlation work. Then, I introduce novel generic composite event semantics, with interval-based semantics, for event detection. This precisely defines complex timing constraints among correlated event instances. An ultimate goal is to extend the functionality of simple publish/subscribe filters to enable stateful subscriptions including parametrisation as part of the publish/subscribe functionality.

7.1 Filtering, Correlation, and Aggregation

Ubiquitous computing, with a dramatic increase of event monitoring capabilities from wireless devices and sensors, requires more complex temporal and spatial resolution in event correlation. An example of such a subscription is *Notify the products in the store that become on sale with 30% discount price within 50 minutes after the total number of customers in the store reaches 120*. This subscription may not be described by filtering, where an exact temporal constraint (e.g., time instant, time interval) is expected, while a time interval is relative to a time point (e.g., 50 minutes after the number of customers reaches 120) in the above example. Thus, the semantics of filtering and correlation need to be defined together.

Publish/subscribe systems may offer content-based filtering, which allows subscribers to declare

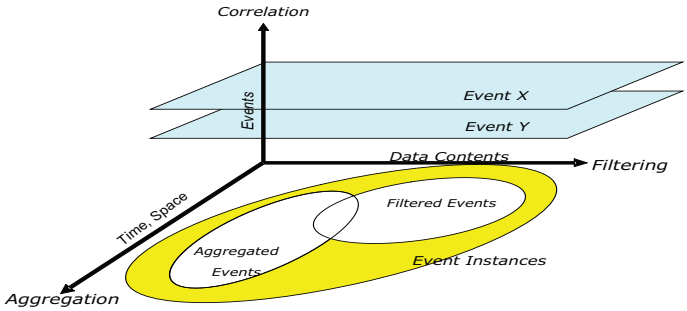


Figure 7.1: Filtering, Correlation and Aggregation

their interests in the attribute values of events via a flexible subscription language. A subscription can apply to different event types but the aim is to select individual events. On the other hand, event correlation addresses the relationship among, or patterns of, instances of different event types. Filtering and correlation share many properties. I have discussed filters on content-based subscription in Chapter 4. It is complex to include spatial and temporal attributes, if subscriptions need to define correlations of multiple event instances (i.e., composite events). Currently, there is no clear definition of required tasks for filtering and correlation processes in most publish/subscribe systems.

WSNs raise new issues to be addressed such as data aggregation and in-network operation to deal with redundancy, summarisation, and quality control of data. For example, some events last longer than the sensing interval, which causes repetitive event reports. If the size of the detection group is too small, the same events will be reported by several spatial groups. Traditional networking research has approached data aggregation as an application-specific technique that can be used to reduce network traffic. In WSNs, the main issue on aggregation is to summarise current sensor values in some or all of a sensor network. The group of nodes often acts as a single unit of processors. For a single phenomenon, multiple events may be produced to avoid the loss of event instances, which is a totally different concept from the traditional duplication of events. Normally aggregation of sensor data includes three stages: local, neighbour, and global.

TinyDB [MFH+02] is a query processing system for sensor networks and takes a data centric approach. Each node keeps the data and executes retrieval and aggregation, with on-demand based operations to deliver the data to external applications. The declarative SQL-like query interface is executed for data retrieval and aggregation. TinyLIME [CGG+05] enhances LIME (Linda In Mobile Environments) to operate on TinyOS. In TinyLIME, LIME is maintained on each sensor node together with a partition of a tuple space. A coordinated tuple space is created across the nodes, connecting with the base station in one hop. TinyLIME works as middleware by offering this abstracted interface to the application. It does not currently provide any data aggregation function, only a data filtering function based on Linda/LIME at the base station node. On the other hand, TinyDB supports data aggregation via SQL query, but redundancy/duplication handling is not well defined.

This new paradigm requires a composition of events in heterogeneous network environments, with varying network conditions. Event Correlation is a multi-step operation from event sources to the final subscribers, combining information collected by wireless devices into more specific information or target subscriptions. When real-time (meaning event occurrence time) constraints involve event correlation, semantic ambiguity may arise from the timing of multiple instances of the same event, which may be created due to the nature of unstable network environments. For example, original timestamps of the same events from sensor nodes may be transformed to

interval-based timestamps at the gateway node (see Section 7.7.5), and these timestamps may vary even if they are transformed from the same event. Dealing with these event duplication requires a clear semantics for the selection and consumption policy. A unified semantics needs to be defined to resolve this ambiguity. Temporal ordering in real-time is a critical aspect of event correlation in wireless ad hoc network environments (e.g., object tracking). Neither logical time nor classical physical clock synchronisation algorithms may be suitable. Events can be triggered by physical phenomena such as glaciers and earthquakes, and the order in which data is sensed is important for causality.

Fig. 7.1 shows the relationships between aggregation, filtering and correlation. Middleware research on WSNs has recently been active, but most work focuses on in-network operation for specific applications. Event matching algorithms are becoming more complex, to deal with multidimensional data and group queries. I take a global view of event correlation in distributed systems with unified correlation semantics.

7.2 Correlation Definition Language

Traditional publish/subscribe systems are scalable, but the subscription language is often a simple subset of SQL, which is not sufficiently expressive. In contrast, data stream management systems provide an expressive language (e.g., CQL [ABW02] [MWA+03]) but with only little work on scalability in distributed environments. Another contradicting and complex issue is to balance the expressiveness of event/query with the quality and accuracy of events. The applications that process events could range from medical applications to shopping guidance. An exact event detection mechanism with crucial real-time support may be required for some but not all applications.

Correlation and aggregation are mostly handled externally, and little work has been done on multi-event type optimisation for expressiveness. Time-based correlation can help to establish causality such as tracing a connectivity problem among different incidents. Some problems can only be determined through such temporal correlation. Thus, it is important to decide whether to define either a combined language for both subscription of individual event types and composition of subscriptions, or two layers of language.

Many expressive languages have been introduced in the Active Database community [WC96] for complex event-condition-action (ECA) rules. The composite event definition languages of Snoop [CM96] and ODE [GJS92] describe composite events in regular expressions using a non-deterministic FSA model. The FSA construction of [GJS92] supports a limited parameterisation for composite events (e.g., equality constraints between attributes). On the other hand, the more expressive event languages [GA02][ZU99] do not provide clear semantics. Thus, it is not easy to compare the expressiveness of languages.

From a survey on formal methods in event specification and analysis [BD02], most of the popular approaches are based on theoretical models such as finite state machines (FSM), timed automata, process algebra, and Petri-Nets. [DOT+96] discusses the difficulty of using FSM to deal with hierarchical modelling and synchronisation. FSMs can be augmented to Timed Automata by incorporating a finite set of real-valued clocks. FSM and Timed Automata contain deterministic models.

A number of approaches exist that are based on event/process algebra and composition logic. The composition of events can be described in ECA rules based on defined event algebra. [LCB99] [YC99] propose composition algebra for distributed system environments. The event algebra developed in [Hay96] [Pro05] [HB02] provides time restricted operations.

Petri-Nets have advantages for describing events including Coloured-Petri-Nets [Jen81], Timed-Petri-Nets and Stochastic Petri-Nets [DiC93] [TC95]. For example, they provided non-determinism, and spatial properties can be incorporated into the token, extending Coloured-Petri-Nets.

The semantics of composite events in event algebras is defined in Section 7.5, so that any other formalisation language can be used to convert to FSM, Petri-Net, etc. Formal semantics will help to determine expressiveness and scalability, and defined event algebras can be formalised such as in the π-calculus. Existing compilers from π-calculus to Java or any other language will automatically provide programming language interfaces to applications.

Fuzzy Subscription: The subscribers may not know exactly what to subscribe to when issuing subscriptions (i.e., composite event subscriptions). For example, if the subscriber is interested in any abnormality in a time series of data, the pattern of the composite event is unknown. This requires a new type of subscription definition (i.e., fuzzy subscription), using a more ambiguous subscription language based on natural language or a series of keywords. In [LJ04], subscriptions are expressed in a formula using fuzzy set to integrate with publish/subscribe systems. The concept of fuzzy subscription is not explored in depth in publish/subscribe systems yet, and will be an interesting future work.

7.3 Event Correlation in Middleware

This section shows a comparative study of existing event correlation mechanisms in middleware. Event correlation may be deployed as part of applications, as event notification services, or as part of a middleware framework. Definition and detection of composite events vary, especially over distributed environments. Equally, named event composition operators do not necessarily have the same semantics, while similar semantics might be expressed using different operators. Moreover, the exact semantic description of these operators is rarely explained. Thus, I define the following schema to classify existing operators: conjunction, disjunction, sequence, concurrency, negation, iteration, selection, aggregation, spatial restriction, and temporal restriction. Considering the analysed systems, it becomes clear that it is not sufficient to simply consider the operators to convey the full semantic meaning. Each system offers parameters, which further define/change the operators' semantics. The problem is that the majority of systems reflect parameters within the implementations. Parameters for consumption mode and duplicate handling are rarely explicitly described. Table 7.1 and Table 7.2 show a comparative study of event composition semantics in ten existing systems and the approach taken in ECCO. The tables give an overview of event-based composition languages typically supported in event-based systems and provide an analysis of the languages from a unified viewpoint. Note that the list of analysed systems cannot be exhaustive but considers a representative set of selected composition semantics. None of the listed systems includes wireless network environments except ECCO. Recent event correlation services for WSNs or embedded systems provide time restricted operations (see [YB05b] for the detail). Most event notification systems still only support primitive events, and their focus is on efficient filter algorithms. Brief explanations of the characteristics of each system are given below.

Table 7.1 shows composition operators, while Table 7.2 emphasises temporal order related parameters such as consumption modes. Concepts of conjunction, disjunction, sequence, concurrency, negation, and iteration operators are based on event algebra. Selection defines the event instance selection, by which events qualify for a composite event, and how duplicate events are handled. Conjunction and disjunction are supported in most systems, some of which implement sequence operators (requiring ordering), while fewer support negation. Selection and concurrency are

	Operators						
	Conjunction	Disjunction	Sequence	Conc.	Negation	Iteration	Selection
ECCO	$A + B$	$A \mid B$	$A; B$	$A \| B$	$-A$	A^*	A^N
Opera	$A \| B$	$A \mid B$	$A; B or AB$	-	$\neg A$	A^*	-
CEA	$A \& B$	$A \mid B$	$A; B$	-	$-A$	-	-
Schwiderski	A, B	$A \mid B$	$A; B$	$A \| B$	$NOT A$	$A^* or A^+$	-
A-mediAS	$A \& B$	$A \| B$	$A; B$	-	$-A$	-	$A^{[i]}$
Ready	$A \&\& B$	$A \| B$	$A; B$	-	$not A$	-	-
Eve	$CON(A, B)$	$DEX(A, B)$	$SEQ(A, B)$	$CCR(A, B)$	$NEG(A, B)$	$REP(A, n)$	-
GEM	$A \& B$	$A \| B$	$A; B$	-	$!A$	-	-
Snoop	A, B	$A \vee B$	$A; B$	-	-	A^*	-
Rebeca	$A \wedge B$	$A \vee B$			$\neg A$	A^*	-
SAMOS	A, B	$A \mid B$	$A; B$	-	$NOT A$	$TIMES(n, A)$	$A^*/last(A)$

Table 7.1: Event Composition Semantics - Composition Operators

	Time Operator		Timestamp	Composition	T.C.	Consumption			Subset	Detection
	Period	Life				Unrest.	Recent	Chron.		
ECCO	$(A; B)_T, (A + B)_T$	$(A)_T$	P, PI, I	I	×	×	×	×	×	algorithm
Opera	$(A, B)_T$	-	PI	P	-	-	-	-	-	T-FSA
CEA	-	-	P	P	-	×	-	-	×	FSA
Schwiderski	-	-	P	P	-	-	-	-	-	Rule
A-mediAS	$(A, B)_T, (A; B)_T, -A_T$	-	P	P	-	×	×	×	×	T-FSA
Ready	-	-	P	P	-	-	-	-	×	-
Eve	-	-	P	P	-	-	-	×	×	Graph
GEM	(A+timeperiod)	-	P	P	×	-	×	-	×	Rule
Snoop	$(A, [tiestring] : param, B)$	-	P	P	-	-	×	×	×	Graph
Rebeca	slide window	-	PI	P	-	-	×	×	-	Rule
SAMOS	-	-	-	-	-	-	-	×	×	PetriNet

Time Operator: Period (between event A and B) Life (valid time for event A)

Timestamp: P (Point-based), PI (Point represented in Interval-based format), and I (Interval-based).

Composition: Event composition semantics. P (Point-based), and I (Interval-based).

Temporal Condition (T.C.): Temporal conditions such as *A before B, A meets B, etc. for event composition.*

Consumption: Event consumption (Unrestricted, Recent, and Chronicle).

Subset: Parameters for selection or subset on duplication handling.

Detection: Implementation methods.

Table 7.2: Event Composition Semantics - Time-related Parameters

rarely supported. Concurrency is difficult to determine for distributed systems. Time operators are not always supported, requiring a time handling strategy for distributed systems. Consumption mode and temporal conditions are rarely made explicit. If they are explicit, several options are supported as otherwise they are hard-coded in the system and difficult to determine. The listed systems are notification services, event composition languages, and workflow coordinators, in which common characteristics are fairly complete semantics of event composition.

ECCO: the proposed prototype described in Section 7.5.
Opera: a framework for event composition in a large scale distributed system [PSB04] aiming at reduction of event traffic by distributed composite event detectors. The language of composite events is based on FSA.
CEA: our group's early work on the Cambridge Event Architecture (CEA) extended the then-predominant, object-oriented middleware (CORBA and Java) with an *advertise, subscribe, publish, and notify* paradigm [BBH+95]. COBEA [MB98] is an event-based architecture for the

management of networks using CEA based composite event operators.
Schwiderski: enhanced distributed event ordering and introduced 2g-precedence-based sequence and concurrency operators for detection of composite events [Sch96].
A-mediAS: an integrating event notification service that is adaptable to different applications specifically on handling composite events and event filtering methods [HB02].
Ready: an event notification service from AT&T Research similar to SIENA. Ready supports composite events and its grouping functionality can be shared among clients [GKP99].
EVE: combines characteristics of active databases and event-based architectures to execute event driven workflows [GJS92].
GEM: GEM [MSS97] is an interpreted, generalised, event monitoring, rule-based language.
Snoop: a model-independent event specification language [CM96] supporting parameter contexts. It supports temporal, explicit and composite events for active databases.
Rebeca: an event-based electronic commerce architecture focusing on event filtering in a distributed environment [MFB02]. Temporal delays in event composition have been addressed in [LCB99].
SAMOS: The SAMOS (Swiss Active Mechanism-based Object-oriented Database System) [GD94] project addresses the specification of active behaviour and its internal processing, supporting ECA rules. The detection of composite events is implemented based on coloured Petri-Nets.

7.4 Emergence of WSN Data

High volumes of sensor nodes can congest the network, and data aggregation and fusion are desirable for reducing network traffic. Three approaches for performing aggregation are common. First, diffusion algorithms assume that data are transmitted from one node to the next, thus propagating through the network to the destination. Along the way the data may be aggregated. Normally, only simple aggregation is applied and homogeneous data is expected. Second, SQL extensions for continuous queries operate on streaming queries. Third, event graphs for composite events are deployed, based on event algebra. There have been an effort to extend event algebras with temporal constraints for event correlation for WSNs.

The three approaches above decide when and where aggregation of data in WSNs should be performed. This involves how to deal with time in distributed environments with state, asynchrony and unstable communication, redundancy and so forth. Each aggregation mechanism needs specific resources and affects routing strategy. Query processing can be performed at sensor nodes, sink nodes, resource rich nodes, or a combination of these. Typical aggregation operations include maximum, minimum, average, sum and well-known database operations.

To specify more complex and *higher-level* events, an event-description language (SNEDL) based on Petri-Nets is introduced [JSS05]. This provides a system with asynchronous and distributed features. The language SNEDL can form a hierarchy of events. [LKGH03] supports a distributed index for multidimensional range queries such as *List all events that have temperature values between* 50℃ *and* 60℃ *and light levels between 10 and 20 luces.*

Existing techniques such as SQL-like languages have a number of limitations. They cannot capture complex dependencies and interactions among different events or sensor types. They make it difficult to describe complex temporal constraints and data dependencies, which causes problems to provide a global view or to support event-based systems. A description language that can incorporate knowledge of sensing into event definitions is necessary to support WSNs.

SQL-like Expressions: Fig. 7.2 shows an example. Three types of sensors (e.g., humidity, light, and temperature) are supposed to detect fire, where the fire event is a combination of

```
SQL like Expression:
INSERT INTO EventList Fire (ID, subevenet_set)
       VALUES (0001, Subeventset)
       WHERE Subeventset

       denoted as SubEventset
       = (Humidity, Light, Temperature, Function:
       0.2*humidity + 0.3*light + 0.5*temperature >=1.0)
```

Figure 7.2: Query Expressions: SQL

```
CQL Expression:
SELECT  P.AnimalId, COUNT(*)
        FROM   Pulse P  [Frequency 1 minute : Range 1 minute]
        WHERE  P. Species = "zebra"
        GROUP BY P.AnimalId
```

Figure 7.3: Query Expressions: CQL

positive readings from the sensors. The function to calculate the possibility of fire is described.

The notation in SQL-like Expressions cannot describe a more complex situation in determining a fire event; the reading for humidity is valid for only 15 seconds, light for 1 second, and temperature for 20 seconds. For example, if a sensor detects abnormal light, but within the 1 second interval for light reading there is no positive reading from temperature or sound, then this is not taken as a fire event. This example shows that SQL cannot handle complicated temporal constraints for events.

CQL Expression: In CQL [ABW02], the SQL primitive is extended to incorporate streaming support, where the desired sample rate can be specified. Fig. 7.3 shows an example, where the pulse rate per minute of a zebra is measured. Nevertheless, due to the inherent limitation of SQL, it is not suitable as a formal description language for events in WSNs.

A formal method (i.e., Petri-Net) gives unambiguous event specification and an integration of Coloured-Petri-Nets and Timed-Petri-Nets with CQL provides more complex event definitions. Furthermore Stochastic Petri-Net property makes probabilistic analysis and evaluation possible. This is convenient for a vast number of distributed and individually unreliable sensors.

Peri-Net Expression: The same example described in an SQL expression is depicted as a SNEDL Petri-Net notation [JSS05] in Fig. 7.4. Temporal, spatial and probability features are added. For example, a time guard function is defined for transitions: $gamma : T- > (t1, t2)$, where $t1 <= t2$ are real. This means a transition can only fire during a given time interval. Thus, for light sensing data only a second interval is specified to force a temporal constraint.

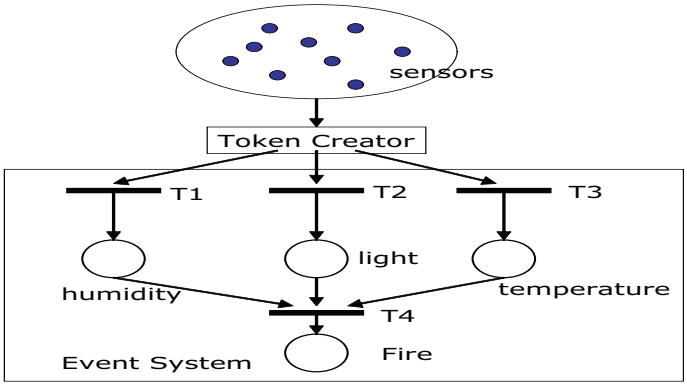


Figure 7.4: Explosion Event in Petri-Net Expression

Another example is that the threshold function can be defined for places. Entering the place *Fire* has a threshold function, where the relations among humidity, light and temperature can be defined.

7.5 Event Correlation Semantics

I define unified composite events by expressions built from primitive and composite events and algebraic operators. Event algebra is an abstract description of event composition independent of the actual composite event definition languages. Parameters including time, selection, consumption policy, subset policy, and precision policy are also supported. Basic operators provide the potential of expressing the required semantics and are capable of restricting expressions by parameters. Defined composite events can be transformed using algebraic laws, into more feasible and implementable forms for resource constrained environments.

Interval-based semantics for event detection is introduced based on [AC03] and extended, defining precisely complex timing constraints among correlated event instances [YB05c]. WSNs produce a high volume of both continuous and discrete data, and both types need to be dealt with efficiently. Timed automata can process time but not space as continuous variables. Also, an interval-semantics supports more sensitive interval relations among events in environments where real-time concerns are more critical such as wireless networks or multi-media systems. The temporal operators introduced in [All83] are not uniformly defined in many applications, and precise timing constraints are defined.

7.5.1 Composite Event Operators

An event algebra is used to define the semantics of composite events. A composite event consists of any primitive or composite events. The algebra has a relatively simple declarative semantics, and it also ensures that detection can be implemented with restricted resources.

Event instances are denoted by e, while event types are denoted by E. An event instance e that belongs to an event type E is denoted as $e \in E$. An event type may have subtypes such as $e \in E \subset E_{super}$, where E_{super} indicates a super-class of E. $NULL$ denotes an event set $\mathbb{NULL} = \phi$ with no event instances.

The occurrence time of the composite event depends on the event composition semantics. t refers to the occurrence time defined, based on the time system; T is a time span in reference time units.

Table 7.3 defines the timing constraints related to event composition operators in the classification of timing semantics defined in [All83].

The event operators are informally defined as follows:

Definition 7.1 (Conjunction A + B) *Events A and B occur in any order. $(A + B)_T$ with a temporal parameter T denotes the maximal length of the interval between the occurrences of A and B. Note that $(A + B)_\infty$ or $(A + B)$ has no temporal restrictions.*

Definition 7.2 (Disjunction A | B) *Event A or B occurs.*

Definition 7.3 (Concatenation A B) *Event A occurs before event B, where timestamp constraints are A* meets *B, A* overlaps *B, A* finishes *B, A* includes *B, and A* starts *B.*

Definition 7.4 (Sequence A ; B) *Event A occurs before B, where timestamp constraints are A* before *B, and A* meets *B. $(A; B)$ ensures events are disjoint, whereas (AB) allows overlaps between events. $(A; B)_0$ is a special case of A* meets *B.*

Examples:

- $(A; NULL; B)$*: denotes no occurrence of any event between event* A *and* B*.*
- $(A;B)_T$*: means an interval* T *between event* A *and* B*.*
- $(A; B)_0$*: denotes that event* A *and event* B *occur without any time-interval.*
- $(A; NULL; B)_0$*: denotes that event* A *and event* B *occur contiguously without any other event occurrence at the meeting time.*

Definition 7.5 (Concurrency A‖B) *Event* A *and* B *occur in parallel.*

Definition 7.6 (Iteration A*) *Any number of event* A *occurrences.*

Examples:

- $A(A|B)^*C$*: would match input such as* AAC *or* $AABAC$*.*

Definition 7.7 (Negation $-\mathbf{A}_T$) *No event* A *occurs for an interval* T*.*

Examples:

- $(A - B)$*: denotes no* B *starts during* A*'s occurrence.*
- $(A - B)_T$*: denotes no* B *starts after starting* A*'s occurrence within an interval* T*.*
- $(A; B) - C$*: denotes that event* A *is followed by* B *and there is no* C *in the duration of* $(A; B)$*.*

Definition 7.8 (Selection $\mathbf{A}_T^N$) *The selection* A^N *defines the occurrence defined by the operation* N *within an interval* T*, where* $N \in ALL, n, FIRST, LAST, ...$

Examples:

- A_T^{ALL}*: denotes taking all the event instances during an interval* T*.*
- A_T^n*: denotes taking the nth instance during an interval* T*.*
- A_T^{FIRST}*: denotes taking the oldest instance during an interval* T*.*
- A_T^{LAST}*: denotes taking the most recent instance during an interval* T*.*

Definition 7.9 (Aggregation $\mathbf{A}_T^G$) *The aggregation* A^G *defines the operation of aggregation* G *within an interval* T*, where* $G \in AVG, MAX, MIN, ...$ *The attribute to be used for the aggregation operation is assumed to be implicitly identified.*

Examples:

- A_T^{AVG}*: denotes taking the average during an interval* T*.*
- A_T^{MAX}*: denotes taking the maximum during an interval* T*.*
- A_T^{MIN}*: denotes taking the minimum during an interval* T*.*

Definition 7.10 (Spatial Restriction $\mathbf{A}_S$) *Event* A *occurs within a spatial restriction defined by* S *such as location identifiers or GPS coordinate values.*

Examples:

- A_{CB30FD}*: The area code* $CB30FD$ *identifies the zone around the Computer Laboratory in Cambridge. Event* A *is valid only when this spatial condition is satisfied.*

	Relation	Timestamps of Primitive Events	Point	Interval	Interval/Point	Point/Interval
1	A before B	P-P: $t_p(A) < t_p(B)$	○ A	○–A–○	○–A–○	○ A
		I-I: $t_i(A)^h < t_i(B)^l$	○ B	○–B–○	○ B	○—B–○
	(A + B)	I-P: $t_i(A)^h < t_p(B)$	●–●	●——— ●–	●——— ●–	●——— ●–
	(A | B)	P-I: $t_p(A) < t_i(B)^l$	●	●—— ●	●—— ●	●
	(A ; B)		●–●	●——— ●–	●——— ●–	●——— ●–
2	A meets B *	P-P: NA		○—A–○	○——A——○	○ A
		I-I: $t_i(A)^h = t_i(B)^l$		○—B—○	○ B	○——B——○
	(A + B)	I-P: $t_i(A)^h = t_p(B)$		●——— ●–	●——— ●–	●——— ●–
	(A | B)	P-I: $t_p(A) = t_i(B)^l$		●——- ●–	●——— ●	●
	(A B)			●——— ●–	●——— ●–	●——— ●–
	$(A ; B)_0$			●——— ●–	●——— ●–	●——— ●–
3	A overlaps B	P-P: NA		○—-A—-○		
		I-I: $(t_i(A)^l < t_i(B)^l) \wedge (t_i(A)^h > t_i(B)^l)$		○—-B—○		
	(A + B)	I-P: NA		●——— ●–		
	(A | B)	P-I: NA		●——— ●		
	(A B)			●——— ●–		
4	A finishes B	P-P: NA		○———A———○	○——A——○	○A
		I-I: $(t_i(A)^l < t_i(B)^l) \wedge (t_i(A)^h = t_i(B)^h)$		○—B—○	○ B	○——B——○
	(A + B)	I-P: $t_i(A)^h = t_p(B)$		●——— ●–	●——— ●–	●——— ●–
	(A | B)	P-I: $t_p(A) = t_i(B)^h)$		●——— ●–	●——— ●–	●
	(A B)			●——— ●–	●——— ●–	
5	A includes B	P-P: NA		○———A———○	○——A——○	
		I-I: $(t_i(A)^l < t_i(B)^l) \wedge (t_i(A)^h > t_i(B)^h)$		○–B–○	○ B	
	(A + B)	I-P: $(t_i(A)^l < t_p(B)) \wedge (t_i(A)^h > t_p(B))$		●——— ●–	●——— ●–	
	(A | B)	P-I: NA		●——— ●–	●——— ●–	
	(A B)			●——— ●–	●——— ●–	
6	A starts B	P-P: NA		○—A—○	○——A——○	○ A
		I-I: $(t_i(A)^l = t_i(B)^l) \wedge (t_i(A)^h < t_i(B)^h)$		○———B———○	○ B	○——B——○
	(A + B)	I-P: $t_i(A)^l = t_p(B)$		●——— ●–	●——— ●–	●——— ●–
	(A | B)	P-I: $t_p(A) = t_i(B)^l$		●——— ●	●	●
	(A B)			●——— ●–		●——— ●–
7	A equals B	P-P: $t_p(A) = t_p(B)$	○ A	○———A———-○		
		I-I: $(t_i(A)^l = t_i(B)^l) \wedge (t_i(A)^h = t_i(B)^h)$	○ B	○———B———-○		
	(A + B)	I-P: NA	●	●——— ●–		
	(A | B)	P-I: NA	●	●——— ●–		
	(A || B)		●	●——— ●–		

●—● depicts the timestamp for the composite events
* A meets B where $t(A)^h$ and $t(B)^l$ share the same time unit

$t(A)$: timestamp of an event instance A
$t_p(A)$: Point-based timestamp
$t_i(A)_l^h$: Interval-based timestamp from event composition
$t_{pi}(A)_l^h$: Point-interval-based timestamp
(compared as point-based timestamp but using interval comparison)

P-P: Between Point-based and Point-based timestamps
I-I: Between Interval-based and Interval-based timestamps
I-P: Between Interval-based and Point-based timestamps
P-I: Between Point-based and Interval-based timestamps

Real-time Period T:

$$[t(B)^l, t(B)^h] - [t(A)^l, t(A)^h] < T = \begin{cases} YES: & max(t(A)^h, t(B)^h) - min(t(A)^l, t(B)^l) \leq T(1-\rho) \\ NO: & max(t(A)^l, t(B)^l) - min(t(A)^h, t(B)^h) < T(1+\rho) \\ MAYBE: & otherwise \end{cases}$$

where ρ is maximum clock skew.

Table 7.3: Interval Semantics - Timestamps for Composite Events

Definition 7.11 (Temporal Restriction A_T) *Event A occurs within T.*

Examples:

- $(A;B)_T$*: B occurs within an interval T after A.*
- B_T*: B is valid for an interval T.*

For event expression A and B, $A \equiv B$ can be established using the laws of algebra. This can be used for adapting the expression for resource constrained environments. For example, $A;(B|C) \equiv (A;B)|(A;C)$ could be applied when the event rate C is low or $(A-B)-C \equiv A-(B|C)$ to avoid the negation operation as much as possible.

The following examples illustrate the use of the operators to describe composite events.

Example 1: The temperature of rooms with windows facing south is measured every minute and transmitted to a computer placed in the corridor. T denotes a temperature event and T_{30}^{AVG} denotes a composite event of the average temperature during 30 minutes. $(T_{room1}+T_{room7})_{30}^{AVG}$ denotes to take the average of room 1 and 7.

Example 2: Two sensing receivers are placed before and after a stop sign on the street. When car passes, the sensors generate events to the local computer. Suppose the event received before the stop sign is B and after the stop sign is A for a given car. $(B;A)_2$ denotes A occurs 2 seconds after B, and indicates a car did not make a full stop at the stop sign. On the other hand, $((B;A)_{60})^*$ indicates cars may not be flowing in the street, which indicates potential traffic congestion.

Example 3: At a highway entrance, a sensor detects movement of a passing car as event E. The number of cars entering the highway $(E_{10}^{SUM})_{HWY1Ent7}$ can be locally calculated at a computer, which can be used to detect traffic congestion on the highway (e.g., a congestion event C = $((E_{10}^{SUM})_{HWY1Ent7})^{LESS12}$).

Example 4: At the four roads of a roundabout, sensors capture car movement and produce events: N for movement towards north, E for east, S for south, and W for west. $(N^{n-1};N^n)_2 + (E^{n-1};E^n)_2 + (S^{n-1};S^n)_2 + (W^{n-1};W^n)_2$ denotes a composite event for any car movement within a 2 second interval. Detection of this composite event indicates traffic flow, otherwise either no traffic or heavy congestion is assumed in the roundabout area.

Example 5: On a road, sensors are placed every kilometre to detect car movement, and they collect only specific subsets of events $E_{tagxxx001}$. Drivers may be interested in finding the speed of cars ahead of them to find travelling time by use of roadside sensors. This information can be transmitted over the air to drivers who are interested in estimating driving times.

Example 6: Primitive events include strong wind detected W, high humidity detected H, air pressure increase detected P, and above zero degree temperature recorded T. An alerting avalanche system is operated, which detects an event A when strong wind is detected followed by high humidity detection within 3 minutes, unless either *air pressure goes up* or *temperature goes above zero* occurs in between. A set of rules specifying reactions to three primitive events can express the combined behaviour. The above composite event A corresponds to the expression $(S;H)_3 - (P|T)$.

Example 7: *Notify the frozen products that are in the store, when the temperature changes 2 % in a 10 second time interval* can be expressed with ${}_{(s)}((E^{FIRST}+E^{LAST})_{10})$, where s denotes the sub-setting function (i.e., temperature change rate = 2 %).

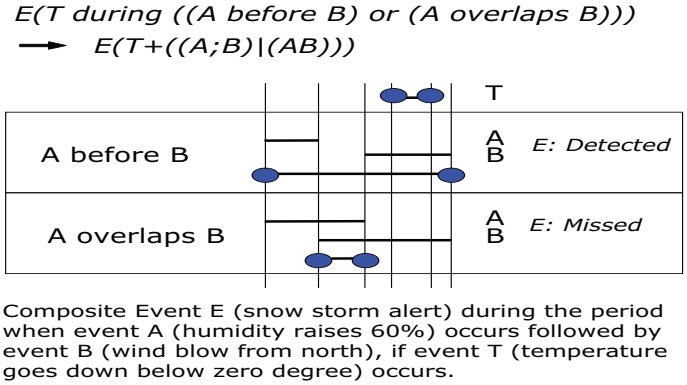


Figure 7.5: Semantic Ambiguity

7.5.2 Temporal Conditions

Defining temporal conditions for the semantics of composite events can be tricky, especially when timing constraints are important as in processing transactions. This may cause an incorrect interpretation according to the intuition of the user. Fig. 7.5 depicts a composite event E (*snow storm alert*): during the period when primitive event A (*humidity raises 60%*) occurs followed by primitive event B (*wind blows from north*), if primitive event T (*temperature goes down below zero degrees*) occurs. Two situations are shown. If we follow the interpretation of temporal conditions described in [GA04], in the first situation, event E is detected and in the second situation, event E is missed. In [GA04], *overlaps* and *during* only comprise the period when two events are simultaneously occurring, while every other operator takes the period over both event occurrences. This inconsistency may cause a problem. On both occasions, the natural interpretation of event E is the same. With the proposed definition, both examples will yield consistent results.

7.5.3 Interval Semantics

In most event algebras, an individual primitive and/or composite event occurrence is detected at a point time (i.e., the time of detection of the end of the occurrence). This may cause incorrect detection of operator combinations. The main reason for such problems is that composite events are defined for detection conditions but not in terms of occurrence conditions. Examples are nested sequence operators.

Instead, the detection time of composite events, using the interval of the occurrence, will solve this issue. In [All83], 13 basic interval relations are shown. Table 7.3 summarises and maps to

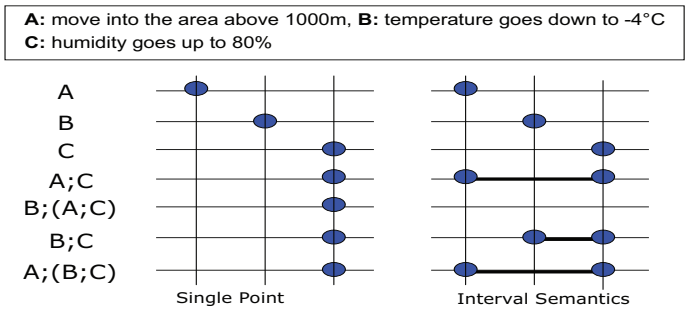


Figure 7.6: Point and Interval Semantics

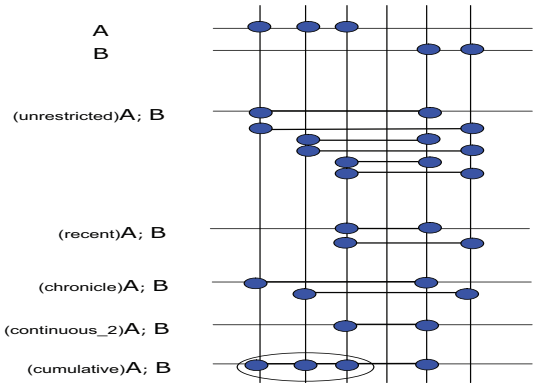


Figure 7.7: Event Consumption Policy

the interpretation of interval relations for composite event operators.

In [GA02][CL03b], the ambiguity of detection condition is described, which is illustrated in Fig. 7.6. When a point timestamp mechanism is used, an event $B;(A;C)$ is detected, although the actual order of event occurrences is A, B, and C. This is because detection of $B;(A;C)$ is dependent on $A;C$. Interval semantics with precise timing conditions solves this problem. In this example, the sequence B followed by $A;C$ is a relation of *overlap*, which is not considered as a *sequence*. Thus, $B;(A;C)$ would not be detected.

7.5.4 Event Context

Adding a policy defining the constraints provides a way to create modified operator semantics. This parameter-dependent algebra can accommodate different policies on event consumption or subsetting. Each operator is given a principal definition of the operation. Then, the various event contexts can be defined, acting as modifiers to the native operator semantics. Selection mechanism of event instances may be defined by this, creating a unique operator for each composite event. This helps resource constrained environments, e.g., keeping the most recent instance for future use instead all instances. Event filtering can be incorporated with event context as parameterisations.

Consumption Policy

For event consumption policy, five contexts can be defined: *unrestricted*, *recent*, *chronicle*, *continuous* and *cumulative*. Snoop [CM96] uses these contexts but is not capable of applying an individual context to different event operators. The parameterised algebra solves this problem. The following gives an informal definition for detecting sequence operation $A;B$. Fig. 7.7 illustrates the sequence operator with these contexts.

- Unrestricted: All combinations of instances of A and B.
- Recent: The most recent instance of A is used for composition.
- Chronicle: Let A be the initiator and B be the terminator. The initiator and terminator are paired uniquely in occurrence order. Thus, the oldest initiator pairs with the oldest terminator.
- Continuous: A moving time window determines selection of A. In Fig. 7.7, the window size is 2 and only the most recent A is used for composition.
- Cumulative: All instances of A are grouped that are used for composition.

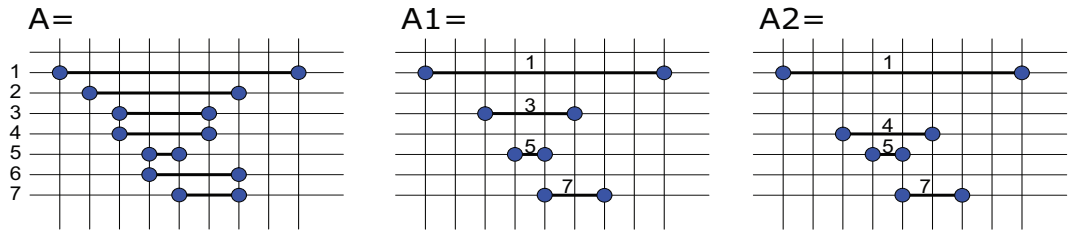


Figure 7.8: Subset Policy: Valid Restrictions of A

Subset Policy

The subset policy defines the subset of events to detect. When the subset policy is applied for efficient use of resources, the subset policy should not give crucial impact to unrestricted semantics. At the same time, efficient detection without wasting resources should be realised. For example, conjunction and sequence operations should be able to detect non-valid instances as early as possible. The main task of the subset policy is to make an effective algebra to implement in resource-constrained environments.

For example, the subset policy restricts an event stream, and a subset contains only instances with the same end-time. Thus, exactly one event instance with the same end-time and the maximal start-time is stored for the composition event detection. Fig. 7.8 shows the subset policy applying to an event stream A. From the instances of $A(2, 6, 7)$ with the same end-time, 2 and 6 must be removed. 5 and 7 do not share the end-time with the others. The choice whether to remove 3 or 4 results in two restricted event streams A, named $A1$ and $A2$.

Precision Policy

The precision Policy defines the required precision of the events to be detected. Dynamic spatial-temporal data from sensor networks is generated at a rapid rate and all the generated data may not arrive at the event correlation node over the network due to its lossy/faulty. Various techniques, including compression and model adaptation, have provided certain levels of guarantee [LM03]. On the other hand, defining the available precision becomes important if some imprecision of the collected data can be tolerated by the application. Parameters can be based on position, periodicity, urgency, relative deadline, or timeliness.

For example, *High, Default, Low* can map to:

- the ratio of sensor nodes that are awake: *80%, 20%, 5%*
- the delivered time-series data: *100%, 70%, 50%*
- the interval of data collection: *1 second, 10 seconds, 60 seconds*
- the frequency of data report: *Urgent, Periodic, Available.*

7.5.5 Duplication Handling

Interpretation of duplication and redundancy depends on the application, and it can be handled by *Selection* and *Aggregation* operators together with an event consumption policy. This enables an individual duplicate handling policy for the event stream.

A duplicate set of event instances D for event type A within the time period T can be defined by the composite event A_T^{ALL} or A_T^{LAST}. An aggregation operator can be used, e.g., A_T^{MAX}. A consumption policy can be applied over the composite event, e.g., ${}_{(recent)}A_T^{LAST}$.

7.5.6 Adaptation to Resource-Constrained Environments

I have used event algebra for the definition of composite events. Event algebra helps to use properties such as laws of distributivity to transform one form of event algebra to another to make detection more efficient or with fewer resources. As a simple and intuitive example, an expression $(A; B)_t$ can be transformed to $(A; B_t)_t$. This will restrict the buffer size for B before $(A; B)$ is evaluated.

Event Stack Size: Consider a composite event with a sequence operator, $A; B$, where the length of the instance of B never exceeds t (i.e., $B \equiv B_t$). The event stack size for A can be optimised by only storing A with maximum t time unit earlier. Furthermore, when the consumption policy defines A with maximum start-time for the composition, only a single instance of A is enough to be stored. Without this mechanism, multiple instances of A need to be stored.

The event algebra operation itself should not be restricted, and event contexts (e.g., consumption, subset, and precision policies) can be defined for restricting each operator. Wireless devices are resource constrained, and use of the algebra property can be used to determine necessary resources for detection of composite events. An expensive operation such as keeping event instances for a specific event type for infinite time can be prevented by applying rules as event context so that detection may become possible. If the event expressions are known before execution, a canned detection component can be created for common use. Once the semantics are defined in an event algebra, the implementation can be done by any means such as FSA or even by rule-based approaches (i.e., ECA).

7.6 Event Detection

Defining composite events in an event algebra achieves simplicity, but implementing the defined algebra in an efficient manner is difficult. After the investigation of different approaches [CM96], [Pro05], [Hay96], [PSB04], [CL03b], and [HB02], the current detection uses a simple imperative algorithm based on a combined approach of [CL03b] and [PSB04].

7.6.1 Detection Algorithm

The event detection process consists of three stages: applying event context policies to the incoming stream (e.g., consumption, subset, and precision policies), matching the operations on event types, and event composition operations. Fig. 7.9 shows the detection algorithm of event composition operations. When this algorithm is executed, the matching operations on the target event types, and restriction on the event streams (i.e., step1 and 2), have already been applied. Depending on the consumption policy, duplicated events may be input to the detection.

Let E be a composite event expression to be detected, and index sub-expressions of E from 1 to k in bottom-up order. Thus, the final result of the operation is $E^k (= E)$, and an initial expression is a primitive event $E^1 \in P$, where P is a finite set of primitive events. The main loop selects sub-expressions dynamically and calculates the current composite event instance from the current event and stored information. This operation loops over every time instant.

Let E^i be an output of composite event in an individual step, E^x a left hand side event, and E^y a right hand side event. Each operation in the expression needs its own indexed state variables (e.g., past events, time instant, and spatial information). Each operator in the composite event expression requires its own variables to keep the state, and thus indexed variables are denoted from 1 to k. Let v_i be a variable of the output composite event instance of E^i, and thus the final v_k contains the result of the algorithm.

FOR i = 1 to k
CASE E^i OF
$[E^i \in P]$: $v_i = E^i \vee \phi$
$[E^x + E^y]$:
CASE *timestamp* OF
$[x_{i(start-time)} < v_{x(start-time)}]$: $x_i = v_x$
$[y_{i(start-time)} < v_{y(start-time)}]$: $y_i = v_y$
$[v_{x(start-time)} \leq v_{y(start-time)}]$: $v_i = x_i \cup v_y$
[OTHERS]: $v_i = v_x \cup y_i$
$[E^x | E^y]$:
CASE *timestamp* OF
$[v_{x(start-time)} < v_{y(start-time)}]$: $v_i = v_x$
[OTHERS]: $v_i = v_y$
$[E^x E^y]$:
$v_i = \phi$
IF $v_x \neq \phi$ THEN
REPEAT event α in z_i
IF $(\alpha_{(end-time)} = v_{y(start-time)})$ THEN $v_i = \alpha$
IF $(v_i \neq \phi)$ THEN $v_i = v_i \cup v_y$
$[E^x; E^y]$:
$v_i = \phi$
IF $v_x \neq \phi$ THEN
REPEAT event α in z_i
IF $(\alpha_{(end-time)} \leq v_{y(start-time)})$ THEN $v_i = \alpha$
IF $(v_i \neq \phi)$ THEN $v_i = v_i \cup v_y$
$[E^x || E^k]$:
$v_i = \phi$
IF $v_x \neq \phi$ THEN
REPEAT event α in z_i
IF $((\alpha_{(start-time)} = v_{y(start-time)}) \wedge (\alpha_{(end-time)} = v_{y(end-time)}))$ THEN $v_i = \alpha$
IF $(v_i \neq \phi)$ THEN $v_i = v_i \cup v_y$
$[E^{x*}]$: $v_i = v_x \cup y_i$; $x_i = v_i$
$[E^x - E^y]$:
CASE *timestamp* OF
$[t_i < v_{y(start-time)}]$: $t_i = v_{y(start-time)}$
$[t_i < v_{x(start-timte)}]$: $v_i = v_x$
[OTHERS]: $v_i = \phi$
$[E^{xN}]$: $v_i = funcN(x_i, v_x)$; $x_i = v_i$
$[E^{xG}]$: $v_i = funcG(x_i, v_x)$; $x_i = v_i$
$[E^x_s]$:
CASE *spacestamp* OF
$[(v_{x(spacestamp)}) \equiv s]$: $v_i = v_x$
[OTHERS]: $v_i = \phi$
$[E^x_t]$:
CASE *timestamp* OF
$[(v_{x(end-time)} - v_{x(start-time)}) \leq t]$: $v_i = v_x$
[OTHERS]: $v_i = \phi$
ENDFOR

Figure 7.9: Event Detection Algorithm

Let x be a variable for storing the left-hand side of past event, y be a variable for the right-hand side of past event, t be a time instant, s be a spatial instant, and z be a set of instances of event. The variables x, y, t, s, z are used for storing past information for each step of composition operation. $e_{(start-time)}$ and $e_{(end-time)}$ denote the start- or end-time of the interval timestamp in event e. Creation of new interval timestamps after the detection of the target composite events

follows the definition in Table 7.3.

In the main loop, when one of the event instances is empty (e.g., E^x happens to be ϕ in conjunction with operation $E^x + E^y$) or both are empty, the existing event is carried out for the operation or it becomes automatically empty. To highlight the main algorithm, these intuitive operations are not shown in Fig. 7.9. When the composite event expressions are static and can be defined before the system deployment, the operation can be statically determined as the canned operation.

Conjunction:
Both sub-expression with the maximum start-time are kept in the variables x and y. Thus, when entering the operation of the conjunction, simply combining both instances results in the output event.

Disjunction:
The disjunction operation simply takes the event with the minimum start-time. When the start-times are the same, the event with the minimum end-time is taken. When start and end-times are the same, the right sub-expression is automatically taken.

Concatenation:
Concatenation is a special case of the *Sequence* operation, when timestamps are *overlaps*, $finishes$, or *starts* relations between two sub-expressions.

Sequence:
Multiple instances of the left-hand side sub-expression are kept in the variable z. For example, the composite event $A;B$ requires storing several instances of A in z. Once B occurs, the start-time of the B instance determines the A to be combined with it to form the result of $A;B$.

Concurrency:
Concurrency is a special case of the *Sequence* operation, where both start and end-times on sub-expressions are the same.

Iteration:
Iteration is a special case of $Conjunction$ operation, when the same event types are elements of the conjunction operation. The variable x keeps the accumulation of composite events.

Negation:
For example, *Negation* $E = A - B_T$: an instance of A is an instance of $A - B$ excluding the occurrence of an instance of B within T time interval. Thus, if the current instance of A is valid, the instance of B with maximum start-time within T can be the only one to invalidate it. Storing a start-time is therefore sufficient, since the end-time is determined automatically, which is earlier than the end-time of the instance of B.

Selection and Aggregation:
Selection and Aggregation operate on the corresponding functions (e.g., $funcN$ or $funcG$), where the input parameters are the existing event and the current event.

Spatial Restriction:
If the spatial value is defined similar to the time system (i.e., countable discrete value), then the operation could be the same as $Temporal\ Restriction$. The current implementation is a simple comparison between the previous event and the current event.

Temporal Restriction:
The temporal restriction is a simple operation to carry out the existing event or remove it based on the duration of the current time instant.

7.7 Temporal Ordering

Recent progress in Internet business requires precise timing for processing data. For example, timing is critical for buying and selling orders for stock markets and auctions. Multimedia synchronisation for real-time tele-conferencing, distributed network gaming, and traditional network monitoring also require precise timing. Timing accuracy depends on the applications, and some applications (e.g., multimedia synchronisation, Internet gaming) may require real-time traffic for inter-arrival times in the order of milliseconds. On the other hand an Internet auction system may require inter-arrival times in the order of a few seconds.

Time is fundamental information in ubiquitous computing, especially in WSNs for data aggregation from distributed sensor nodes to judge data duplication/redundancy. Determination of directions and speed of moving objects requires highly accurate timestamps. For example, to detect the direction and speed of a phenomenon, mobile computing devices with sensors and clocks can record the time of an event and forward this information to other devices surround them. The temporal ordering of events originating from different devices has to be determined. Events can be triggered by physical phenomena, such as glaciers and earthquakes, and the order of occurrence of sensed data is important. When real-time constraints involve event correlation, semantic ambiguity may arise from the timing of multiple instances of the same event, which may be created due to the nature of unstable network environments. Real-time in this dissertation refers to real-world event occurrences.

The notion of a global time provided by synchronised local clocks in distributed system environments is important for the semantics of event-driven systems. Distributed publish/subscribe systems follow the model of store-and-forward paradigms, and message delay is unavoidable. Traditional message ordering based on transport layer protocols is not applicable. Thus, each message needs to contain a timestamp that should be used for correlation. In existing systems, the semantics of event order often depends on the application. Even the established Java Message Service (JMS) only guarantees the event order within a session, with a session being a single-threaded context that handles message passing. Thus, temporal order using the original timestamps is necessary and construction of the global view is critical.

The current time synchronisation in the Internet is based on NTP (Network Time Protocol), which provides milliseconds-scale error margins. NTP requires a static distribution of time synchronised servers. In WSNs, there are variants, and in 802.11 networks, synchronisation is performed by precise clock agreement within a cluster. In ubiquitous computing environments, events flow over heterogeneous networks, and although issues on temporal ordering are essential there are not yet any established solutions. When GPS is available, there will be ways to obtain the same level of accuracy of timestamps as in Internet environments. Because of high power consumption and a required line of sight to GPS satellites, use of GPS for time synchronisation may not be suitable over wireless ad hoc networks. The time model in a wireless ad hoc network environment is different from the wired Internet environment since the system lacks a global clock, or centralised controller, expected within the systems.

This section discusses the issues around temporal ordering of events in the context of event correlation and outline our solution.

7.7.1 Time Model

Temporal ordering of events is greatly impacted by the event detection method, the timestamping method, and the underlying time systems. Time in the real world may be continuous, but I define time as discrete and finite with limited precision. I assume that time has a fixed origin and

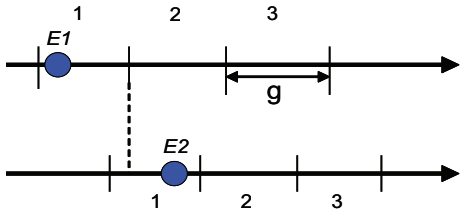


Figure 7.10: 2g-Precedence Model

equidistant time domain. Absolute time systems (e.g., 15:00 GMT) can map to the time-axis. Two constants are defined: NOW and ∞ (infinity). NOW is a moving time point that keeps track of the current time; ∞ is used to represent an indefinite time interval.

The ordering of events within a process is well understood regarding the concept of time. This changes fundamentally when events are considered in distributed systems. In distributed environments, a set of processes communicates by exchanging messages, and the delay of messages can not be ignored compared to the time between events within a process. Furthermore, in the asynchronous model of distributed message systems, the message propagation delay is finite but not deterministic. All the nodes in the network have their own clocks, and individual clocks tend to drift apart. Deterministic clock synchronisation algorithms require communication between nodes and their upper bound on clock skew is lower than $(d_{max} - d_{min})(1 - 1/N)$, where d_{max} and d_{min} are the maximum/minimum message delays in the system respectively, and N is the number of nodes in the network. As the system size increases, this value also increases. Messaging systems require real-time information on the occurrence times of events, and the logical time is not sufficient.

Timestamping: Most point-based timestamps consist of a single value indicating a point in time. An interval-based timestamp with interval semantics is used (see Section 4.4.2).

7.7.2 Time Systems

The accuracy of timestamps on the primitive event in an event message underlies an error caused by the time systems. I briefly look at the existing time systems supporting real-time mechanisms. For real-time support, a common solution in wired networks provides a virtual global clock that bounds the value of the sum of precision and granularity within a few milliseconds. The following approaches aim to support a real-time mechanism:

- 2g-Precedence model
- Network Time Protocol (NTP)
- Interval-based time system

The 2g-Precedence model is extended for distributed event ordering and composite event detection using 2g-precedence-based sequence and concurrency operators [Sch96]. Ordering events from different sites is only possible if these events are at least two clock ticks apart. The 2g-precedence model can only be applied in networks where servers are interconnected. Fig. 7.10 shows an example for a case where events $E2$ and $E1$ cannot be ordered, because they are less than two clock ticks apart (i.e., 2g). NTP is an Internet standard that supports assigning real-time timestamps with given maximal errors. However, in open distributed environments, not all servers are interconnected and event ordering based on NTP may lead to false event detection. In [LCB99], timestamps of events can be related to UTC (Universal Coordinated Time) with bounded accuracy, and event timestamps are modelled using accuracy intervals with interval timestamps. They use NTP that provides reference time injected by a GPS time server with

reliable error bounds.

For wireless network environments, [PB04] presents a GPS-based virtual global clock, which is used for timestamping events, and deploys a concept similar to 2g-precedence. Without GPS, there is no means of synchronising the clocks of all the nodes in a deterministic fashion. Logical time is not sufficient to determine temporal ordering, because causal ordering of events in the real world must be maintained. Thus, physical time has to be used, which requires clock synchronisation. However, most of the synchronisation algorithms rely on partitioned networks. Post-facto synchronisation [ER02] is based on using unsynchronised local clocks and synchronisation is limited within the transmit range of the nodes. In [Röm01], they take a similar approach to [ER02] using unsynchronised clocks.

7.7.3 Experiments with Unsynchronised Local Clock

This section describes experiments taking temporal message ordering over a time synchronisation algorithm [Röm01] for ad hoc networks.

When a publisher node records an event in real-time it generates a timestamp using its unsynchronised local clock, which is passed to other nodes. The algorithm uses interval-based timestamps with lower and upper bounds for representing the exact value. At each node, it transforms them to the local time of the node instead of adjusting the clocks. This operation is propagated until the message arrives at the destination node. The algorithm does not require much resources, message overhead is low, and topology changes do not affect. Thus, it is well suited for resource-constrained wireless ad hoc networks. The initial interval timestamp represents the event detection/creation time, i.e., when an external event has been sensed. An assumption is that event generation must be inside the computer device, meaning a ZERO-length time interval. However, a real world event is often sensed by an external sensor that is connected to the computing device.

The time transformation between the nodes is:

$$\Delta C \quad \longrightarrow \quad [\frac{\Delta C}{1+\rho_1}, \frac{\Delta C}{1-\rho_1}] \quad \longrightarrow \quad [\Delta C \frac{1-\rho_2}{1+\rho_1}, \Delta C \frac{1+\rho_2}{1-\rho_1}]$$

ρ denotes the clock skew. ΔC indicates the local clock of node 1, which is transformed to the interval timestamp (estimated real-time) at first, then the interval timestamp is transformed to the interval timestamp relative to the local time of node 2. In wireless network environments, a constant message delay cannot be assumed since the network is highly dynamic and message delay needs to be determined for each transferred message for accuracy. A message transfer between two nodes often involves two messages: message-send and acknowledgement-back. This makes it possible to measure the round trip time (rtt) using the local clock of the sender. The rtt is the time from message departure from the sender until arrival of the acknowledgement at the sender. The message delay can be estimated by the lower bound 0 and the upper bound rtt. Then, the estimated message delay is transferred from sender to receiver by another pair of message exchange.

In Fig. 7.11, the event is sent from the sender at t_2 and arrives at the receiver at t_5. An estimate of the delay of this event in terms of the sender's clock is:

$$0 \leq d \leq (t_3 - t_2) - (t_6 - t_5)\frac{1-\rho_s}{1+\rho_r}$$

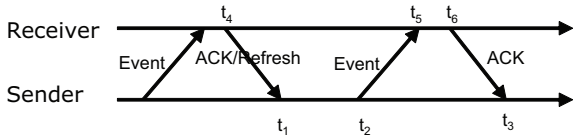


Figure 7.11: Message Graph

where ρ_s and ρ_r are the ρ values for sender and receiver respectively. An estimate in terms of the receiver's clock is:

$$0 \leq d \leq (t_5 - t_4) - (t_2 - t_1)\frac{1-\rho_r}{1+\rho_s}$$

The receiver may know the estimate of delay without message exchange if the event carries $t_2 - t_1$.

Fig. 7.12 shows the timestamp transformation. A publisher node wants to pass a timestamp to a subscriber along the chain of routing nodes. Each node i has three attributes: the local time r for message-receive time, the local time s for message-send time, and the clock drift ρ. Each edge has two attributes: the round trip time (rtt) and the idle time ($idle$). At the publisher node the timestamp forms:

$$[r_1, r_1] = [NOW, NOW]$$

which at the subscriber node results in:

$$\begin{aligned}\big[r_N - (1+\rho_N)\sum_{i=1}^{N-1}\frac{s_i - r_i + rtt_{i-1}}{1-\rho_i} - rtt_{N-1} \\ +(1-\rho_N)\sum_{i=1}^{N-1}\frac{idle_i}{1+\rho_i}, \\ r_N - (1-\rho_N)\sum_{i=1}^{N-1}\frac{S_i - r_i}{1+\rho_i}\big]\end{aligned}$$

In the publisher node, the timestamp consists of just the single point in time NOW. For the subscriber node, $s - r$ indicates the processing time from message arrival time until the send-time in the previous node. The round trip time (rrt) minus the idle time ($idle$) is used as the upper bound for message delay. The timestamps are updated along their way by each node using their own local clocks. To preserve high precision of the timestamp, the algorithm for the transformations uses time intervals consisting of lower and upper bounds for the exact value. Due to clock shift and event propagation delay, the final timestamp is expressed as lower and upper bound relative to the local time. This algorithm assumes that the maximum clock skew

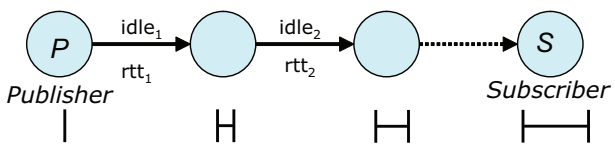


Figure 7.12: Message Delay

is known. Note that the interval timestamp is only necessary for events that are monitored by event correlation services for specific subscription groups.

The interval arithmetic is based on [PSB04] with real-time consideration. For example, to determine whether $[t_1^l, t_1^h]$ happened before $[t_2^l, t_2^h]$, the following formula can be used:

$$[t_1^l, t_1^h] < [t_2^l, t_2^h] = \begin{cases} YES: & t_1^h < t_2^l \\ NO: & t_2^h < t_1^l \\ MAYBE: & otherwise \end{cases}$$

Object Tracking

This experiment is to confirm the accuracy of the message ordering described in the previous section. It is operated in a simulated environment, and the scenario is that movement of two objects is traced by 20 smart devices along their route. The subscriber's interest is obtaining data on the leading object. The subscriber device is within 5 hops distance from any devices. Object 1 shows linear movement with a constant speed of 50 metres per second, while object 2 shows irregular speed. After 2000 seconds at a distance of 100 kilometres from the starting point, the two objects meet each other. The average speed is about 180 kilometres per hour such as strong wind or speedy cars. Fig. 7.13 shows the movement of the objects in real-time.

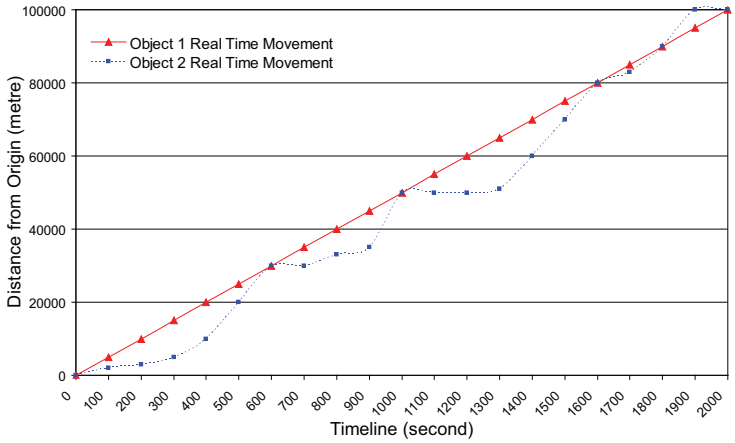
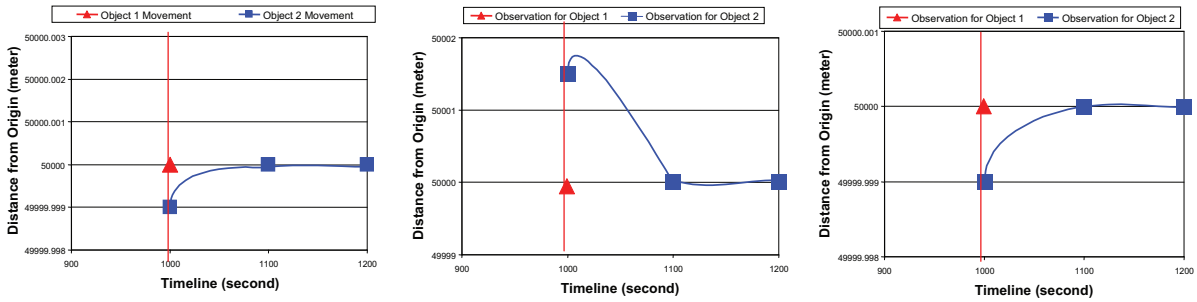


Figure 7.13: Object Tracking



(a) Real-Time (b) Drifted Clock (c) Timestamp Transformation

Figure 7.14: Order of Object 1 and 2

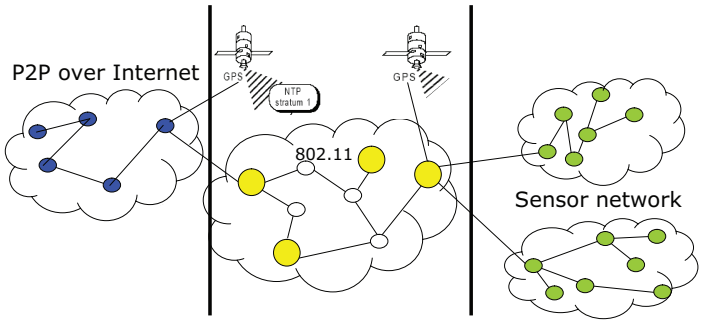


Figure 7.15: 2-Tier Timestamp Transformation

The experiment setting is that 20 devices are placed along the route of two objects, and each device records the passing object location. After 600, 1000, and 1600 seconds, the two objects almost cross each other, and the subscribers are interested in the order of the two objects at these points. The observing smart devices have various clock drift values. A typical value of ρ with today's hardware is 10^{-6}, which means a one second clock drift away from real-time in eleven days. Thus, 10^{-6} is used as the maximum value for the clock drift, and each device ranges (from 1 to 9)$*10^{-7}$. Initially, all devices are synchronised with the real-time clock, and there is no time synchronisation process during the experiment, which lasts for about 30 minutes. Fig. 7.14 shows a magnified version of the section at 1000 seconds after the start. Fig. 7.14(a) shows that object 1 (*triangle*) is ahead. The traced movement from the information from devices with drifting clocks is shown in Fig. 7.14(b) that depicts object 2 (*square*) leading object 1, which conflicts with the real-time result. Fig. 7.14(c) shows the movement tracked by *Timestamp Transformation* with the correct order. The number of the route nodes between the source of event and the destination is two on all devices in this experiment, and the clock drift value is set at random but consistent for a device once chosen. This experiment demonstrates that the temporal ordering with timestamp transformation can distinguish two events with time separation in the order of 1 millisecond.

7.7.4 2-Tier Timestamp Transformation

In many real world scenarios, wireless networks may be deployed with relay nodes to the Internet, especially for collecting data from WSNs. It is likely that relay nodes can connect to GPS. Thus, the use of GPS in distributed systems is reconsidered, including wireless network environments. However, it is not suitable to use GPS for a large number of smart devices because of high power consumption and a required line of sight to satellites. GPS does not work indoors, underwater, and in forests. But GPS may be key for providing accurate time adjustment at certain nodes that are less resource constrained within wireless ad hoc networks.

In principle, timestamps embedded in events are used for correlation to provide a real-time mechanism. I define two categories of network environment: 1) NTP is deployed with GPS, such as the Internet domain, and 2) networks are isolated in ad hoc mode without GPS or any other deterministic time synchronisation mechanism (Fig. 7.15). For the first category, interval-based point timestamps for primitive events are used, where the low and high end values of the interval are computed as described in [LCB99] to allow for clock uncertainty and network delay. For timestamping composite events, I use interval-based semantics, unlike [LCB99], where a new timestamp is taken on detection of a composite event. For the second category, I investigated the temporal ordering based on [Röm01] (see the previous section). The idea of the algorithm

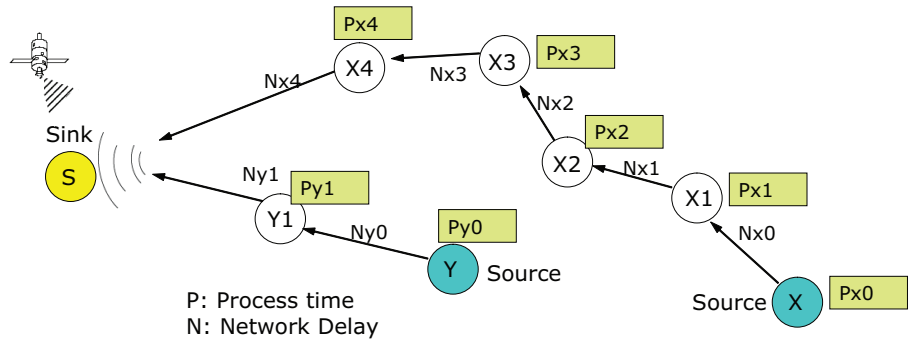


Figure 7.16: Lightweight Local Clock Propagation

is not to synchronise the local computer clocks of devices but instead to generate timestamps from a local clock. When such locally generated timestamps are forwarded to the devices, transformation of the timestamp is performed using the local time of the receiving device. This approach produces high accuracy of temporal ordering. The question is whether this level of accuracy is really necessary in the majority of correlation services in publish/subscribe systems, especially considering the typical scenario from WSNs through MANETs to Internet backbone nodes.

Thus, I propose a simplified protocol *Lightweight Local Clock Propagation* for wireless network partitions, described in the next section. The proposed approach is a coordinated approach with and without the use of GPS. interval-based timestamps are used throughout the transformation of timestamps in heterogeneous networks. Sensor events could be aggregated at gateway nodes with transformed timestamps and passed towards a subscriber node in an Internet environment, where GPS-based time synchronisation is deployed.

7.7.5 Lightweight Local Clock Propagation

Lightweight Local Clock Propagation (LLCP) is on-demand based timestamp synchronisation. The basic idea is that a timestamp is generated to record the event occurrence time, and the timestamp is updated along its way over the network, by each node, using its own clock. As a result of clock shift and message propagation delay, the final timestamp is expressed as a lower and upper bound. Fig. 7.16 shows the operation from source nodes X and Y to the sink node S. Each node calculates its processing time using a local clock. At the sink node, the sum of the processing times is subtracted from the event arrival time to estimate the occurrence time. Comparable timestamps are therefore created at sink nodes rather than network-wide. This algorithm requires the following two assumptions:

- Network delay is negligible where nodes are close to the radio or dense networks.
- Clock drift is negligible where the node carries an oscilloscope that guarantees less than 10 ppm drift (one part per million (ppm) corresponding to 1 second in 11.5 days).

Thus, the timestamp value at the sink node T_s of an event from the source node x is:

$$T_s = T_x + \sum_{i=0}^{k} P_x + \sum_{i=0}^{k} N_x$$

where T_x is the timestamp value at the source node and k is the number of hop counts from the source node to the sink node.

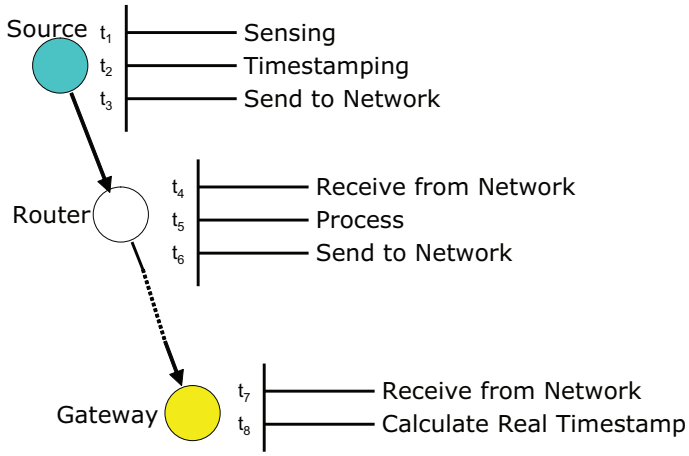


Figure 7.17: Message Propagation

In Fig. 7.17, time propagation along the event propagation is depicted. In a typical WSN board, the Analogue to Digital Converter (ADC) resides between the sensing unit and the processor. The delay caused by passing through ADC is negligible ($t_2 - t_1 \approx 0$). The processor adds timestamps to the sensor data before sending it out to the network. There may be some processing delay between processor and network components, which is normally negligible ($t_3 - t_2 \approx 0$, $t_6 - t_4 \approx 0$). I assume that if there is any significant delay, it is measurable and can be included as an interval within the timestamp. When the data sampling rate is high, and the processor gives timestamps on a group of raw sensed data, timestamp accuracy is dependent on the applications. LLCP needs further experiments and will require more work.

7.8 Correlation Services in Publish/Subscribe Systems

I have discussed a platform for ubiquitous computing (i.e., SOA) in Chapter 3. This new platform enables the seamless use of various resources in physically interacting environments. The system architecture to support such platforms will be service management, with communication based on the publish/subscribe paradigm.

Event broker nodes on the Internet backbone can act as a gateway from a WSN, performing data aggregation and distributing filtered data to other networks, based on content. Mobile devices can be deployed in remote locations without a network infrastructure. They can act as brokers to keep sensed data while in remote locations, and convey the data to the brokers at the Internet backbone nodes by various communication methods or by moving themselves. Event broker nodes that offer data aggregation services can efficiently coordinate data flow. Especially when event-based communication is implemented in P2P overlay networks, the construction of event broker grids will extend the seamless messaging capability over scalable heterogeneous network environments. Thus, it is essential to integrate event correlation service with event brokers.

Complimentarily stream data management is extended to support distributed environments (see Section 4.3.1). This extension aims to seamlessly integrate query processing in WSNs and data mining server networks, where primary concerns are power and performance (e.g., latency) in server networks, which can be unified into a service metric. Examples are STREAM [ABB+03] and Borealis [AAB+05], aiming at the next generation of stream processing systems (see Fig. 7.18 for an architectural overview of Borealis). In these distributed stream-processing systems, query operators are placed at the node, where resources are available, based on the application's goal.

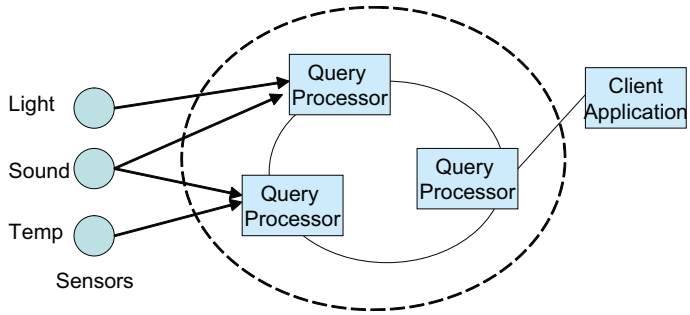


Figure 7.18: Borealis Distributed Stream Processing

This placement operation is difficult, because network conditions are dynamic, while streams interact with each other.

I expect event correlation services in publish/subscribe systems will eventually merge with stream data management. Placement of correlation services in an event broker overlay (e.g., stream processing servers) will require more complex consideration and policy-based placement.

7.9 Experiments

Event detection by imperative algorithms is implemented as a prototype with limited functions. Thus, experiments in this section highlight the resource efficiency by controlled event consumption mechanisms such as time restriction, subset policies, and use of durative events. This significantly reduces event buffer for composite event detection. Object tracking with real Active BAT data is described in Section 7.10.

7.9.1 Prototype Implementation

The prototype has about a 20KB class library in Java with JDK 1.5 SE and J2ME CDC Profile, which supports resource constrained devices. The prototype currently does not provide a high level language interface, and composite events need to be described as event expressions with the operators described in Table 7.4. For example, a composite event expression $A; B; C$ must be described as $(A; B); C$ or $A; (B; C)$. The prototype implements a subset of operators and tested. Event expressions are parsed and kept. Each valid expression is associated with a correlation listener that operates the correlation process. The primitive events are processed every 0.1 seconds. The experiment expects the deterministic results. Each experiment is repeated minimum 3 times and it produced expected deterministic results.

The event operators for composite event expressions are shown in Table 7.4. An example of a valid expression is $((A; B)_5)|((A; B) - B)$, and several experiment results are shown below.

+	Conjunction
|	Disjunction
=	Concatenation
;	Sequence
:	Concurrency
*	Iteration
−	Negation
”	Selection
@	Spatial Restriction
!	Temporal Restriction

Table 7.4: Operators in the Prototype

Example 1

Fig. 7.19 shows the basic operation of two different composite event detections.

```
Timeline:         1 2 3 4 5 6 7 8
---------------------------------------
Primitive Event:  A C B C D B A D
---------------------------------------
Composite Event Expression: A;C
                            A;D

At 2:  (Detected A;C)
At 4:  (Detected A;C)
At 5:  (Detected A;D)
At 8:  (Detected A;D)
```

Figure 7.19: Composite Event Detection: Multiple Composite Event Detection

Example 2

Fig. 7.20 shows that an ordered set of primitive events (i.e., input stream) should trigger both specified composite events (e.g., simultaneous Disjunction operation). The composite event detection is reported at the time-line described.

```
Timeline:         1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
-----------------------------------------------------------------
Primitive Event:  A B C D A C E D C B  A  D  B  A  C  B  D
-----------------------------------------------------------------
Composite Event Expression: A;((B;D)|(C;D))
                            A;((C;D)|(B;D))

At 4:  (Detected A;((B;D)|(C;D)) A;((C;D)|(B;D)))
At 8:  (Detected A;((B;D)|(C;D)) A;((C;D)|(B;D)))
At 12: (Detected A;((B;D)|(C;D)) A;((C;D)|(B;D)))
At 17: (Detected A;((B;D)|(C;D)) A;((C;D)|(B;D)))
```

Figure 7.20: Composite Event Detection: Simultaneous Disjunction

Example 3

Fig. 7.21 shows the associativity of sequence.

```
Timeline:         1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
-------------------------------------------------------------------
Primitive Event:  A B C D E B B D A E  D  C  B  A  E  D  C  D
-------------------------------------------------------------------
Composite Event Expression: (A;B);(C;D)

At 4:  (Detected (A;B);(C;D))
At 7:  (Detected (A;B);(C;D))
At 8:  (Detected (A;B);(C;D))
At 11: (Detected (A;B);(C;D))
At 13: (Detected (A;B);(C;D))
At 16: (Detected (A;B);(C;D))
At 18: (Detected (A;B);(C;D))
```

Figure 7.21: Composite Event Detection: Associativity of Sequence

Example 4

Fig. 7.22 shows the use of temporal restriction. The composite event $(A_5); B$ denotes that a primitive event A is valid for 5 time units and, under that condition, event B follows A must be detected. The target composite event is detected up to the time line 7 in Fig. 7.22, however after time unit 8, an event A is no longer valid.

```
Timeline:         1 2 3 4 5 6 7 8 9
---------------------------------------
Primitive Event:  A A B B B B B B B
---------------------------------------
Composite Event Expression: (A_5);B

At 3:  (Detected (A_5);B)
At 4:  (Detected (A_5);B)
At 5:  (Detected (A_5);B)
At 6:  (Detected (A_5);B)
At 7:  (Detected (A_5);B)
```

Figure 7.22: Composite Event Detection: Temporal Restriction

Example 5

Fig. 7.23 shows the use of the negation operator. The composite event $(A; B) - C$ denotes that a primitive event A is followed by B, but event C must not occur between the occurrences of A and B. The result shows the correct detection.

```
Timeline:         1 2 3 4 5 6 7 8 9 10 11 12
-----------------------------------------------
Primitive Event:  A A B A A C C B C A  D  B
-----------------------------------------------
Composite Event Expression: (A;B)-C

At 3:  (Detected (A;B)-C)
At 12: (Detected (A;B)-C)
```

Figure 7.23: Composite Event Detection: Negation

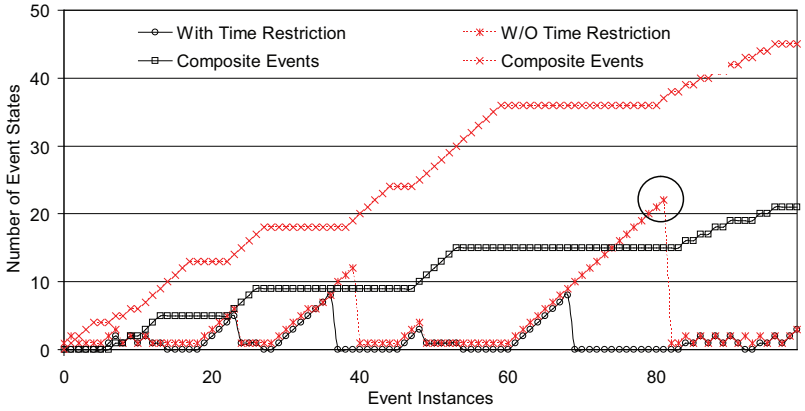


Figure 7.24: Time Restriction

7.9.2 Controlled Event Consumption

This experiment highlights the efficiency of restriction rules: temporal restriction and subset policies. I used surface meteorological data from the NOAA Aeronomy Laboratory TRMM profile system recorded in 1999 for the experiments [Lab99]. The data contain wind, temperature, relative humidity, pressure, and solar radiation. Data from each instrument were sampled every 0.5 seconds. Every 10 seconds, a 10 second average was transmitted to a base station. As part of the 10 second average, a timestamp was added to the data. The clock is kept in UT (*Universal Time*) time and is set to the GPS time standard every week.

The logged data have been assembled for individual events and a discrete event generator simulates event occurrences as if sensed data are reported from the sensor network in a single hop communication range. Interval-based timestamps are generated based on the embedded timestamp values. Events are assumed to be ordered. The composite event used in the experiments is as follows: humidity below 60% (H) followed by temperature over 30% (T). In the first experiment, *time restriction* is applied: the sequence operation is meaningful when it occurs within 10 seconds, denoting $(H;T)_{10}$. In the second experiment, a subset policy is applied: only one event instance with maximal start-time is used for correlation.

Time Restriction

Fig. 7.24 shows a simulation of memory usage with the number of event states to be kept. For the detection policy, the most recent instance of H is used. The time restriction value is set to 10, a relatively large number. Time restriction is a similar concept to event detection with a sliding window. With our approach, however, the semantics are unambiguous, and the capability for individual definition of each event gives another advantage. Our approach ensures that detection of composite events can be efficiently implemented with restricted resources, which can be critical for applications. Throughout the detection, all instances of H that have ended more than 10 time units ago can be discarded. The number of detected composite events is also illustrated in Fig. 7.24, and shows that only half of the composite events are detected when a time restriction is specified. In other words, it prevents stale data being used for detection of composite events. If undetected composite events impose loss of information, then the time restriction number should be set to the correct value. This figure shows that stale events are removed during the composite event detection process. The circled point in Fig. 7.24 identifies a potential risk of exceeding device resources (e.g., out of memory), without the time restriction. The gap between

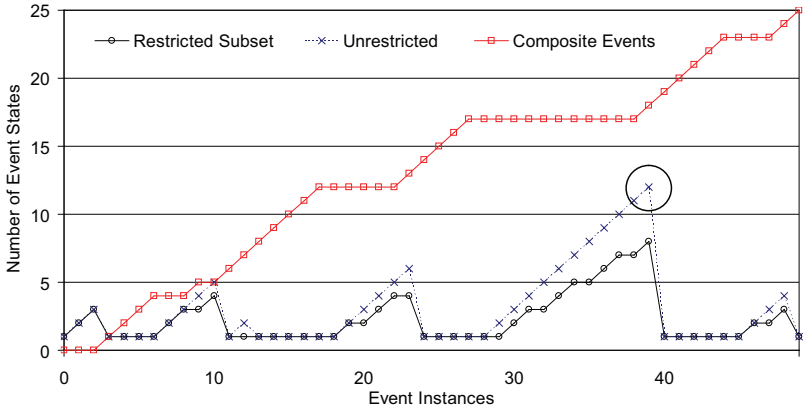


Figure 7.25: Subset Policy

the approach *with and without* time restriction shows how much memory can be saved.

Subset Policy

Fig. 7.25 illustrates memory usage when a subset policy is applied: the rule is that an event instance has no same end-time instance and exactly one with maximal start-time. This denotes $({}_{(S)}H;T)_{10}$, where S is the subset policy. This is an equivalent sequence operation as $(H;T)_{10}$, but with the subset policy. It requires less state information to be kept (Fig. 7.25). The circle points out the gap where the subset policies save memory. Both cases (i.e., with and without subset policy) show the same results for composite event detection.

7.10 Object Tracking with Active BAT

Sentient computing is ubiquitous computing using sensors to perceive the environment and react accordingly. Sensors are used to construct a world model, which allows location-aware or context-aware applications. One research prototype of a sentient computing system has been developed at AT&T Research in the 1990s, and this research continues at the University of Cambridge as the Active BAT system [HHS+99]. It is a low-power, wireless indoor location system, which is accurate up to 3 cm. It uses an ultrasound time-of-flight trilateration technique to provide more accurate physical positioning. Users and objects carry Active BAT tags. In response to a request that the controller sends to a BAT via short-range radio, a BAT emits an ultrasonic pulse to a ceiling-mounted receiver grid. At the same time, the central controller sends the radio

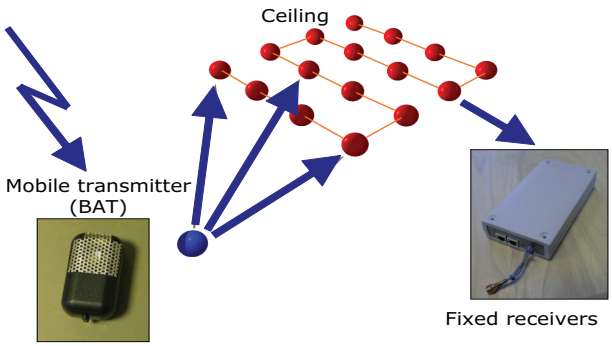


Figure 7.26: Active BAT

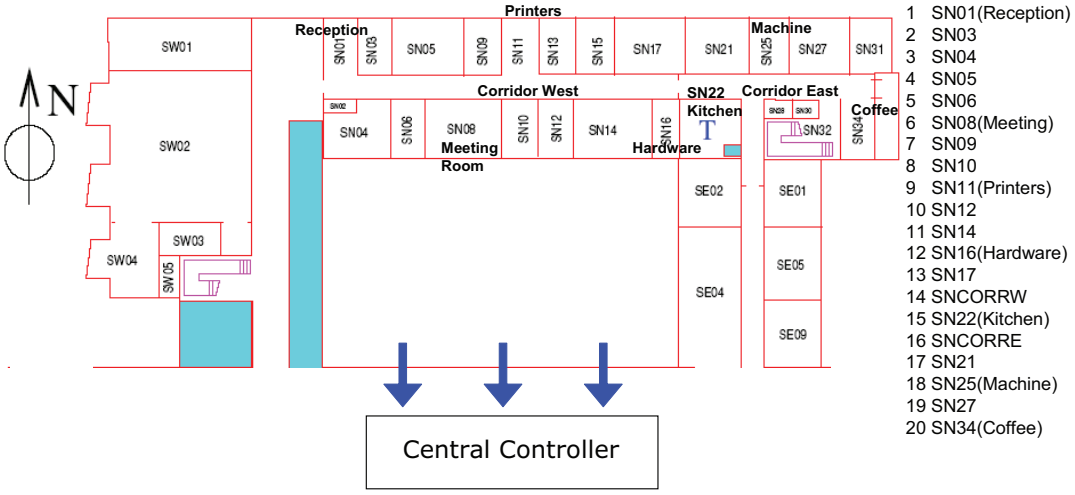


Figure 7.27: Active BAT Location Map

frequency request packet to the ceiling sensors to reset the baseline via a wired serial network. Each ceiling sensor computes the distance to the BAT from the time interval from the baseline to ultrasonic pulse arrival from the BAT. This distance value is forwarded to the central controller, where the trilateration computation is performed for object location. Statistical pruning removes erroneous measurements by sensors caused by a ceiling sensor detecting a reflected ultrasound pulse, instead of one that travelled along the direct path from BAT to sensor. Fig. 7.26 depicts an overview of the Active BAT system.

SPIRIT (SPatially Indexed Resource Identification and Tracking)[HHS+99] provides a platform for maintaining spatial context based on raw location information derived from the Active BAT location system. It uses CORBA to access information and spatial indexing to deliver high-level events such as *Alice has entered the kitchen* in context aware applications. SPIRIT models the physical world in a bottom up manner, translating absolute location events for objects into relative location events and calculating containment and overlap relationships among such spaces, by means of a scalable spatial indexing algorithm.

7.10.1 Distributed Gateways

Using real Active BAT data, a hypothetical environment described below is created that regenerates events of the Active BAT data. To create complete distributed environments, more issues need to be solved such as time synchronisation (see Section 7.10.4).

The current Active BAT system employs a centralised architecture, and all the data are gathered in the database. The Active BAT system, as described, is expensive to implement in that it requires large installations, and a centralised structure. The centralised structure allows for easy computation and implementation, because all distance estimates can be quickly shipped to a place where computational power is cheap. Moreover, the active mobile architecture facilitates the collection of multiple simultaneous distance samples at the fixed nodes, which can produce more accurate position estimates relative to a passive mobile architecture.

It is inherently scalable both in terms of sensor data acquisition and management as well as software components. However, when constructing real-time mobile ad hoc communications with resource-constrained devices, distributed coordination must be supported, so that mobile device users can promptly subscribe to certain information. It is simulated that each room and corridors

that hold gateway nodes (see location map Fig. 7.27), which are able to participate in event broker grids. The software design follows the service-oriented architecture described in Chapter 3. Thus, each local gateway node performs event filtering and correlation by registering the service that associates subscriptions with abstractions such as *Andy in room SN04*. These subscriptions are decomposed to the units operable by the event broker grid, where event correlation, aggregation and filtering are supported. The details for the high-level language for service composition are under development and out of scope for this dissertation. Communication among gateways is based on ECCO-PPS, described in Chapter 6.

The data is taken on May 20th 2003 for 24 hours. The total number of original events received by the ceiling units is around 400,000, and a sample is shown in Fig. 7.28. On average, a sighting gets around 10 receptions, of which perhaps 2 will be 100% noise, 2 will be too noisy and will be rejected and the rest will be used for the final estimate. After the position calculation, the total number of around 200,000 events are created (Fig. 7.29). This shows BAT data after the location of the BAT is calculated, which consists of timestamp, user, area, coordination (X, Y, Z) and orientation. In this experiment, this is the input stream data to the distributed gateway. Thus, gateway nodes are assumed to perform the trilateration computation of the BAT location from the raw data. Gateways are event brokers and form a distributed publish/subscribe system, where subscribers register the composite events, and, based on subscription information, event brokers distribute relevant events to the target brokers. The experiment uses the BAT data as input and produces the deterministic result.

```
---------- Position Start
TIME: [02 0c 30 bb fa c5]
DABR: 2 1000.582092 1044.230957 2.320667 31052.382812 1.302316 1 -
DABR: 22 999.148926 1043.030518 2.319667 4677.762695 2.356863 1 -
DABR: 23 999.549744 1044.223877 2.319667 2388.645020 2.217386 1 -
DABR: 24 999.149475 1045.423706 2.323667 4777.290039 1.539556 1 -
DABR: 24 999.149475 1045.423706 2.323667 3383.913574 2.123533 2 -
Temperature: 27Curtailed: 0
RESULT: 0 1000.422546 1045.085571 1.182180 0.673943 1.631099 1.966668 0.511598 00 11

TIME: (UNIX TIME in hex)
DABR: (Receiver chain)(Rec x pos)(Rec y pos)(Rec z pos)(amplitude)(range)(set)(state)
RESULT: (error flag)(x)(y)(z)(error)....
```

Figure 7.28: Original Raw Data

```
1  30408.802618,10,SN09,1002.335327,1033.320801,1.261441,-22.443605
2  30408.856115,10,SN09,1002.520386,1033.289429,1.251856,-20.335649
3  30409.063099,10,SN09,1002.533203,1033.279297,1.285185,-20.326197
4  30409.112594,10,SN09,1002.732910,1033.234863,1.270585,-22.712467
5  30409.315079,10,SN09,1002.921448,1033.175903,1.271525,-54.598316
6  30409.370575,10,SN09,1002.994690,1033.126587,1.283121,-56.499645
7  30409.564561,10,SN09,1003.170227,1033.044556,1.298443,-52.581676
```

Figure 7.29: Gateway Input Data

7.10.2 Durative Event Efficiency

Several event correlations are performed. Among them, the use of durative events is shown below. Fig. 7.30 depicts the number of events over the local gateway nodes without durative events, while Fig. 7.31 shows the same operation with durative event compositions. During this experiment, 21 BATs participated. The result shows a dramatic reduction of event occurrences by

using durative events, where the state of the target object is maintained. Specific durative events can be obtained by a composite event $E_{location}$ using *spatial restriction* or E^* using *iteration*. There are no specific subset policies for the consumption of events specified in this experiment.

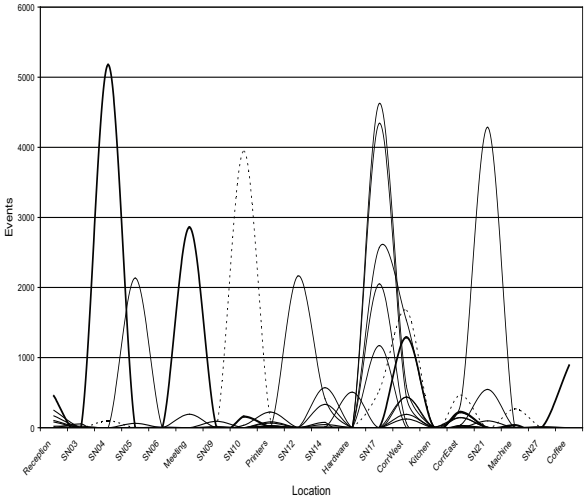


Figure 7.30: Events over Locations

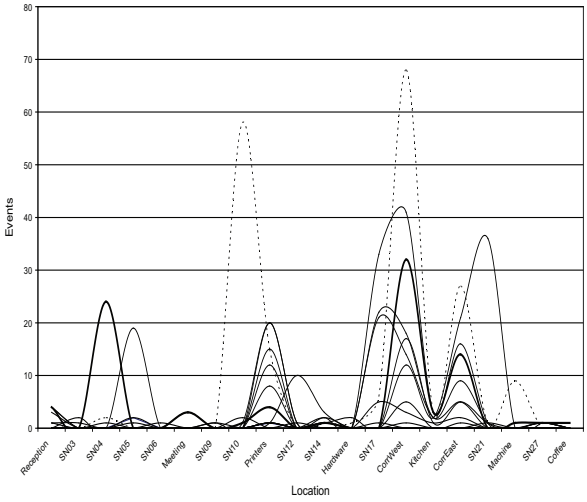


Figure 7.31: Durative Events over Locations

Fig. 7.32 and Fig. 7.33 depict the events identified on the BAT holders Andy and Brian. Andy's office (room SN04) is most likely the location where the highest number of events recorded. The numbers corresponding to the location in Fig. 7.27. Fig. 7.32 and Fig. 7.33 show the events over the location, but they do not indicate when they occurred.

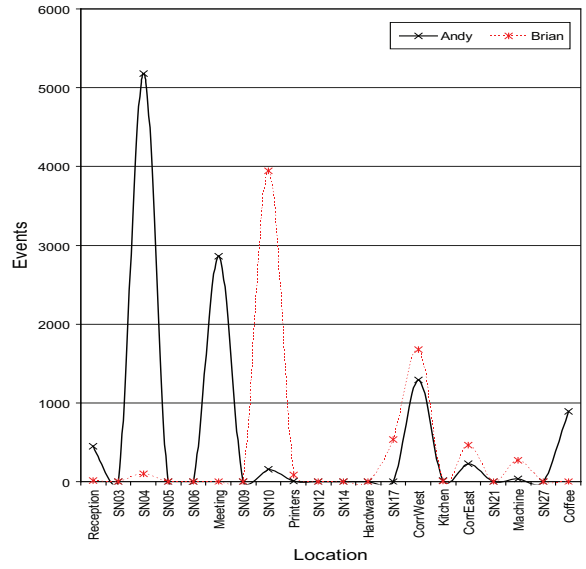


Figure 7.32: Events (Andy, Brian)

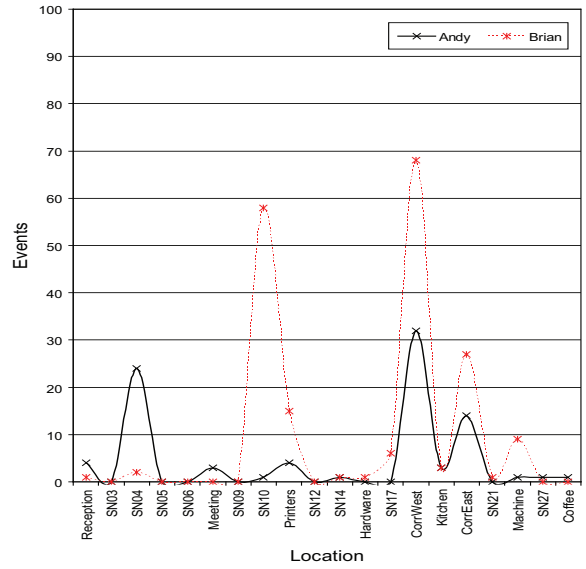


Figure 7.33: Durative Events (Andy, Brian)

Fig. 7.34 and Fig. 7.35 depict the events over the time-line (24 hours). Most activities are recorded during day time. Durative event composition over the time-line (24 hours) shows a significant reduction of the number of events.

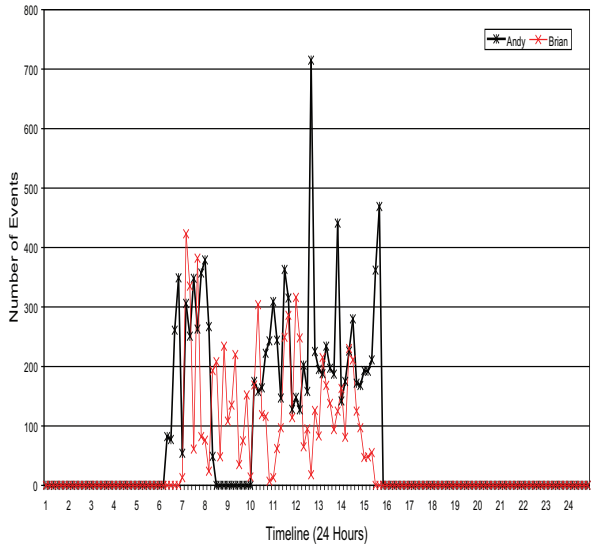


Figure 7.34: Events over Time

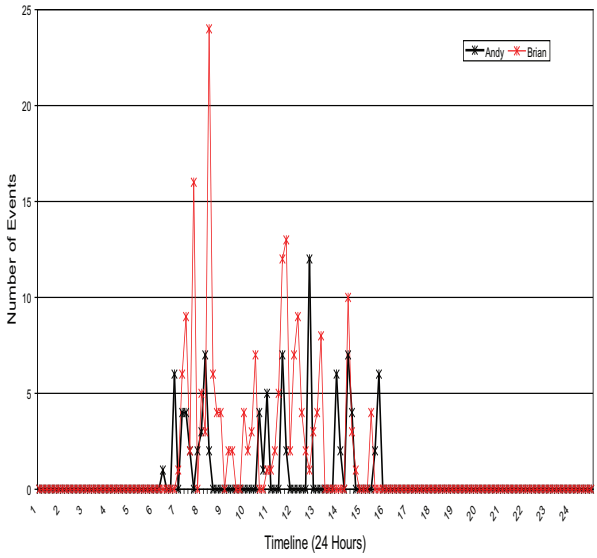


Figure 7.35: Durative Events over Time

7.10.3 Event Correlation

Fig. 7.36 traces Andy and Brian over time and location between time units 1500 and 3300. One unit is 15 seconds, and 7.5 hours of activities are shown. It looks like Andy and Brian spent much time in location 8 (room SN10), Brian's office.

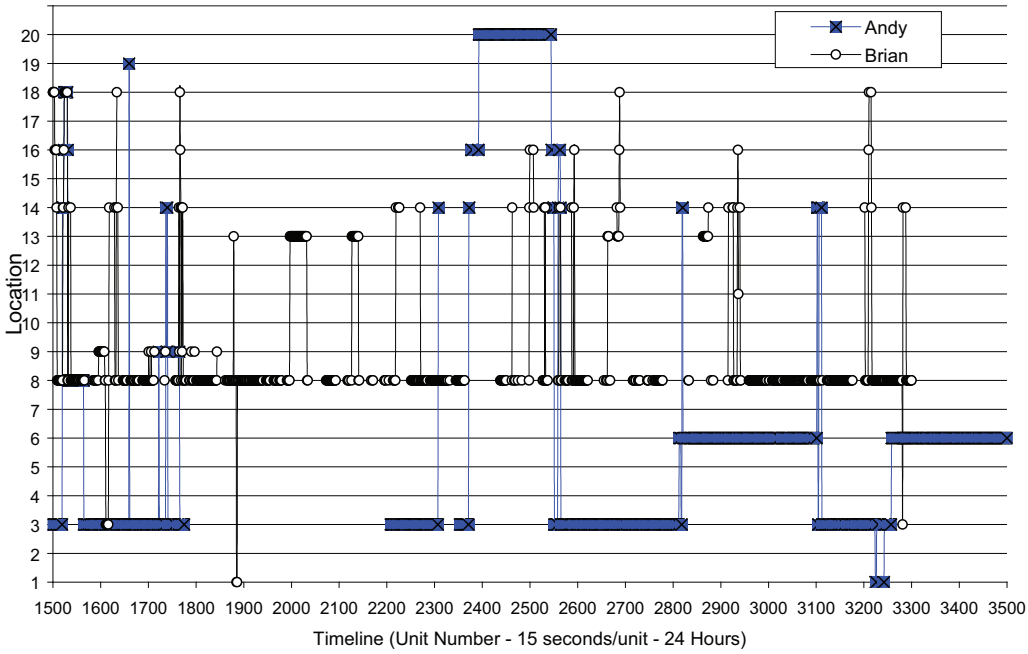


Figure 7.36: Events over Location and Time

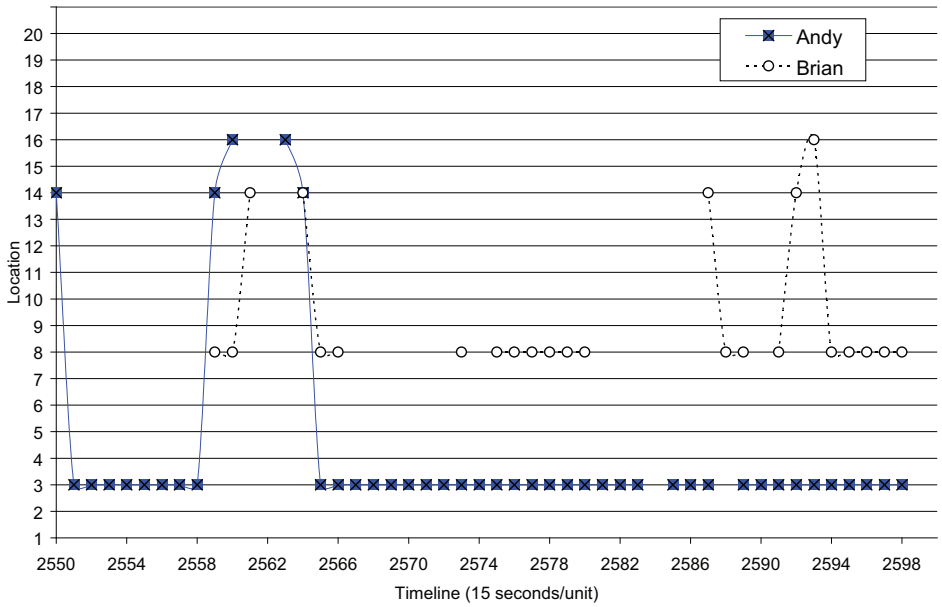


Figure 7.37: Composite Event - $(Brian; Andy)_{corrwest}$

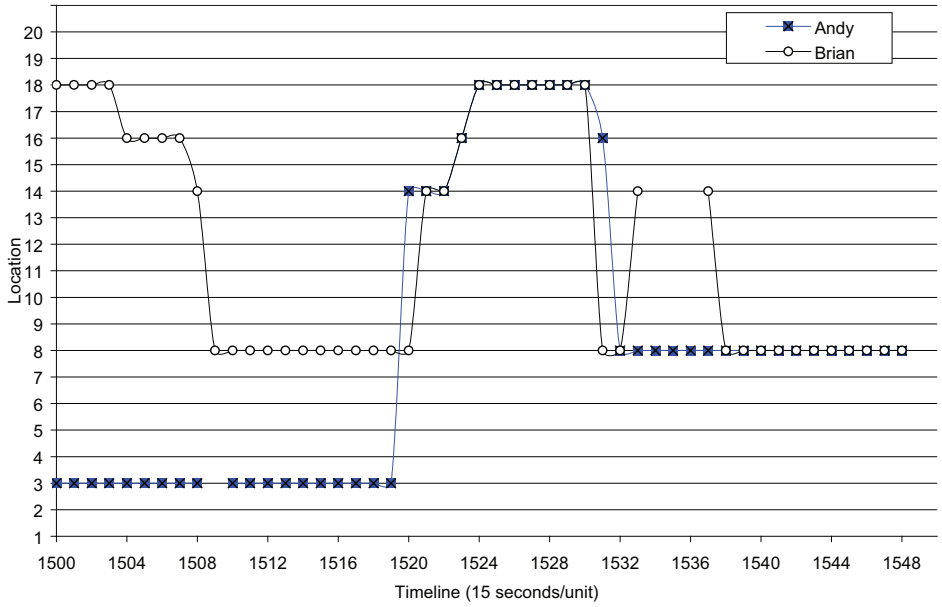


Figure 7.38: Composite Event - $(Andy + Brian)_{machine}$

Fig. 7.37 shows a specific period, when they were positioned in *corridor west*. The composite event $(Brian; Andy)_{corrwest}$ is detected at time unit 2564.

In Fig. 7.38, the detection of composite event $(Andy + Brian)_{SN25(machine)}$ is shown at the time unit between 1523 and 1529. A local gateway can detect this correlation, if the composite event is subscribed to through the event broker. This composition could be a part of services provided by the service grid.

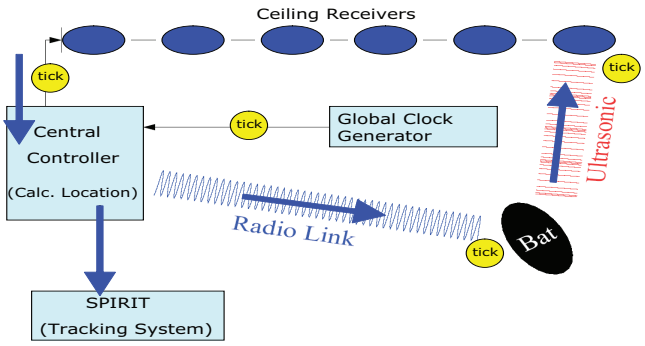


Figure 7.39: Time Synchronisation Mechanism in Active BAT and SPIRIT

7.10.4 Temporal Ordering in the Active BAT System

The applications of Active BAT require high accuracy and real-time object tracking. The Active BAT system are wired with a centralised controller. It therefore does not encounter problems of time synchronisation and coordination amongst beacons. The timestamp is derived from a Global Clock Generator (GCG), which is a hardware clock that sends *ticks* to every component of the system over a serial link. When a location is computed, the location message is timestamped using GCG. In general, the GCG delay is in the order of microseconds, and the slowest part of the system is the bit that waits for ultrasound to propagate (speed of sound) after a position is requested but before it can be calculated. This delay is used to measure the distance between the BAT and the receiver at the ceiling. Once the location is calculated, the message has to travel up to SPIRIT (order of milliseconds), and the event will be generated. There is therefore a delay from real-time event occurrence until an event is generated and given a timestamp. When gateways are distributed, temporal ordering of events requires more complex time synchronisation. The implementation of temporal ordering mechanisms described in Section 7.7.5 is desirable. The current experiment assumes that all timestamps are properly synchronised.

7.11 Summary and Outlook

This chapter has shown various investigations of event filtering, correlation, and aggregation as part of publish/subscribe functionality. I introduce generic composite event semantics with interval-based semantics for event detection. This precisely defines complex timing constraints among correlated event instances.

Aggregating data/events in ubiquitous computing requires management of stateful events over the network. The described durative event model, combined with interval-semantics, provides a new vision for data processing in ubiquitous computing. The sensed data must be aggregated and integrated into higher-level information or knowledge at appropriate points, while flowing over heterogeneous networks. The ultimate goal is to extend the functionality of simple publish/subscribe filters to enable stateful subscriptions, parametrisation, and computation of aggregates, while maintaining high scalability.

My main contribution is a novel algebra for composite events, which supports more expressive and stateful subscriptions including definition of complex temporal ordering. This correlation function can be integrated within the context of communication mechanisms to support ubiquitous computing. The work presented in this chapter is an essential step towards this goal. Future work will include formalisation of semantics, transformation of algebra, and high-level language definition.

8

Conclusions and Future Work

The focus of this dissertation is data centric networking in global distributed ubiquitous computing, which relies on content addressing instead of host-based addressing. A symmetric communication paradigm between publishers and subscribers is also focused, where an event is disseminated based on the rules and conditions not only based on subscriptions. The symmetric publish/subscribe paradigm brings another level to the data centric paradigm, and this new paradigm leads to a fundamental change in the functionality at the network level of asynchronous group communication and membership maintenance. Providing network independence for applications can thus be achieved. Network independence, through data centric networking, makes it easier to develop robust applications that are resilient to network dynamics. Publish/subscribe asynchronous group communication realises the vision of data centric networking that is particularly important for supporting mobile clients over heterogeneous wireless networks.

Recent evolution in WSNs has opened up a new dimension of data processing to ubiquitous computing, where sensors are used to gather high volumes of data of different types, to feed as context to a wide range of applications. This data processing requires novel mechanisms that make intelligent use of multiple simple data to create meaningful information. Complex data management must be integrated within the context of data centric networking.

This dissertation focuses on the following specific topics among the challenging issues in data centric communication. Ultimately all of them assist the design of distributed ubiquitous computing systems over heterogeneous networks.

- Event and query modelling for multidimensional events.
- Efficient event filtering with hypercube indexing in content-based publish/subscribe.
- ECCO-PPS: publish/subscribe by context adaptive controlled flooding in ad hoc networks.
- Unified event correlation semantics.

The dissertation began with an overview of publish/subscribe systems and wireless mobile computing including recent progress in WSNs. The outline of the vision of distributed system architecture in ubiquitous computing followed, where event-based computing (i.e., reactive pro-

gramming) will establish the backbone of system design. Brief explanation of service oriented architecture (SOA) is presented, where a P2P paradigm is the basis of interactions among components, and I identified the issues involved. This highlighted specific subjects that were tackled in this dissertation.

Firstly, particular emphasis was placed on the design of the event itself. Besides the existing attributes of events, continuous context information such as time or geographic location is incorporated within an event description. Semantics and representation of event and query will be even more important in future ubiquitous computing, where events flow over heterogeneous networks. I have shown that multidimensional event indexing with an RTree enables efficient indexing, matching, and scalability in publish/subscribe systems. Applications are dependent on XML and RDF for data representation now, and there is a growing need for efficient data centric routing protocols that can support query language expressiveness. The current event indexing structure used in this dissertation will need more compact encoding and lightweight implementation in the future. There have been efforts to optimise multidimensional data structures and many variants have been reported mainly from database research. It will be important to combine the research on database and publish/subscribe middleware.

A sensor captures a state change in the real world, and time critical event notification is set to become a crucial building block in future ubiquitous systems. Furthermore, a semantic event model will be required to provide a flexible interaction mechanism for open distributed and heterogeneous event-based applications. Unified access and interpretation of data are difficult, and correct data handling requires a precise understanding of the underlying meaning and proper interpretation of information. An unambiguous event model is the foundation for a higher level-addressing model for event dissemination.

Secondly, I integrated the hypercube event filter with a typed content-based publish/subscribe system (i.e., Hermes) for improvement of event filtering processes. I attempted to exploit a hashing mechanism to preserve locality so that type name and string attributes can take advantage of a multidimensional data structure, which includes hierarchical types. More work will be required on this.

A ubiquitous computing environment requires service advertisement and discovery in wireless networks, in a spontaneous and ad hoc fashion, and aims at providing ubiquitous connectivity and network centric services. An important aspect of such networks is that they can be deployed on demand with minimal planning and management, and with little dependency on existing network infrastructures.

As a primary focus, I investigated a structureless asynchronous group communication system over hybrid wireless ad hoc networks (i.e., ECCO-PPS). I presented context adaptive casting using restricted flooding with a cross layer approach between middleware and the network layer. Asynchronous group communication (many-to-many aka publish/subscribe) is integrated into mobile ad hoc networks. The main idea can apply over different types of networks such as wireless mesh. A data centric communication abstraction over heterogeneous wireless networks will have substantial impact for constructing reactive distributed applications.

Controlled flooding, policy based routing, and parameterised, gossip-based routing address similar issues; these are key approaches to realise more efficient routing based on the various contexts. I demonstrated controlled flooding as an early attempt in this area. DHT may not work well in dynamic MANET environments, but using super-peers to create an overlay network on top of dynamic epidemic based networks (e.g., ECCO-PPS) enable the addition of various functions (e.g., storage, directory). Mobile ad hoc networks will not be stand-alone, and an integration

with wireless mesh, DTN, and mobile agents to create further hybrid and heterogeneous environments will necessitate this multi-layered approach. Mobility and reliability issues were also studied to improve event dissemination mechanisms. ECCO-PPS also addresses dynamic group membership including two aspects: content-based addressing and symmetric group formation (e.g., based on location).

To add an additional dimension to event processing in global computing, understanding event aggregation, filtering and correlation are important. I worked on event correlation as part of publish/subscribe functionality. Temporal ordering of events is essential for event correlation over distributed systems, but it is challenging to realise it for heterogeneous wireless network environments. I introduced generic composite event semantics with interval-based semantics for event detection. This precisely defines complex timing constraints among correlated event instances. The sensed data must be aggregated and combined into higher-level information at appropriate points, while flowing over heterogeneous networks. Complex temporal and spatial relationships among correlated events from heterogeneous network environments must be addressed and coordinated.

This dissertation addresses the second wave of ubiquitous computing, where high volumes of sensor data will be flooded in everyday life. It provides a view of global architecture, and identifies event-based reactive communication as the key element to pursue. To realise content-based publish/subscribe for such pervasive data, use of multidimensional indexing is an effective solution for query/subscription processing. This work represents a step towards accomplishing a universal content-based filtering and routing network. Part of work described in this dissertation focused on particular wireless networks, however most of work presented in this dissertation can be applied not only wireless network environments but wired network environments.

In database research, stream processing is becoming more distributed. This addresses similar issues to composite event detector allocation over networks. It is necessary to explore the commonalities among event processing, stream processing, and filtering, and to determine how to combine their strengths. Static database stream processing now seems to correspond to the distributed messaging, which evolved in the form of publish/subscribe.

Network technologies aim to extend the capability of the Internet into large-scale interconnected networks such as DTN, WSNs, MANETs, and any future developments. The issues addressed in this dissertation should help to create a comprehensive software platform for extending Internet-based applications to a wide range of network-edge (pervasive) devices that can enable end-to-end solutions. The thesis's main theme shows how data centric networking could be integrated with other developments in the hope of making an important contribution to future ubiquitous computing.

8.1 Future Work

A wide range of exciting subjects are now emerging in ubiquitous computing research. The work presented in this dissertation provides a first step into the emerging field of data centric networking. Major research challenges in this area are information diffusion over wireless mobile network environments, especially event aggregation, filtering, and correlation over networks along selective routing. A next focus will be on WSN-specific data processing, integrated with global computing. Fig. 8.1 shows an experiment to observe the changes of temperature and light in our garden to follow conditions when nobody is around. Five motes connect to collect and deliver data towards the computer inside the house. The dynamic propagation of high-level knowledge to individual motes is work-in-progress. The following examines possible future research directions

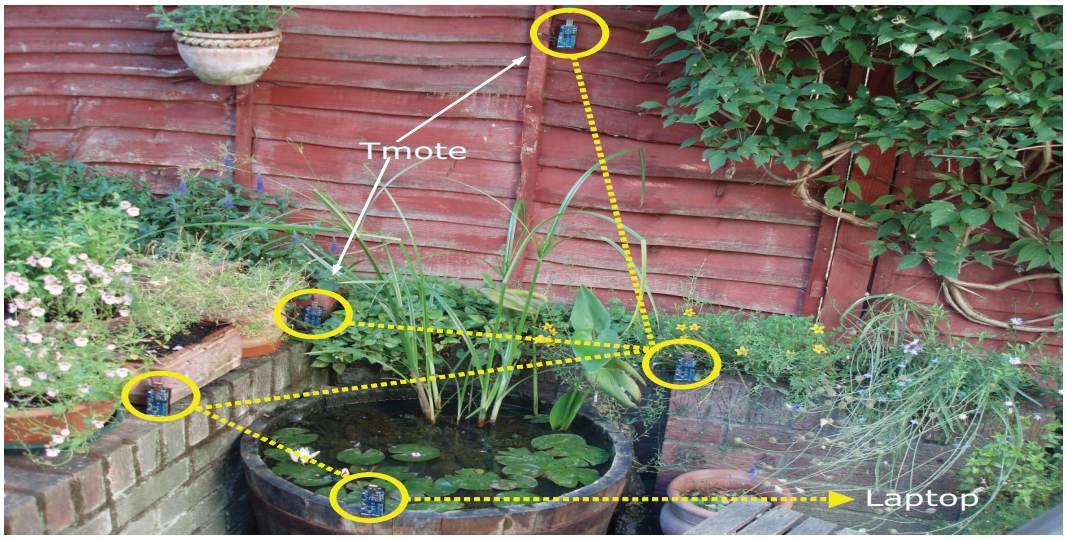


Figure 8.1: 5 Tmote Sensing Temperature and Light

for the next steps.

8.1.1 Multidimensional Indexing

Various multidimensional data structures have been proposed, mainly to support moving object types for databases. RTree has several variants and UB-Tree [Bay96] with the use of z-curves seems to provide improved performance. The complementary CAN's approach in P2P indexing places a key impact on high performance store-and-forward communication. RETree [CRW01] is a regular expression version of RTree and a good indexing structure to support event filtering. Transformation mechanisms such as a feature extraction process to reduce the number of dimensions may be useful. Note that compact, lightweight versions of these trees are important to support resource-constrained devices.

8.1.2 Fuzzy Semantic Query

Selective event dissemination requires further consideration of event matching. Accepting uncertainty as part of the whole system is essential when modelling the real world in ubiquitous computing, particularly in areas where human judgement, evaluation, and opinions are important factors. Fuzziness is often used to express the uncertainty or vagueness concerning the semantic meaning of events, phenomena, or statements. The power of range matching may not be sufficient for many applications, as users may not have the knowledge to formulate precise queries. It is desirable to apply fuzzy logic to model publish/subscribe including approximate matching for uncertainty of wireless sensor data. This issue links to group membership management for group communication.

Current search engines such as Google use keyword-based queries to help users find relevant information. Traditional tools assume that the parameters of a query model represent exactly the features of the modelled objects. However, some query processes are uncertain and hard to express in the form of traditional query languages. This may be problematic for two reasons: 1) the real situation is vague and a query cannot be described precisely, and 2) a complete description of a query object requires more detailed knowledge than one can expect from non-expert users. This changes the way we issue and model a query, when query tasks without

explicit keywords cannot be translated into clear forms.

8.1.3 Semantic Data Model

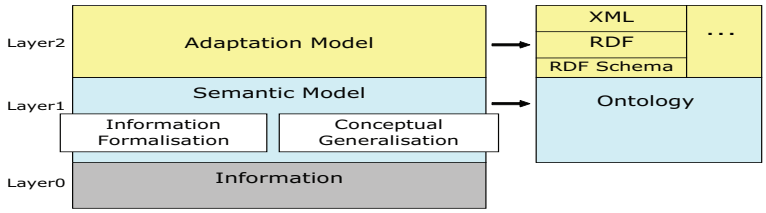


Figure 8.2: Ontology-Based Event Model

Unified access and interpretation of data on a common interpretation basis are rarely possible because of the different modelling assumptions at the data sources. Cultural, political or commercial context determine the assumptions made when interpreting the data. In most cases, these fundamental assumptions are not explicitly stated. Applications require the interpretation of information to be correct, whatever its origin and place of consumption. This can be achieved by supporting ontologies that provides the basis for correct data and event interpretation. Fig. 8.2 depicts an overview of an ontology-based event model. Through the RDF and web ontology language (OWL), the syntax can be well defined. These standards are based on XML, so that existing tools can be used. For example, RDF and RDFS [L$^+$99] [BG03] can be used to annotate data on the Web and provide the ways by which computer systems can exchange and communicate through them. All resources have unique resource identifiers (URIs plus anchor ids). Annotations about resources are based on various schemata that are defined in RDFS and are stored in RDF. Distributed annotations can be added to the metadata.

In [YB04c], I created a P2P grid over both wired and wireless networks (i.e., Hermes and ECCO-PPS). RDF [L$^+$99] is used to add semantics to the event and experiment with the possibility of federation via a semantic event model.

8.1.4 High Level Language for Event Correlation Semantics

I have described the semantics of event correlation with a parameterised event algebra. A high level language is desirable, which may combine subscription and correlation languages. Transformation of correlation semantics to $\pi - calculus$ can automatically provide the capability to compile into Java code or other conventional programming languages from the definition. Formalisation and transformation will help to create implementable, reliable, and QOS-guaranteed event detection in various resource equipped devices.

8.1.5 Reliability

I discussed issues for reliability of event delivery, while the current ECCO-PPS takes a best effort approach. Gossip-based reliability can be added for mission critical applications.

8.1.6 Programmable Networks

The majority of programming for WSNs follows a component-based event-driven programming model, where an event triggers the operation to migrate data from one node to another. TAG[MFH$^+$02] considers WSNs as a database and provides a high-level SQL-like language for queries. Programmability of sensor networks has got attentions recently. [Nag02] introduces a

high-level language inspired by biology, where program agents are considered as cells to form a globally specified shape via local computation and communication.

Ubiquitous computing infrastructures require software technologies that enable ad hoc assemblies of devices to spontaneously form a coherent group of cooperating components. This is challenging when individual components are heterogeneous in nature and engage in complex activity sequences to achieve a user goal. Currently, the interaction between the components of these environments is carefully hard-coded. Most sensor network applications are implemented as complex, low level programs that specify the behaviour of individual sensor nodes. WSNs need to organise themselves from components built by different applications. Programming for WSNs raises two main issues; programming abstractions and programming support. The former focuses on providing programmers with abstractions of sensors and sensor data. The latter provides additional runtime mechanisms that simplify program execution. Programming abstractions fall into two categories: application level and system level. The former defines and manipulates events at the desired level of semantic abstraction (e.g., latest position of target). The latter precisely specifies distributed computation and communication (e.g., apply f(x) to all x, send d to the 10 nearest nodes). The tradeoffs between these two models are expressiveness, efficiency, reusability and automation. The programmable network research area is still at an early stage and widely open to new developments.

8.1.7 Security

Wireless networks are becoming more pervasive and devices are becoming more programmable, therefore facilitating malicious and selfish behaviour. A ubiquitous application may involve collaboration between ad hoc groups of members. There are complex issues in trust among members. Based on predefined trust, recommendations, risk evaluation and experience from past interactions, an entity may derive new trust metrics to use as a basis for authorisation policies for access control [CGS+03]. This raises concerns about privacy and freedom of action. While providing location information can clearly be a one-way system, where the location providing tools do not track who is receiving; once your device receives the information, your location is potentially available to others. This makes systems in which clients learn their location without centralised tracking particularly appealing (e.g., [PCB00]). An important research problem is the fusion and synchronisation of location information obtained from multiple sources (e.g., GPS, dead-reckoning, proximity sensors, scene analysis). Fusion is required to increase the accuracy of location information and its availability in variable terrain conditions.

8.1.8 Formalisation of Publish/Subscribe System

Publish/subscribe is an asynchronous communication paradigm for the interconnection of distributed components (e.g., agent, broker). It is a promising solution for applications in distributed, mobile and dynamic environments, because of the decoupling among components. This is especially true for ubiquitous computing, where wireless sensor nodes are part of interaction components. However, the design and validation of these systems remains an open problem. It is easy to design individual components and validate them, but it is challenging to understand the global view of the behaviour of components. Formal modelling and specification of components and their interactions are desirable, including the underlying communication mechanism (i.e., publish/subscribe). Therefore, a methodology for verification of more complex publish/subscribe systems based on a real-world scenario is desirable.

Bibliography

[AAB+05] D. J. Abadi, Y. Ahmad, M. Balazinska, et al. The design of the borealis stream processing engine. In *Proc. CIDR*, pages 277–289, 2005. (pp 29, 43, & 177)

[ABB+03] A. Arasu, B. Babcock, S. Babu, et al. Stream: The stanford stream data manager. In *Proc. ACM SIGMOD*, 2003. (pp 29, 42, 43, & 177)

[ABKM02] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris. Resilient overlay networks. In *Proc. ACM SOSP*, 2002. (pg 116)

[ABW02] A. Arasu, S. Babu, and J. Widom. An abstract semantics and concrete language for continuous queries over streams and relations. Technical Report 2002-57, Stanford University, 2002. (pp 42, 155, & 159)

[AC03] R. Adaikkalavan and S. Chakravarthy. SnoopIB: Interval-based event specification and detection for active databases. *Advances in Databases and Information Systems*, 2798, 2003. (pg 160)

[ACC+03] D. J. Abadi, D. Carney, U. Cetintemel, M. Cherniack, et al. A new model and architecture for data stream management. *VLDB Journal*, 12(2), 2003. (pg 29)

[ADKM92] Y. Amir, D. Dolev, S. Kramer, and D. Malki. Membership algorithms for multicast communication groups. In *Proc. WDAG*, 1992. (pg 114)

[AGK+01] M. Adler, Z. Ge, J. F. Kurose, D. Towsley, and S. Zabele. Channelization problem in large scale data dissemination. In *Proc. ICNP*, 2001. (pg 79)

[ALJ02] G. Ashayer, H. K. Y. Leung, and H. A. Jacobsen. Predicate matching and subscription matching in publish/subscribe systems. In *Proc. ICDCS*, pages 539–548, 2002. (pg 25)

[All83] J. Allen. Maintaining knowledge about temporal intervals. *CACM*, 26(11), 1983. (pp 47, 160, & 164)

[AMW01] H. K. Ahn, N. Mamoulis, and H. M. Wong. A survey on multidimensional access methods. Technical report, Utrecht University, 2001. (pg 53)

[Ani] Nam:Network Animator. http://www.isi.edu/nsnam/nam/. (pg 132)

[AS03] J. Aspnes and G. Shah. Skip graphs. In *Proc. ACM-SIAM SODA*, 2003. (pg 30)

[Asi04] Media Lab Asia. *DakNet, Rural WiFi.* http://www.medialabasia.org/, 2004. (pg 111)

[ASS+99] M. K. Aguilera, R. E. Strom, D. C. Sturman, et al. Matching events in a content-based subscription system. In *Proc. PODC*, pages 53–61, 1999. (pp 25, 43, 52, & 53)

[AT05] I. Aekaterinidis and P. Triantafillou. Internet scale string attribute publish/subscribe data networks. In *Proc. ACM CIKM*, 2005. (pg 80)

[AWD04] M. Abolhasan, T. Wysocki, and E. Dutkiewicz. A review of routing protocols for mobile ad hoc networks. *Ad Hoc Networks*, 2, 2004. (pg 103)

[AWW05] I. F. Akyildiz, X. Wang, and W. Wang. Wireless mesh networks: A survey. *Elsevier Computer Networks Journal*, 47, 2005. (pp 14, 28)

[B+04] A. Berglund et al. *XML Path Language (XPath) 2.0. Working Draft.* http://www.w3c.org/TR/xpath20/, 2004. (pg 122)

[Bay96] R. Bayer. The universal B-tree for multidimensional indexing. Technical Report TUM-I9637, Technische Universitat Munchen, 1996. (pp 53, 192)

[BBC+05] R. Baldoni, R. Beraldi, G. Cugola, M. Migliavacca, and L. Querzoni. Structure-less content-based routing in mobile ad hoc networks. In *Proc. ICPS*, 2005. (pg 109)

[BBH+95] J. Bacon, J. Bates, Richard Hayton, et al. Using events to build distributed applications. In *Proc. IEEE SDNE*, pages 148–155, 1995. (pp 21, 23, 24, 44, & 157)

[BBR+05] A. Bharambe, R. Bharambe, S. G. Rao, et al. The impact of heterogeneous bandwidth constraints on DHT-based multicast protocols. In *Proc. IPTPS*, 2005. (pg 75)

[BCG04] S. Baehni, C. S. Chhabra, and R. Guerraoui. Mobility friendly publish/subscribe. Technical report, EPFL, 2004. (pg 109)

[BCM+99] G. Banavar, T. D. Chandra, B. Mukherjee, J. Nagarajarao, R. E. Strom, and D. C. Sturman. An efficient multicast protocol for content-based publish-subscribe systems. In *Proc. ICDCS*, pages 262–272, 1999. (pp 24, 76)

[BCT01] J. Boleng, T. Camp, and V. Tolety. Mesh-based geocast routing protocols in an ad hoc network. In *Proc. IPDPS*, pages 184–193, 2001. (pg 104)

[BD02] F. Babich and L. Deotto. Formal methods for specification and analysis of communication protocols. *IEEE Communication Survey and Tutorials*, 2002. (pg 155)

[BEK+05] C. Barrett, S. Eidenbenz, L. Kroc, et al. Probabilistic multi-path vs. deterministic single-path protocols for dynamic ad-hoc network scenarios. In *Proc. ACM SAC*, 2005. (pp 110, 111, & 139)

[Ben75] J. L. Bentley. Multidimensional binary search trees used for associative searching. *CACM*, 18(9), 1975. (pg 30)

[Bet01] C. Bettstetter. Smooth is better than sharp: A random mobility model for simulation of wireless networks. In *Proc. MSWiM*, pages 19–27, 2001. (pg 113)

[Bez74] J. C. Bezdek. Numerical taxonomy with fuzzy sets. *Journal of Mathematical Biology*, 1974. (pg 146)

[BG03] D. Brickley and R.V. Guha. *RDF vocabulary description language 1.0: RDF Schema*. http://www.w3.org/TR/rdf-schema/, 2003. (pp 79, 193)

[BGS00] P. Bonnet, J. Gehrke, and P. Seshadri. Querying the physical world. *IEEE Pervasive Communication*, 7(5), 2000. (pp 42, 115)

[BGS01] P. Bonnet, J. Gehrke, and P. Seshadri. Towards sensor database systems. In *Proc. MDM*, 2001. (pg 40)

[BHO+98] K. Birman, M. Hayden, O. Ozkasap, et al. Bimodal multicast. Technical Report TR-98-1664, Cornell University, 1998. (pg 115)

[Bia67] T. Bially. *A Class of Dimension Changing Mapping and Its Application to Bandwidth Compression*. PhD thesis, Polytechnic Inst. of Brooklyn, 1967. (pg 78)

[BKSS90] N. Beckmann, H.-P. Kriegel, R. Schneidar, and B. Seeger. The R*-Tree: An efficient and robust access method for points and rectangles. In *Proc. ACM SIGMOD*, pages 322–331, 1990. (pg 53)

[BLB02] A. Beaufour, M. Leopold, and P. Bonnet. Smart-tag based data dissemination. In *Proc. WSNA*, 2002. (pg 111)

[BLMT99] E. Bommaiah, M. Liu, A. McAuley, and R. Talpade. *AMRoute: Ad hoc Multicast Routing Protocol*. Internet Draft, draft-manet-amroute-00.txt, 1999. (pg 29)

[Blo70] B. Bloom. Space/time trade-offs in hash coding with allowable errors. *CACM*, 13(7), 1970. (pp 120, 123, & 124)

[BM98] R. Bayer and V. Markl. The UB-Tree: Performance of multidimensional range queries. Technical Report TUM-I9814, Technische Universitat Munchen, 1998. (pg 52)

[BM02] A. Broder and M. Mitzenmacher. Network applications of bloom filters: a survey. *Allerton*, 2002. (pg 123)

[BMB+00] J. Bacon, K. Moody, J. Bates, et al. Generic support for distributed applications. *IEEE Computer*, pages 68–77, 2000. (pg 31)

[BMJ+98] J. Broch, D. Maltz, D. Johnson, et al. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proc. ACM MobiCom*, 1998. (pg 139)

[BMVV05] R. Baldoni, C. Marchetti, A. Virgillito, and R. Vitenberg. Content-based publish/subscribe over structured overlay networks. In *Proc. ICDCS*, 2005. (pp 107, 109)

[BPS94] A. Borgida and P. Patel-Schneider. A semantics and complete algorithm for subsumption in the classic description logic. *Journal of Artificial Intelligence Research*, 1994. (pg 26)

[BV05] R. Baldoni and A. Virgillito. Distributed event routing in publish/subscribe communication systems: a survey. Technical Report 15-05, Universita di Roma, 2005. (pg 105)

[Car98] A. Carzaniga. *Architectures for an Event Notification Service Scalable to Wide-area Networks*. PhD thesis, Politecnico di Milano, 1998. (pg 51)

[Cas03] M. Castro. Scalable application-level anycast for highly dynamic groups. In *Proc. NGC*, 2003. (pg 74)

[CBD02] T. Camp, J. Boleng, and V. Davies. A survey of mobility models for ad hoc network research. *Wireless Communication and Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking Research, Trends and Applications*, 2(5), 2002. (pg 113)

[CBRP05] I. D. Chakeres, E. Belding-Royer, and C. Perkinsand. *Dynamic MANET On-demand Routing Protocol, IETF Draft, draft-ietf-manet-dymo-03.txt*. IETF, 2005. (pg 28)

[CCC+02] D. Carney, U. Cetinternel, M. Cherniack, et al. Monitoring streams: A new class of data management applications. In *Proc. VLDB*, pages 215–226, 2002. (pp 42, 43)

[CCD+03] S. Chandrasekaran, O. Cooper, A. Deshpande, et al. TelegraphCQ: Continuous dataflow processing for an uncertain world. In *Proc. Innovative Data System Research*, pages 269–280, 2003. (pp 29, 42, & 43)

[CCT03] C. Y. Chang, C. T. Chang, and S. C. Tu. Obstacle-free geocasting protocols for single/multi-destination short message services in ad hoc networks. *Wireless Networks*, 9(2), 2003. (pg 104)

[CCW03] M. Caporuscio, A. Carzaniga, and A. Wolf. Design and evaluation of a support service for mobile, wireless publish/subscribe applications. Technical Report CU-CS-944-03, University of Colorado, 2003. (pg 113)

[CDK+02] M. Castro, P. Druschel, A. Kermarrecothers, et al. Scribe: A large-scale and decentralized application-level multicast infrastructure. *Journal on Selected Areas in Communication*, 20, 2002. (pp 18, 21, 25, 72, 74, & 75)

[CDK+03] M. Castro, P. Druschel, A. M. Kermarrec, et al. Splitstream: High-bandwidth multicast in a cooperative environment. In *Proc. ACM Symposium on Operating Systems Principles*, 2003. (pg 75)

[CDW01] A. Carzaniga, J. Deng, and A. L. Wolf. Fast forwarding for content-based networking. Technical Report CU-CS-922-01, University of Colorado, 2001. (pg 24)

[CEK03] L. Christopher, S. J. Eidenbenz, and L. Kroc. Parametric probabilistic sensor network routing. In *Proc. WSNA*, 2003. (pp 110, 111, & 120)

[CFGR02] C. Y. Chan, P. Felber, M. N. Garofalakis, and R. Rastogi. Efficient filtering of XML documents with xpath expressions. In *Proc. ICDE*, pages 235–244, 2002. (pp 43, 122)

[CGG+05] C. Curino, M. Giani, M. Giorgetta, et al. TinyLIME: Bridging mobile and sensor networks through middleware. In *Proc. PerCom*, 2005. (pg 154)

[CGR02] C. Y. Chan, M. Garofalakis, and R. Rastogi. RE-Tree: An efficient index structure for regular expressions. In *Proc. VLDB*, 2002. (pg 70)

[CGS+03] V. Cahill, E. Gray, J. Seigneurothers, et al. Using trust for secure collaboration in uncertain environments. *IEEE Pervasive Computing*, 2(3), 2003. (pg 194)

[CGZ98] C. Chiang, M. Gerla, and L. Zhang. Forwarding group multicast protocol (FGMP) for multihop. *Mobile Wireless Networks: Cluster Comp, Special Issue on Mobile Computing*, 1(2), 1998. (pg 115)

[CHC+05] A. Chaintresu, P. Hui, J. Crowcroft, et al. Pocket switched networks: Real-world mobility and its consequences for opportunistic forwarding. Technical Report UCAM-CL-TR617, University of Cambridge, 2005. (pg 28)

[CHTV99] K. Obraczka C. Ho, G. Tsudik, and K. Viswanath. Flooding for reli- able multicast in multi-hop ad hoc networks. In *Proc. Wrokshop on DIAL-M*, pages 64–71, 1999. (pg 108)

[CIP02] M. Caporuscio, P. Inverardi, and P. Pelliccione. Formal analysis of clients mobility in the SIENA publish/subscribe middleware. Technical report, University of L'Aquila, 2002. (pg 29)

[CJ03] T. Clausen and P. Jacquet. *Optimized Link State Routing Protocol*. IETF RFC3626, 2003. (pp 28, 113)

[CL03a] T. Camp and Y. Liu. An adaptive mesh-based protocol for geocast routing. *Journal of Parallel and Distributed Computing: Special Issue on Routing in Mobile and Wireless Ad Hoc Networks*, 62(2), 2003. (pg 104)

[CL03b] J. Carlson and B. Lisper. An interval-based algebra for restricted event detection. In *Proc. FORMATS*, 2003. (pp 165, 167)

[CM96] S. Chakravarthy and De. Mishra. Snoop: An expressive event specification language for active databases. *Data Knowledge Engineering*, 14(1), 1996. (pp 31, 155, 158, 165, & 167)

[CM99] S. Corson and J. Macker. *Mobile Ad Hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations*. RFC 2501, 1999. (pp 100, 118)

[CMPC04] P. Costa, M. Migliavacca, G. P. Picco, and G. Cugola. Epidemic algorithms for reliable content-based publish-subscribe: An evaluation. In *Proc. ICDCS*, pages 552–561, 2004. (pg 106)

[CN01] G. Cugola and E. Nitto. Using a publish/subscribe middleware to support mobile computing. In *Proc. Workshop MDC*, 2001. (pp 26, 112, & 113)

[CNF98a] G. Cugola, E.D. Nitto, and A. Fuggetta. Exploiting an event-based infrastructure to develop complex distributed systems. In *Proc. ICSE*, 1998. (pp 24, 26, & 72)

[CNF98b] G. Cugola, E.Di Nitto, and A. Fuggetta. The JEDI event-based infrastructure and its applications to the development of the OPSS WFMS. *IEEE Trans. on Software Engineering*, 1998. (pp 23, 112)

[CNP00] G. Cugola, E. D. Nitto, and G. P. Picco. Content-based dispatching in a mobile environment. *Proc. WSDAAL*, 2000. (pg 29)

[CP05] P. Costa and G. P. Picco. Semi probabilistic content-based publish-subscribe. In *Proc. ICDCS*, 2005. (pg 106)

[CRB01] R. Chandra, V Ramasubramanian, and K. Birman. Anonymous gossip: Improving multicast reliability in mobile ad-hoc networks. In *Proc. ICDCS*, 2001. (pg 115)

[CRS01] Y. Chu, S. G. Rao, and S. Seshan. A case for end system multicast. In *Proc. ACM SIGMETRICS*, 2001. (pg 25)

[CRW01] A. Carzaniga, D. Rosenblum, and L. Wolf. Design and evaluation of a wise-area event notification service. *ACM Trans. on Computer Systems*, 19(3), 2001. (pp 18, 21, 24, 76, & 192)

[CRW04] A. Carzaniga, M. Rutherford, and A. Wolf. A routing scheme for content-based networking. In *Proc. IEEE INFOCOM*, 2004. (pg 79)

[CS95] P. Cheeseman and J. Stutz. *Bayesian classification (AutoClass): Theory and results. In Advances in Knowledge Discovery and Data Mining.* AAAI Press, 1995. (pp 146, 150)

[CS04] F. Cao and J. Singh. Efficient event routing in content-based publish-subscribe service networks. In *Proc. IEEE INFOCOM*, 2004. (pg 79)

[CSZ03] Y. Chen, K. Schwan, and D. Zhou. Opportunistic channels: Mobility-aware event delivery. In *Proc. ACM/IFIP/USENIX Middleware*, pages 182–201, 2003. (pg 29)

[CW03] A. Carzaniga and A. L. Wolf. Forwarding in a content-based network. In *Proc. ACM SIGCOMM*, 2003. (pp 21, 23)

[DAF+03] Y. Diao, M. Altinel, M. J. Franklin, et al. Path sharing and predicate evaluation for high-performance XML filtering. *ACM TODS*, 28(4), 2003. (pg 43)

[dBKOS98] M. de Berg, M. V. Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry-Algorithms and Applications.* Springer, 1998. (pg 53)

[Dee98] S. Deering. *Internet Protocol, Version 6 (Ipv6) Specification, RFC 2460.* IETF, 1998. (pg 37)

[Del05] F. Delmastro. From pastry to CrossROAD: Cross-layer ring overlay for ad hoc networks. In *Proc. Workshop on MP2P*, 2005. (pg 107)

[DFL01] J. A. Davis, A. H. Fagg, and B. N. Levine. Wearable computers as packet transport mechanisms in highly-partitioned ad-hoc networks. In *Proc. of IEEE ISWC*, 2001. (pg 112)

[DGH+87] A. Demers, D. Greene, C. Hauserothers, et al. Epidemic algorithms for replicated database maintenance. In *Proc. PODC*, pages 1–12, 1987. (pg 110)

[DGR04] A. Demers, J. Gehrke, and M. Riedewald. The architecture of the cornell knowledge broker. In *Proc. ISI*, 2004. (pg 30)

[DiC93] F. DiCesare. *Practice of Petri Nets in Manufacturing.* Chapman and Hall, 1993. (pg 156)

[DOT+96] C. Daws, A. Olivero, S. Tripakis, et al. *The tool kronos in Hybrid Systems III.* LNCS 1066. Springer, 1996. (pg 155)

[DPRM01] S. R. Das, C. E. Perkins, E. M. Royer, and M. K. Marina. Performance comparison of two ondemand routing protocols for ad hoc networks. *IEEE Personal Communications Magazine special issue on Ad hoc Networking*, 20:16–28, 2001. (pg 139)

[DPZ04] R. Draves, J. Padhye, and B. Zill. Comparison of routing metrics for static multihop wireless networks. In *Proc. ACM SIGCOMM*, 2004. (pg 139)

[DQA04] A. Datta, S. Quarteroni, and K. Aberer. Autonomous gossiping: A selforganizing epidemic algorithm for selective information dissemination in wireless mobile ad-hoc networks. In *Proc. IC-SNW*, 2004. (pg 110)

[DZD+03] F. Dabek, B. Zhao, P. Druschel, J. Kubiatowicz, and I. Stoica. Towards a common API for structured peer-to-peer overlays. In *Proc. Workshop on IPTPS*, 2003. (pg 107)

[EFGH02] P. Eugster, P. Felber, R. Guerraoui, and S. Handurukande. Event systems: How to have your cake and eat it too. In *Proc. Workshop on DEBS*, 2002. (pg 80)

[EFGK03] P. Eugster, P. Felber, R. Guerraoui, and A. Kermarrec. The many faces of publish/subscribe. *Computing Surveys*, 35, 2003. (pp 23, 48)

[EFGS00] P. Eugster, P. Felber, R. Guerraoui, and J. Sventek. Type-based publish/subscribe. Technical report, EPFL, 2000. (pp 23, 26)

[EG01] P. Eugster and R. Guerraoui. Hierarchical probabilistic multicast. Technical report, EPFL, 2001. (pg 114)

[EGH+01] P. Eugster, R. Guerraoui, S. B. Handurukandeothers, et al. Lightweight probabilistic broadcast. In *Proc. IEEE DSN*, 2001. (pg 115)

[ER02] J. Elson and K. Römer. Wireless sensor networks: A new regime for time synchronization. In *Proc. Hotnets-I*, 2002. (pg 172)

[Fal04] K. Fall. Messaging in difficult environments. Technical Report IRB-TR-04-019, Intel Research Berkeley, 2004. (pg 102)

[FB74] R. A. Finkel and J. L. Bentley. Quad trees: A data structure for retrieval on composite keys. *Acta Informatica*, 4, 1974. (pg 30)

[FGKZ03] L. Fiege, F. C. GÄartner, O. Kasten, and A. Zeidler. Supporting mobility in content-based publish/subscribe middleware. In *Proc. ACM/IFIP/USENIX Middleware*, pages 103–122, 2003. (pg 29)

[FHM+04] A. Ferscha, M. Hechinger, R. Mayrhoferand, et al. A light-weight component model for peer-to-peer applications. In *Proc. Mobile Distributed Computing*, 2004. (pg 34)

[Fis87] D. H. Fissher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 1987. (pg 146)

[FJL+01] F. Fabret, H. A. Jacobsen, F. Llirbat, J. Pereira, K. A. Ross, and D. Shasha. Filtering algorithms and implementation for very fast publish/subscribe systems. In *Proc. SIGMOD*, pages 115–126, 2001. (pg 43)

[FLPS00] F. Fabret, F. Llirbat, J. Pereira, and D. Shasha. Efficient matching for content-based publish/subscribe systems. Technical report, INRIA, 2000. (pg 49)

[FMB02] R. Fenk, V. Markl, and R. Bayer. Interval processing with the UB-tree. In *Proc. IDEAS*, pages 12–22, 2002. (pg 53)

[FRZ+05] R. Fonseca, S. Ratnasamy, J. Zhao, C.T. Ee, D. Culler, S. Shenker, and I. Stoica. Beacon vector routing: Scalable point-to-point routing in wireless sensor nets. In *Proc. ACM NSDI*, 2005. (pg 107)

[GA02] A. Galton and J. C. Augusto. Two approaches to event definition. In *Proc. DEXA*, pages 547–556, 2002. (pp 155, 165)

[GA04] R. Gomez and J. C. Augusto. Durative events in active databases. In *Proc. ICEIS*, 2004. (pg 164)

[GAA03] A. Gupta, D. Agrawal, and A. El Abbadi. Approximate range selection queries in peer-to-peer systems. In *Proc. CIDR*, pages 141–151, 2003. (pp 25, 43, & 78)

[Gar99] J. Garcia. A multicast routing protocol for ad-hoc networks. In *Proc. IEEE INFOCOM*, 1999. (pg 29)

[GD94] S. Gatziu and K. R. Dittrichothers. Detecting composite events in active database systems using Petri Nets. In *Proc. RIDE-AIDS*, 1994. (pp 31, 158)

[GEG+03] B. Greenstein, D. Estrin, R. Govindan, S. Ratnasamy, and S. Shenker. DIFS: A distributed index for features in sensor networks. In *Proc. Workshop on SNPA*, 2003. (pg 30)

[GEH02] D. Ganesan, D. Estrin, and J. Heidemann. Dimensions: Why do we need a new data handling architecture for sensor networks? In *Proc. ACM Workshop on Hot Topics in Networks*, pages 143–148, 2002. (pg 30)

[GG98] V. Gaede and O. Guenther. Multidimensional access methods. *ACM Computing Surverys*, 30(2), 1998. (pg 53)

[GHP02] M. Gerla, X. Hong, and G. Pei. *Landmark Routing Protocol (LANMAR) for Large Scale Ad Hoc Networks*. http://www.ietf.org/internet-drafts/draft-ietf-manet-lanmar-05.txt, 2002. (pg 103)

[GIM99] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *Proc. VLDB*, 1999. (pg 30)

[GJS92] N. H. Gehani, H. V. Jagadish, and O. Shmueli. Composite event specification in active databases: Model and implementation. In *Proc. VLDB*, pages 327–338, 1992. (pp 155, 158)

[GKMS01] A. C. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. J. Strauss. QuickSAND: Quick summary and analysis of network data. Technical Report 2001-43, DIMACS, 2001. (pg 40)

[GKP99] B. Gruber, B. Krishnamurthy, and E. Panagos. The architecture of the READY event notification service. In *Proc. ICDCS*, 1999. (pp 24, 158)

[GLAM99] J. J. Garcia-Luna-Aceves and E. Madruga. The core assisted mesh protocol. *IEEE Journal on Selected Areas in Communications*, 17(8), 1999. (pg 115)

[GLF89] J. H. Gennari, P. Langley, and D. Fisher. Models of incremental concept formation. *Artificial Intelligence*, 1989. (pg 146)

[GLM+04] D. K. Goldenberg, J. Lin, A. Morseand, et al. Towards mobility as a network control primitive. In *Proc. ACM MobiHoc*, 2004. (pg 112)

[GM01] R. Gold and C. Mascolo. Use of context-awareness in mobile peer-to-peer networks. In *Proc. Workshop on FTDCS*, 2001. (pg 119)

[GM03] C. Gui and P. Mohapatra. Efficient overlay multicast for mobile ad hoc networks wireless communications and networking conference. In *Proc. WCNC*, 2003. (pg 106)

[Gon01] L. Gong. Project JXTA: A technology overview. Technical report, Sun Microsystems, 2001. (pg 79)

[Gro] Open Grid Services Architecture Working Group. http://www.ggf.org/ogsa-wg/. (pg 34)

[Gry] Gryphon. http://www.research.ibm.com/gryphon/. (pg 72)

[GS99] S. Gupta, , and P. Srimani. An adaptive protocol for reliable multicast in mobile multi-hop radio networks. In *Proc. IEEE WMCSA*, 1999. (pg 115)

[GS04] J. Gao and P. Steenkiste. An adaptive protocol for efficient support of range queries in DHT-based systems. In *Proc. IEEE International Conference on Network Protocols*, 2004. (pg 78)

[GSAA04] A. Gupta, O. D. Sahin, D. Agrawal, and A. E. Abbadi. Meghdoot: content-based publish/-subscribe over P2P networks. In *Proc. ACM/IFIP/USENIX Middleware*, 2004. (pp 26, 31, 72, & 76)

[GT02] M. Grossglauser and D. Tse. Mobility increases the capacity of ad-hoc wireless networks. *IEEE/ACM Trans. on Networking*, 10:477–486, 2002. (pg 108)

[Gut84] A. Guttman. R-trees: A dynamic index structure for spatial searching. In *Proc. ACM SIGMOD*, 1984. (pp 18, 30, 39, 40, 53, 55, 57, 66, & 145)

[GV03] M. Grossglauser and M. Vetterli. Locating nodes with ease: Last encounter routing in ad hoc networks through mobility diffusion. In *Proc. IEEE Workshop on FTDCS*, 2003. (pg 113)

[GWJ+03] L. Galanis, Y. Wang, S. Jeffery, et al. Locating data sources in large distributed systems. In *Proc. VLDB*, pages 874–885, 2003. (pg 78)

[GYGM04] P. Ganesan, B. Yang, and H. Garcia-Molina. One torus to rule them all: Multi-dimensional queries in P2P systems. Technical Report 24, Stanford University, 2004. (pg 78)

[H+99] E. N. Hanson et al. Scalable trigger processing. In *Proc. ICDE*, pages 266–275, 1999. (pg 52)

[Haa97] Z. Haas. A new routing protocol for reconfigurable wireless networks. In *Proc. ICUPC*, pages 562–565, 1997. (pg 113)

[Haa98] Z. J. Haas. Panel report on ad hoc networks. *Mobile Computing and Communications Review*, 2(1), 1998. (pg 41)

[Hay96] R. Hayton. *OASIS: An Open architecture for Secure Inter-working Services*. PhD thesis, University of Cambridge, 1996. (pp 24, 31, 44, 155, & 167)

[HB02] A. Hinze and S. Bittner. Efficient distribution-based event filtering. In *Proc. Workshop on DEBS*, 2002. (pp 52, 155, 158, & 167)

[HCKW90] N. Hanson, M. Chaabouni, C. H. Kim, and Y. W. Wang. A predicate matching algorithm for database rule systems. In *Proc. ACM SIGMOD*, pages 271–280, 1990. (pg 52)

[HGM01] Y. Huang and H. Garcia-Molina. Publish/subscribe in a mobile environments. In *Proc. Workshop on MobiDE*, 2001. (pp 106, 109, 111, & 112)

[HHH+02] M. Harren, J. Hellerstein, Ryan Huebschothers, et al. Complex queries in DHT-based peer-to-peer networks. In *Proc. Workshop on P2P Systems*, pages 242–250, 2002. (pp 30, 78)

[HHL02] Z. J. Haas, J. Halpern, and L. Li. Gossip-based ad-hoc routing. In *Proc. IEEE INFOCOM*, 2002. (pg 109)

[HHS+99] A. Harter, A. Hopper, P. Stegglesand, et al. The anatomy of a context-aware application. In *Proc. MobiCom*, 1999. (pp 182, 183)

[HPS02] Z.J. Haas, M.R. Pearlman, and P. Samar. *The Zone Routing Protocol (ZRP) for Ad Hoc Networks*. Internet Draft, draft-ietf-manet-zone-zrp-04.txt, 2002. (pp 28, 103, & 106)

[HRS03] J. Hellerstein, S. Ratnasamy, and S. Shenker. Range query over DHTs. Technical Report IRB-TR-03-009, Intel Research, 2003. (pg 78)

[HW+94] B. H. Hoffmann-Wellenhof et al. *GPS: Theory and Practice*. Springer, 1994. (pg 46)

[IBM00] IBM. *IBM MQ Series*. http://www.ibm.com/software/ts/mqseries/, 2000. (pp 21, 37)

[IBM03] IBM. *WebSphere MQ: Connecting your applications without complex programming*. WebSphere White Papers, 2003. (pg 29)

[IET04] IETF. *Delay Tolerant Network Research Group (DTNRG)*. http://www.dtnrg.org, 2004. (pg 102)

[IET05] IETF. *Network Mobility (NEMO), RFC3963*. http://www.ietf.org/html.charters/nemo-charter.html, 2005. (pg 113)

[IM98] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proc. ACM Symposium on Theory of Computing*, 1998. (pg 30)

[IMRV97] P. Indyk, R. Motwani, P. Raghavan, and S. Vempala. Locality-preserving hashing in multi-dimensional spaces. In *Proc. ACM Symposium on Theory of Computing*, 1997. (pg 30)

[JBRA+03] A. Jardosh, E. M. Belding-Royer, K. C. Almerothand, et al. Towards realistic mobility models for mobile ad hoc networks. In *Proc. MobiCom*, pages 217–229, 2003. (pg 113)

[JC01] L. Ji and M. S. Corson. Differential destination multicast - a manet multicast routing protocol for small groups. In *Proc. IEEE INFOCOM*, 2001. (pg 118)

[Jen81] K. Jensen. Colored Petri Nets and the invariant-method. *Theoretical Computer Science*, 1981. (pg 156)

[JFP04] S. Jain, K. Fall, and R. Patra. Routing in a delay tolerant network. In *Proc. ACM SIGCOMM*, 2004. (pp 28, 117)

[JH04] D. Jung and A. Hinze. A meta-service for event notification. In *Proc. CoopIS*, 2004. (pg 31)

[JMH02] D. B. Johnson, D. A. Maltz, and Y.-C. Hu. *The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks*. http://www.ietf.org/internet-drafts/draft-ietf-manet-dsr-07.txt, 2002. (pp 103, 113, 115, & 133)

[JMQ99] P. Jacquet, P. Muhlethaler, and A. Qayyum. *Optimized link state routing protocol*. http://www.ietf.org/proceedings/99mar/I-D/draft-ietf-manet-olsr-00.txt, 1999. (pg 103)

[JNS02] JNS. *Java Network Simulator*. http://jns.sourceforge.net, 2002. (pg 131)

[JPA03] D. Johnson, C. Perkins, and J. Arkko. *Mobility Support in IPv6*. Internet-Drafts, draft-ietf-mobileip-ipv6-24.txt, 2003. (pp 112, 113)

[JSS05] B. Jiao, S. H. Son, and J. A. Stankovic. Gem: Generic event service middleware for wireless sensor networks. In *Proc. INSS*, 2005. (pp 158, 159)

[jxt] jxta.org. http://www.jxta.org/. (pg 75)

[KCC05] S. Kurkowski, T. Camp, and M. Colagrosso. MANET simulation studies: The incredibles. *ACM's Mobile Computing and Communications Review*, 9(4), 2005. (pg 132)

[KF94] I. Kamel and C. Faloutsos. Hilbert R-tree: An improved R-tree using fractals. In *Proc. VLDB*, pages 500–510, 1994. (pg 56)

[KK00] B. Karp and H. Kung. GPSR: Greedy perimeter stateless routing for wireless networks. *Computing and Networking*, pages 243–254, 2000. (pg 107)

[KK05] V. Kawadia and P. R. Kumar. A cautionary perspective on cross layer design. *IEEE Wireless Communication Magazine*, 12(1), 2005. (pg 110)

[KS91] C. Kolovson and M. Stonebraker. Segment indexes: Dynamic indexing techniques for multi-dimensional interval data. In *Proc. ACM SIGMOD*, pages 138–147, 1991. (pg 56)

[KSF+03] M. Koubarakis, T. Sellis, A. Frankand, et al. *Spatio-Temporal Databases: The Chorochronos Approach*. LNCS 2520. Springer, 2003. (pg 41)

[KV99] Y. Ko and N.H. Vaidya. Geocasting in mobile ad hoc networks: Location-based multicast algorithms. In *Proc. WMCSA*, pages 101–110, 1999. (pp 29, 104)

[KV02] Y. B. Ko and N. H. Vaidya. Flooding-based geocasting protocols for mobile ad hoc networks. *Networks and Applications*, 7(6), 2002. (pg 118)

[KV03] Y. Ko and N. H. Vaidya. GeoTORA: A protocol for geocasting in mobile ad hoc networks. In *Proc. ICNP*, pages 240–250, 2003. (pg 104)

[L+99] O. Lassila et al. W3C resource description framework model and syntax specification. Technical report, www.w3.org, 1999. (pp 79, 193)

[L+00] S.-J. Lee et al. A performance comparison study of ad hoc wireless multicast protocols. In *Proc. IEEE INFOCOM*, pages 565–574, 2000. (pp 103, 137, & 139)

[Lab99] NOAA Aeronomy Laboratory. *Surface meteorological data*. http://www.al.noaa.gov/tdc/trmm/kwajex/disdrometer.html, 1999. (pg 181)

[LCB99] C. Liebig, M. Cilia, and A. Buchmannand. Event composition in time-dependent distributed systems. In *Proc. IFIP CoopIS*, 1999. (pp 45, 155, 158, 171, & 175)

[LCP+04] E. K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim. A survey and comparison of peer-to-peer overlay network schemes. In *IEEE communications survey and tutorial*, 2004. (pg 73)

[LEH02] J. Luo, P. Eugster, and J. Hubaux. Route driven gossip: Probabilistic reliable multicast in ad hoc networks. Technical report, EPFL, 2002. (pg 115)

[LGC99] S.-J. Lee, M. Gerla, and C.-C. Chiang. On-demand multicast routing protocol. In *Proc. IEEE WCNC*, pages 1298–1304, 1999. (pp 108, 115, & 119)

[LH00] G. Liang and Z. Haas. Virtual backbone generation and maintenance in ad hoc network mobility management. In *Proc. IEEE INFOCOM*, 2000. (pg 112)

[LHL05] Yu-En Lu, S. Hand, and P. Lio. Keyword searching in hypercube manifold. In *Proc. IEEE P2P*, 2005. (pp 78, 80)

[LJ03] H. Leung and H. Jacobsen. Efficient matching for state-persistent publish/subscribe systems. In *Proc. CASCON*, pages 182–196, 2003. (pg 43)

[LJ04] H. Liu and H. Jacobsen. Modeling uncertainties in publish/subscribe system. In *Proc. ICDE*, 2004. (pg 156)

[LJ05] G. Li and H. Jacobsen. Composite subscriptions in content-based publish/subscribe systems. In *Proc. ACM/IFIP/USENIX Middleware*, 2005. (pg 43)

[LJC+00] J. Li, J. Jannotti, D. De Coutoand, et al. A scalable location service for geographic ad hoc routing. In *Proc. Mobicom*, 2000. (pg 112)

[LKGH03] X. Li, Y. J. Kim, R. Govindan, and W. Hong. Multi-dimensional range queries in sensor networks. In *Proc. Sensys*, 2003. (pp 30, 115, & 158)

[LLE97] S.T. Leutenegger, M.A. Lopez, and J.M. Edgington. STR: A simple and efficient algorithm for R-tree packing. In *IEEE ICDE*, 1997. (pg 56)

[LM03] I. Lazaridis and S. Mehrotra. Capturing sensor-generated time series with quality guarantees. In *Proc. ICDE*, 2003. (pg 166)

[LMK98] G. Liu, A. K. Mok, and P. Konana. A unified approach for specifying timing constraints and composite events in active real-time database systems. In *Proc. Real-Time Technology and Applications Symposium*, 1998. (pp 31, 48)

[LP05] Z. Li and M. Parashar. Comet: A scalable coordination space for decentralised distributed environments. In *Proc. Hot P2P*, 2005. (pg 53)

[LS03] A. Lerner and D. Shasha. AQuery: Query language for ordered data, optimization techniques, and experiments. Technical Report 836, New York University, 2003. (pg 42)

[LSG99] S.-J. Lee, W. Su, and M. Gerla. Ad hoc wireless multicast with mobility prediction. In *Proc. IEEE ICCCN*, pages 4–9, 1999. (pg 115)

[LTLS00] W.-H. Liao, Y.-C. Tseng, K.-L. Lo, and J.-P. Sheu. GeoGRID: A geocasting protocol for mobile ad hoc networks based on grid. *Journal of Internet Technology*, 1(2), 2000. (pg 104)

[LW02] W. Lou and J. Wu. On reducing broadcast redundancy in ad hoc wireless networks. *IEEE Trans. on Mobile Computing*, 1(2):111–122, 2002. (pg 109)

[M+00] J. Moy et al. Multicast routing extensions for OSPF. *CACM*, 37(6), 2000. (pp 28, 29)

[MA02] J. Mellin and S. F. Adler. A formalized schema for event composition. In *Proc. RTCSA*, 2002. (pg 31)

[Mac67] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proc. Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1967. (pg 146)

[MB98] C. Ma and J. Bacon. COBEA: A corba-based event architecture. In *Proc. COOTS*, 1998. (pg 157)

[MCE02] C. Mascolo, L. Capra, and W. Emmerich. Mobile computing middleware. *IFIP-TC6 Networking Conference, Networking Tutorial*, 2002. (pg 26)

[McK03] L. W. McKnight. Towards a sharing protocol for wireless grids. In *Proc. Computer Communication and Control Technologies*, 2003. (pg 28)

[MCP05] L. Mottola, G. Cugola, and G. P. Picco. Tree overlays for publish-subscribe in mobile ad hoc networks. Technical report, Politecnico di Milano, 2005. (pg 110)

[Mei02] R. Meier. STEAM: Event-based middleware for wireless ad hoc networks. In *Proc. DEBS*, 2002. (pg 29)

[MF02] S. Madden and M. J. Franklin. Fjording the stream: An architecture for queries over streaming sensor data. In *Proc. ICDE*, pages 555–566, 2002. (pg 40)

[MFB02] G. Mühl, L. Fiege, and A. Buchmann. Filter similarities in content-based publish/subscribe systems. In *Proc. ARCS*, 2002. (pp 23, 77, & 158)

[MFH+02] S. Madden, M. J. Franklin, J. Hellerstein, et al. TAG: a tiny aggregation tree for ad-hoc sensor networks. In *USENIX Symposium on Operating Systems Design and Implementation*, 2002. (pp 34, 154, & 193)

[MFH+03] S. Madden, J. Franklin, J. M. Hellerstein, et al. The design of an acquisitional query processor for sensor networks. In *Proc. ACM SIGMOD*, pages 491–502, 2003. (pp 30, 115)

[MFWL03] M. Mauve, H. Foler, J. Widmer, and T. Lang. Position-based multicast routing for mobile ad-hoc networks. *SIGMOBILE Mobile Computer Communication Review*, 7(3), 2003. (pg 112)

[MGA03] M. F. Mokbel, T. M. Ghanem, and W. G. Aref. Spatio-temporal access methods. *IEEE Data Engineering Bulletin*, 26(2), 2003. (pg 41)

[Müh01] G. Mühl. Generic constraints for content-based publish/subscribe. In *Proc. CoopIS*, pages 211–225, 2001. (pg 51)

[MHM05] M. Musolesi, S. Hailes, and C. Mascolo. Adaptive routing for intermittently connected mobile ad hoc networks. In *Proc. IEEE WOWMOM*, 2005. (pg 110)

[Mic00] Sun Microsystems. *Project JXTA*. http://www.jxta.org/, 2000. (pg 21)

[Mic01] Sun Microsystems. *Java Message Service (JMS) Specification*. http://java.sun.com/products/jms/, 2001. (pg 21)

[MNPT05] Y. Manolopoulos, A. Nanopoulos, A. N. Papadopoulos, and Y. Theodoridis. *Rtrees: Theory and Applications*. Springer, 2005. (pg 56)

[MSMA91] P. Melliar-Smith, L. Moser, and V. Agrawala. Membership algorithms for asynchronous distributed systems. In *Proc. ICDCS*, 1991. (pg 114)

[MSS97] M. Mansouri-Samani and M. Sloman. Gem: A generalized event monitoring language for distributed systems. *IEE/IOP/BCS Distributed systems Engineering Journal*, 4(2), 1997. (pp 31, 158)

[MWA+03] R. Motwani, J. Widom, A. Arasu, et al. Query processing, approximation, and resource management in a data stream management system. In *Proc. Innovative Data Systems Research*, pages 245–256, 2003. (pp 42, 43, & 155)

[MZ97a] I. Motakis and C. Zaniolo. Formal semantics for composite temporal events in active database rules. *Journal of Systems Integration*, 7(3-4), 1997. (pg 43)

[MZ97b] I. Motakis and C. Zaniolo. Temporal aggregation in active database rules. In *Proc. ACM SIGMOD*, pages 440–451, 1997. (pg 43)

[Nag02] R. Nagpal. Programmable self-assembly using biologically inspired multiagent control. In *Proc. AAMAS*, 2002. (pg 193)

[Nar02] Narada. *The Narada Event Brokering System*. http://grids.ucs.indiana.edu/ptliupages/projects/narada/, 2002. (pp 21, 25)

[NI97] J.C. Navas and T. Imielinski. Geocast - geographic addressing and routing. *Mobile Computing and Networking*, 1997. (pg 118)

[NS03] J. Newsome and D. Song. Gem: Graph embedding for routing and data-centric storage in sensor networks without geographic information. In *Proc. ACM SenSys*, 2003. (pg 107)

[NT04] N. Ntarmos and P. Triantafillou. AESOP: Altruism-endowed self organizing peers. In *Proc. Workshop on Databases, Information Systems and Peer-to-Peer Computing*, 2004. (pg 131)

[OAA+00] L. Opyrchal, M. Astley, J. Auerbach, et al. Exploiting IP multicast in content-based publish-subscribe systems. In *Proc. ACM/IFIP/USENIX Middleware*, pages 185–207, 2000. (pg 143)

[OCD00] M. Oliveira, J. Crowcroft, and C. Diot. Router level filtering on receiver interest delivery. In *Proc. NGC*, 2000. (pg 79)

[OM84] J. Orenstein and T. Merrett. A class of data structures for associative searching. In *Proc. Principles of Database Systems*, 1984. (pg 75)

[OPSS93] B. Oki, M. Pfluegel, A. Siegel, and D. Skeen. The information bus - an architecture for extensive distributed systems. In *Proc. ACM SOSP*, 1993. (pg 72)

[OSG] OSGi. http://www.osgi.org. (pg 35)

[P+02] C. E. Perkins et al. *Ad Hoc On Demand Distance Vector (AODV) Routing*. http://www.ietf.org/internet-drafts/draft-ietf-manet-aodv-12.txt, 2002. (pp 28, 103, & 113)

[PB02] P. Pietzuch and J. Bacon. Hermes: A distributed event-based middleware architecture. In *Proc. Workshop on DEBS*, 2002. (pp 18, 21, 23, 25, 75, & 81)

[PB03] P. Pietzuch and J. Bacon. Peer-to-peer overlay broker networks in an event-based middleware. In *Proc. Workshop on DEBS*, 2003. (pp 25, 72)

[PB04] R. Prakash and R. Baldoni. Causality and the spatial-temporal ordering in mobile systems. *Mobile Networks and Applications*, 9(5):507–516, 2004. (pg 172)

[PC05] O. Papaemmanouil and U. Cetintemel. Semcast: Semantic multicast for content-based data dissemination. In *Proc. ICDE*, 2005. (pg 79)

[PCB00] N. Priyantha, A. Chakraborty, and H. Balakrishnan. The cricket location-support system. In *Proc. MobiCom*, 2000. (pg 194)

[PFLS01] J. Pereira, F. Fabret, F. Llirbat, and D. Shasha. Efficient matching for web-based publish/-subscribe systems. In *Proc. CoopIS*, 2001. (pg 49)

[PGC00] G. Pei, M. Gerla, and T.W. Chen. Fisheye state routing in mobile ad hoc networks. In *Proc. Workshop on Wireless Networks and Mobile Computing*, 2000. (pg 113)

[PGH00] G. Pei, M. Gerla, and X. Hong. Lanmar: Landmark routing for large scale wireless ad hoc networks with group mobility. In *Proc. ACM MobiHoc*, 2000. (pp 107, 113)

[PMR01] G. P. Picco, A. Murphy, and G. Roman. LIME: A middleware for physical and logical mobility. In *Proc. ICDCS*, 2001. (pg 112)

[PR97] E. Pagani and G. Rossi. Reliable broadcast in mobile multi-hop packet networks. In *Proc. Mobicom*, 1997. (pg 115)

[PR99] C. E. Perkins and E.M. Royer. Ad-hoc on-demand distance vector routing. In *Proc. WMCSA*, 1999. (pg 103)

[Pro05] Cayuga Project. Cayuga technical report. Technical Report http://www.cs.cornell.edu/mshong/cayuga-techreport.pdf, Cornell University, 2005. (pp 43, 155, & 167)

[PSB04] P. Pietzuch, B. Shand, and J. Bacon. Composite event detection as a generic middleware extension. *IEEE Network*, 18(1), 2004. (pp 31, 44, 157, 167, & 174)

[PWR04] G. Perng, C. Wang, and M. K. Reiter. Providing content-based services in a peer-to-peer environment. In *Proc. Workshop on DEBS*, 2004. (pg 26)

[PXK+02] S. Prabhakar, Y. Xia, D. V. Kalashnikov, W. G. Aref, and S. E. Hambrusch. Query indexing and velocity constrained indexing: Scalable techniques for continuous queries on moving objects. *IEEE Trans. on Computers*, 51(10), 2002. (pg 41)

[R+99] E. Royer et al. Multicast operation of the ad-hoc on-demand distance vector routing protocol. In *Proc. MobiCom*, 1999. (pp 25, 29)

[RD01] A. Rowstron and P. Druschel. Pastry: scalable, decentraized object location and routing for large-scale peer-to-peer systems. In *Proc. ACM.IFIP/USENIX Middleware*, pages 329–350, 2001. (pp 21, 25, 72, 73, 74, & 81)

[RDJ02] W. Rjaibi, K. R. Dittrich, and D. Jaepel. Event matching in symmetric subscription systems. In *Proc. CASCON*, 2002. (pg 48)

[Res01] IBM Research. *Gryphon: Publish/Subscribe over public networks*. http://researchweb.watson.ibm.com/grypohn/Gryphon/gryphon.html, 2001. (pp 21, 23, & 51)

[RFH+01] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content addressable network. In *Proc. SIGCOMM*, 2001. (pp 25, 26, 73, & 76)

[RHK+01] S. Ratnasamy, M. Handley, R. Karpand, et al. Application-level multicast using content-addressable networks. In *Proc. of NGC*, 2001. (pp 25, 72, 75, & 76)

[RKS+03] S. Ratnasamy, B. Karp, S. Shenker, et al. Data centric storage in sensornets with GHT, a geographic hash table. *Mobile Networks and Applications*, 8, 2003. (pg 30)

[RKY+02] S. Ratnasamy, B. Karp, L. Yin, et al. GHT: A geographic hash table for data-centric storage. In *Proc. Workshop on WSNA*, 2002. (pp 30, 107)

[RLW+02] A. Riabov, Z. Liu, J. Wolf, P. Yu, and L. Zhang. Clustering algorithms for content-based publication-subscription systems. In *Proc. ICDCS*, 2002. (pp 79, 144)

[RLW+03] A. Riabov, Z. Liu, J. Wolf, P. Yu, and L. Zhang. New algorithms for content- based publication-subscription systems. In *Proc. ICDCS*, 2003. (pg 26)

[Röm01] K. Römer. Time synchronization in ad hoc networks. In *Proc. MobiHoc01*, 2001. (pp 172, 175)

[RM04] K. Römer and F. Mattern. Event-based systems for detecting real-world states with sensor networks: A critical analysis. In *Proc. EWSN*, pages 389–395, 2004. (pg 44)

[RMF+00] F. Ramsak, V. Markl, R. Fenk, M. Zirkel, K. Elhard, and R. Bayer. Integrating the UB-tree into a database system kernel. In *Proc. VLDB*, 2000. (pg 53)

[RMSM01] E. M. Royer, P. M. Melliar-Smith, and L. E. Moser. An analysis of the optimum node density for ad hoc mobile networks. In *Proc. ICC*, pages 857–861, 2001. (pg 113)

[RRHS04] S. Ramabadran, S. Ratnasamy, J. M. Hellerstein, and S. Shenker. Brief announcement: Prefix hash tree. In *Proc. ACM PODC*, 2004. (pg 80)

[RT99] E. Royer and C. Toh. A review of current routing protocols for ad-hoc mobile wireless networks. *IEEE Personal Communications Magazine*, 6, 1999. (pp 28, 115)

[SA98] B. Segall and D. Arnold. Elvin has left the building: A publish/subscribe notification service with quenching. In *Proc. AUUG*, 1998. (pg 23)

[Sam95] H. Samet. *Spatial Data Structures*. Addison Wesley/ACM, 1995. (pg 30)

[SAS01] P. Sutton, R. Arkins, and B. Segall. Supporting disconnectedness - transparent information delivery for mobile and invisible computing. In *Proc. CCGrid*, 2001. (pp 29, 112)

[SAW94] B. Schilit, N. Adams, and R. Wantothers. Context-aware computing applications. In *Proc. Workshop on Mobile Computing*, pages 85–90, 1994. (pg 119)

[SCG01] A. Snoeren, K. Conley, and D. Gifford. Mesh based content routing using XML. In *Proc. SOSP*, 2001. (pg 23)

[Sch96] S. Schwiderski. *Monitoring the Behavior of Distributed Systems*. PhD thesis, University of Cambridge, 1996. (pp 158, 171)

[SCS03] Y. Sasson, D. Cavin, and A. Schiper. Probabilistic broadcast for flooding in wireless mobile ad hoc networks. In *Proc. IEEE WCNC*, 2003. (pg 118)

[SGAA04] O. D. Sahin, A. Gupta, D. Agrawal, and A. El Abbadi. A peer-to-peer framework for caching range queries. In *Proc. ICDE*, pages 165–176, 2004. (pp 25, 78)

[SH98] M. Sullivan and A. Heybey. Tribeca: A system for managing large databases of network traffic. In *Proc. USENIX Annual Technical Conf.*, 1998. (pp 40, 42)

[SIE] SIENA. http://www.cs.colorado.edu/users/carzanig/siena/. (pp 72, 75)

[SMLN+04] I. Stoica, R. Morris, D. Liben-Nowell, et al. Chord: A peer to peer lookup protocol for internet applications. *IEEE/ACM Trans. on Networking*, 11, 2004. (pp 25, 72, & 73)

[Sof98] Softwired. *iBus Messaging*. http://www.softwired-inc.com/, 1998. (pp 21, 23)

[SP03] C. Schmidt and M. Parashar. Flexible information discovery in decentralized distributed systems. In *Proc. High-Performance Distributed Computing*, 2003. (pg 78)

[SRD02] R. J. Shah, Z. Ramzan, and R. Dendukuri. Efficient dissemination of personalized information using content-based multicast. In *Proc. IEEE INFORCOM*, 2002. (pg 79)

[SRJB03] R. Shah, S. Roy, S. Jain, and W. Brunette. Data MULEs: Modeling a three-tier architecture for sparse sensor networks. In *Proc. IEEE Workshop on SNPA*, 2003. (pg 115)

[TAJ03] D. Tam, R. Azimi, and H-A. Jacobsen. Building content- based publish/subscribe systems with distributed hash tables. In *Proc. DBISP2P*, 2003. (pp 24, 26, 76, 77, 78, & 87)

[TBao03] W. W. Terpstra, S. Behnel, and L. Fiege ad others. A peer-to-peer approach to content-based publish/subscribe. In *Proc. Workshop on DEBS*, 2003. (pp 72, 76)

[TC95] M. R. Tremlay and M. R. Cutkosky. Using sensor fusion and contextual information to perform event detection during a phase-based manipulation task. In *Proc. IEEE/RSJ Intelligent Robots and Systems*, 1995. (pg 156)

[TE04] P. Triantafillou and A. Economides. Subscription summarization: A new paradigm for efficient publish/subscribe systems. In *Proc. ICDCS*, 2004. (pp 76, 78, & 124)

[TFW04] M. Transier, H. FüSSler, and J. Widmer. Scalable position-based multicast for mobile ad-hoc networks. In *Proc. Workshop on BroadWim*, 2004. (pg 112)

[TGB00] C. Toh, G. Guichal, and S. Bunchua. Abam: On-demand associativity-based multicast routing for ad hoc mobile networks. In *Proc. IEEE VTC*, 2000. (pg 29)

[THB+02] J. Tian, J. Haehner, C. Becker, et al. Graph-based mobility model for mobile ad hoc network simulation. In *Proc. Annual Simulation Symposium*, pages 337–344, 2002. (pg 113)

[TIB98] TIBCO. *TIB/Rendezvous White Paper*. http://www.rv.tibco.com, 1998. (pp 21, 23)

[TS97] G. Taskale and P. Stirpe. Reliable multicast transport protocol (RMTP). *IEEE Journal on Selected Areas in communications*, 15(3), 1997. (pg 114)

[TV04] J. Tchakarov and N. Vaidya. Efficient content location in wireless ad hoc networks. In *Proc. MDM*, 2004. (pg 119)

[V+06] K. Viswanath et al. Exploring mesh and tree-based multicast routing protocols for MANETs. *IEEE trans. on Mobile Computing*, 5(1), 2006. (pg 139)

[vRBV03] R. van Renesse, K. Birman, and W. Vogels. Astrolabe: A robust and scalable technology for distributed systems monitoring. *ACM Trans. on Computing Systems*, 21, 2003. (pg 72)

[W+04] B. Wang et al. UB-Tree based efficient predicate index with dimension transform for pub/sub system. In *Proc. DASFAA*, pages 63–74, 2004. (pp 52, 53)

[W3C03] W3C. *OWL Web Ontology Language, A set of W3C Candidate Recommendations*. W3, 2003. (pg 34)

[WC96] J. Widom and S. Ceri. *Active Database Systems: Triggers and Rules For Advanced Database Processing*. Morgan Kaufmann Publishers, 1996. (pg 155)

[WC02] B. Williams and T. Camp. Comparison of broadcasting techniques for mobile ad hoc networks. In *Proc. MOBIHOC*, pages 194–205, 2002. (pp 103, 108, 137, & 139)

[Web] WebLogic. http://www.bea.com/products/index.shtml. (pg 21)

[WKM00] T. Wong, R. H. Katz, and S. McCanne. An evaluation on using preference clustering in large-scale multicast applications. In *Proc. IEEE INFOCOM*, pages 451–460, 2000. (pg 143)

[WQA+02] Y. Wang, L Qiu, D Achlioptas, et al. Subscription partitioning and routing in content-based publish/subscribe networks. In *Proc. ICDCS*, 2002. (pp 79, 143)

[WQV+04] Y. Wang, L. Qiu, C. Verbowski, et al. Summary-based routing for content-based event distribution networks. *ACM Computer Communication Review*, 2004. (pg 80)

[YB04a] E. Yoneki and J. Bacon. An adaptive approach to content-based subscription in mobile ad hoc networks. In *IEEE Pervasive Computing and Communications (PerCom) - Workshop on Mobile Peer-to-Peer Computing (MP2P)*, pages 92–97, 2004. (pp 110, 131)

[YB04b] E. Yoneki and J. Bacon. Content-based routing with on-demand multicast. In *IEEE International Conference on Distributed Computing Systems (ICDCS) - Workshop on Wireless Ad Hoc Networking (IWWAN)*, pages 788–793, 2004. (pp 110, 131)

[YB04c] E. Yoneki and J. Bacon. Towards a peer-to-peer event broker grid in a hybrid network environment. In *Proc. IFIP Workshop on GADA*, volume LNCS 3292, pages 198–210, 2004. (pp 79, 193)

[YB05a] E. Yoneki and J. Bacon. Object tracking using durative events. In *Proc. IFIP NCUS*, volume LNCS 3823, pages 652–662, 2005. (pg 33)

[YB05b] E. Yoneki and J. Bacon. A survey of wireless sensor network technologies: Research trends and middleware's role (under revision). Technical Report UCAM-CL-TR646, University of Cambridge, 2005. (pp 16, 33, 37, & 156)

[YB05c] E. Yoneki and J. Bacon. Unified semantics for event correlation over time and space in hybrid network environments. In *Proc. IFIP CoopIS*, volume LNCS 3760, pages 366–384, 2005. (pg 160)

[YB06] E. Yoneki and J. Bacon. *Openness and Interoperability in Mobile Middleware*. ISBN 0-849-33833-6. CRC Press, 2006. (pg 33)

[YC99] S. Yang and S. Chakravarthy. Formal semantics for composite events in distributed systems. In *Proc. ICDE*, 1999. (pg 155)

[YEG99] H. Yu, D. Estrin, and R. Govindan. A hierarchical proxy architecture for internet-scale event services. In *Proc. WETICE*, 1999. (pg 24)

[YGM99] T. W. Yan and H. Garcia-Molina. The sift information dissemination system. *ACM Trans. Database Systems*, 24(4), 1999. (pg 52)

[YLSG02] Y. Yi, S.-J. Lee, W. Su, and M. Gerla. *On-Demand Multicast Routing Protocol (ODMRP) for Ad-Hoc Networks*. http://www.ietf.org/internet-drafts/draft-ietf-manet-odmrp-04.txt, 2002. (pg 103)

[Yon03] E. Yoneki. Many aspects of reliability in a distributed mobile messaging middleware over jms. In *Proc. IFIP Workshop on Reliable and Secure Middleware*, volume LNCS 2889, pages 934–949, 2003. (pg 114)

[Yon05] E. Yoneki. Event broker grids with filtering, aggregation, and correlation for wireless sensor data. In *Proc. IFIP Workshop on GADA*, volume LNCS 3762, pages 304–313, 2005. (pg 33)

[YSG03] A. Yalamanchi, J. Srinivasan, and D. Gawlick. Managing expressions as data in relational database systems. In *Proc. CIDR*, 2003. (pg 43)

[Z+01] S. Q. Zhuang et al. Bayeux: An architecture for scalable and fault-tolerant wide-area data dissemination. In *Proc. ACM NOSSDAV*, pages 11–20, 2001. (pp 25, 72, 74, & 76)

[Z+03] J. Zygmunt et al. Gossip-base ad hoc routing: Probabilistic guarantees and algorithms for ad hoc networks. Technical Report Computer and Communication Sciences, EPFL, 2003. (pg 116)

[ZA03] W. Zhao and M. Ammar. Proactive routing in highly-partitioned wireless ad hoc networks. In *Proc. IEEE Workshop on Future Trends of Distributed Computing Systems*, 2003. (pg 111)

[ZAZ04] W. Zhao, M. Ammar, and E. Zegura. A message ferrying approach for data delivery in sparse mobile ad hoc networks. In *Proc. of ACM MMOBIHOC*, 2004. (pg 112)

[ZAZ05] W. Zhao, M. Ammar, and E. Zegura. Multicasting in delay tolerant networks: Semantic models and routing algorithms. In *Proc. SIGCOMM Workshop*, 2005. (pg 105)

[ZF03] A. Zeidler and L. Fiege. Mobility support with REBECA. In *Proc. Workshop on Mobile Computing Middleware*, 2003. (pg 113)

[Zha04] W. Zhang. *Performance analysis of UB-tree indexed publish/subscribe system.* PhD thesis, University of Tokyo, 2004. (pg 52)

[ZHS+04] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. D. Kubiatowicz. Tapestry: A resilient global-scale overlay for service deployment. *IEEE Journal on Selected Areas in Communications*, 22, 2004. (pp 72, 74, & 76)

[ZS00] H. Zhou and S. Singh. Content based multicast (CBM) in ad hoc networks. In *Proc. MOBIHOC*, 2000. (pg 109)

[ZU99] D. Zimmer and R. Unland. On the semantics of complex events in active database management systems. In *Proc. ICDE*, pages 392–399, 1999. (pg 155)