

DS 2010 event-driven systems

David Evans

de239@cl.cam.ac.uk

Event-driven communication paradigm

- asynchronous message-passing rather than request-reply
- **advertise**, **subscribe**, **publish/notify** for scalability
 - e.g. subscribe to and be notified of
bus-seen-event (busID=uni4., location=*)*
- event-driven paradigm for ubiquitous computing: sensors generate data, notified as events
- compose/correlate events for higher level semantics
 - e.g. **traffic congestion**, **pollution**
- database integration – how best to achieve it?

Event-Driven Systems CEA

Cambridge Event Architecture (CEA), 1992 -

- extension of O-O **middleware**, typed events
 - “**advertise**, **subscribe**, **publish/notify**”, direct or mediated
 - publishers (or mediators if > 1 publisher for a type) process subscription filters and multicast to relevant subscribers
- federated event systems
 - gateways/contracts/XML
- applications
 - multimedia presentation control
 - pervasive environments (active house, active city, active office, active bus)
 - tracking mobile entities (active badge)
 - telecommunications monitoring and control

Event-Driven Systems - Hermes

- **Hermes** large-scale event service, 2001-4
 - PhD work of Peter Pietzuch at CL
- loosely-coupled
- publish/subscribe
- widely distributed event-broker network
- via a P2P overlay network (DHT e.g. Pastry)
- distributed filtering (optimise use of comms.)
- rendezvous nodes for advertisers/subscribers

Use of P2P/DHT substrate

- Broker IP addresses hashed into 128-bit space
- Event topics hashed into 128-bit space. Topic is managed by broker with nearest value $>$ topic hash
- Brokers keep tables of nearest neighbours (for different common prefixes) in 128-bit space – see next slide
- Event messages from pubs and subs routed to broker nearest to event topic's hash value in $O(\log N)$ hops – called the “**rendezvous node**” for that topic
- Paths to same destination converge quickly
- Paths shared to nearby destinations – late fanout
- Resilient to join/leave/failure of nodes
- Scales to millions of nodes

Pastry node 2030xx...’s routing table starts:

0* Id,a	1* Id,a	2*	3* Id,a
20*	21* Id,a	22* Id,a	23* Id,a
200* Id,a	201* Id,a	202* Id,a	203*
2030* etc.	2031* Id,a	2032* Id,a	2033* Id,a

e.g. route to **1**xxxx...

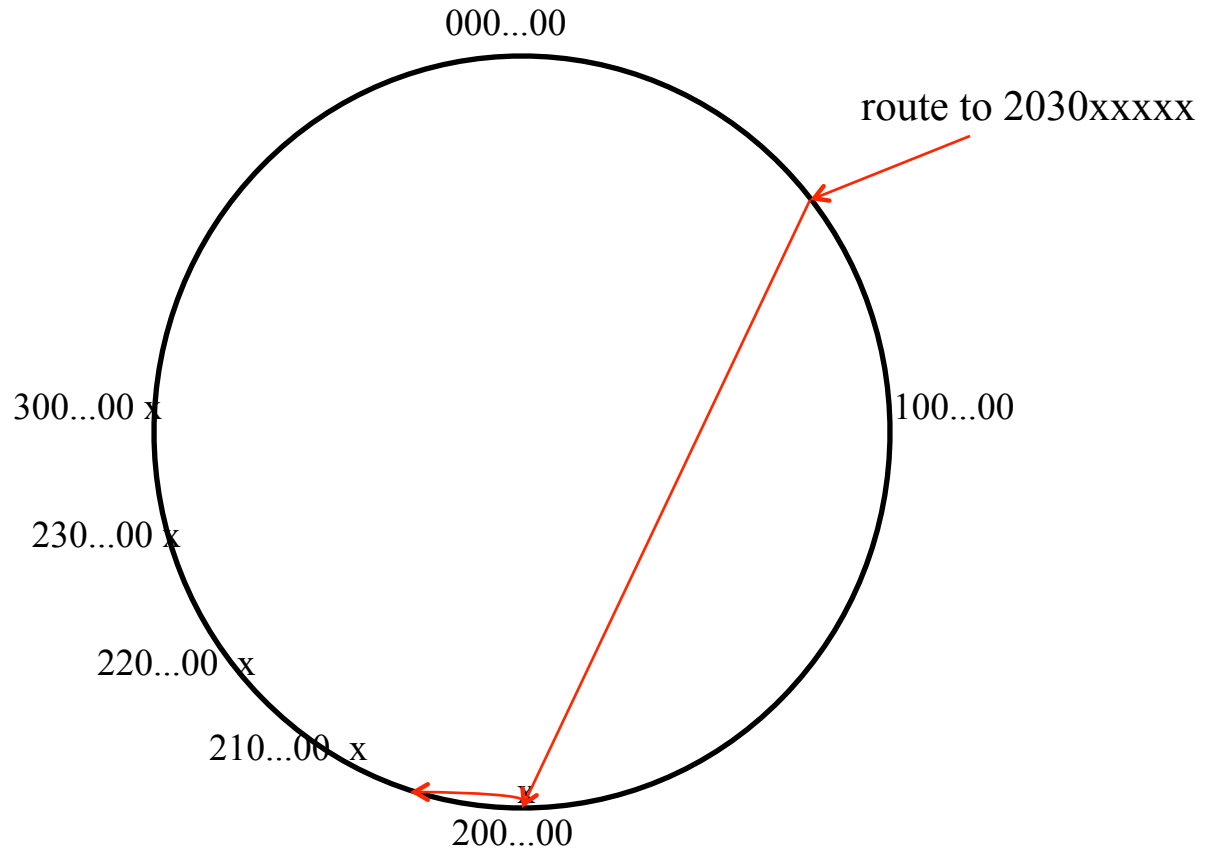
e.g. route to **2**xxx...

e.g. route to **20**xxx...




e.g. route to **203**xx...

- nodeIds and keys are in some base 2^b (e.g. $b=2$ here)
- each entry, except that for the broker itself, contains the ‘Id’ and IP address ‘a’ of another node

Pastry route convergence

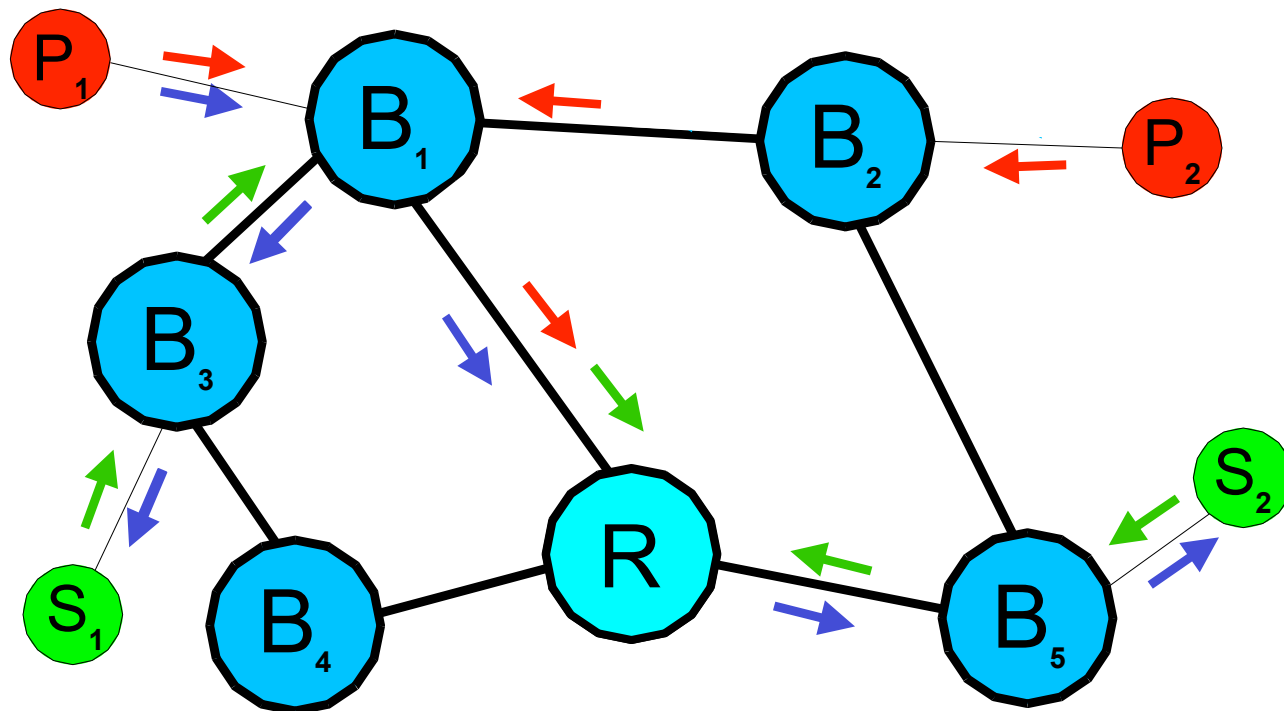


Hermes Pub/Sub Design

- Event Brokers 
 - provide middleware functionality
 - logical overlay P2P network: content-based routing+filtering
 - easily extensible
- Event Clients: Publishers , Subscribers 
 - connect to any Event Broker
 - publishers **advertise**,
 - subscribers **subscribe** (brokers set up routing state),
 - publishers **publish**,
 - brokers route messages and **notify** publications to subscribers
 - lightweight, language-independent

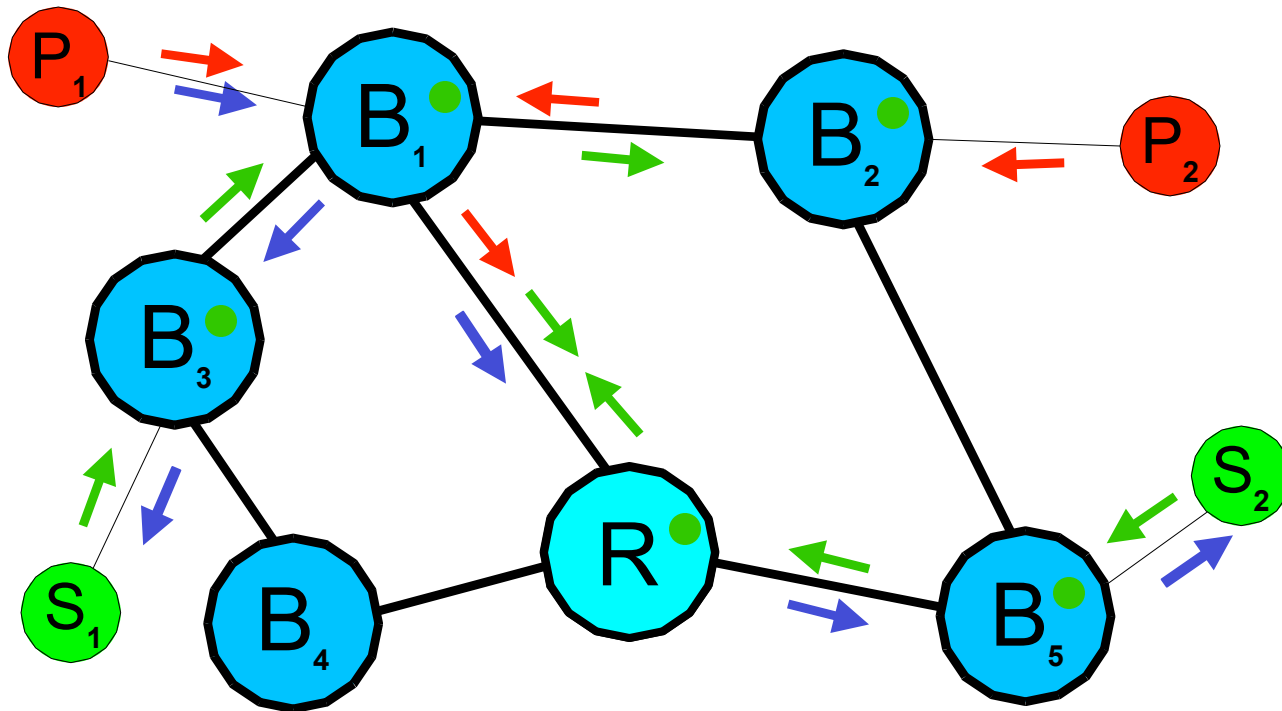
Algorithms I – Topic-Based Pub/Sub

- Type Msg, Advertisements, Subscriptions, Notifications
- Rendezvous Nodes
- Reverse Path Forwarding
 - Notifications follow Ads then the reverse path of Subs



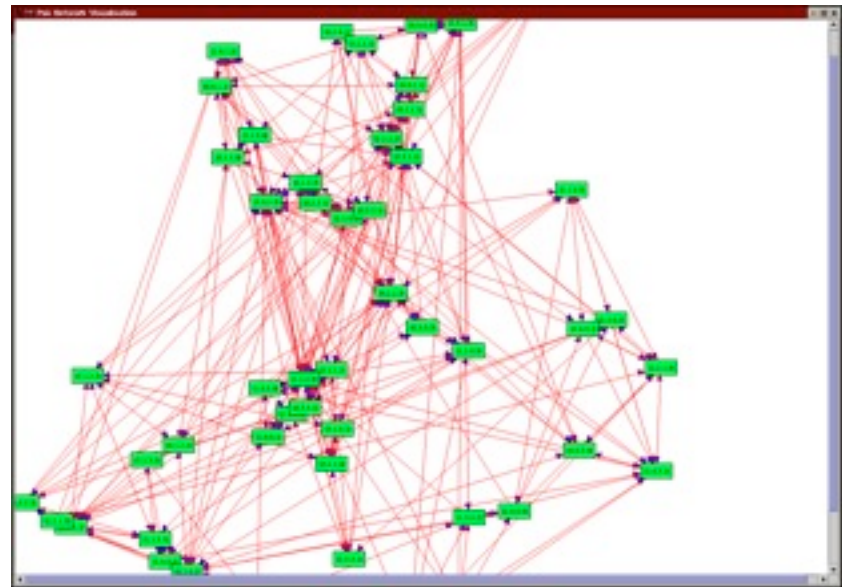
Algorithms II – Content-Based Pub/Sub

- Filtering State ●
- Notifications follow reverse paths of subscriptions
- Subscriptions consolidated by covering and merging for path sharing before late fanout



Hermes Implementation

- Actual implementation
 - Java implementation of event broker and event clients
 - Event types defined in XML Schema
 - Java language binding for events using reflection
- Implementation within a simulator
 - Large-scale, Internet-like topologies
 - over 100 nodes (gosh!)
 - used in later projects



Pub/sub not sufficient for general applications

- decouples publishers and subscribers
 - pubs and/or subs need not be running at the same time
- publishers are anonymous to subscribers
 - subs need to know topic (attributes), not pubs' names and locations
 - but receivers may **need** to know the sender or sender's role
- only multicast, one-to-many communication
 - may also need one-to-one and request-reply
- can't reply
 - either anonymously, e.g. to vote, or identifiably
- efficient notification for large-scale systems using content-based routing
 - but content-based routing may violate privacy of information
 - subscriptions may also be confidential

Event-Driven systems – Composite Events

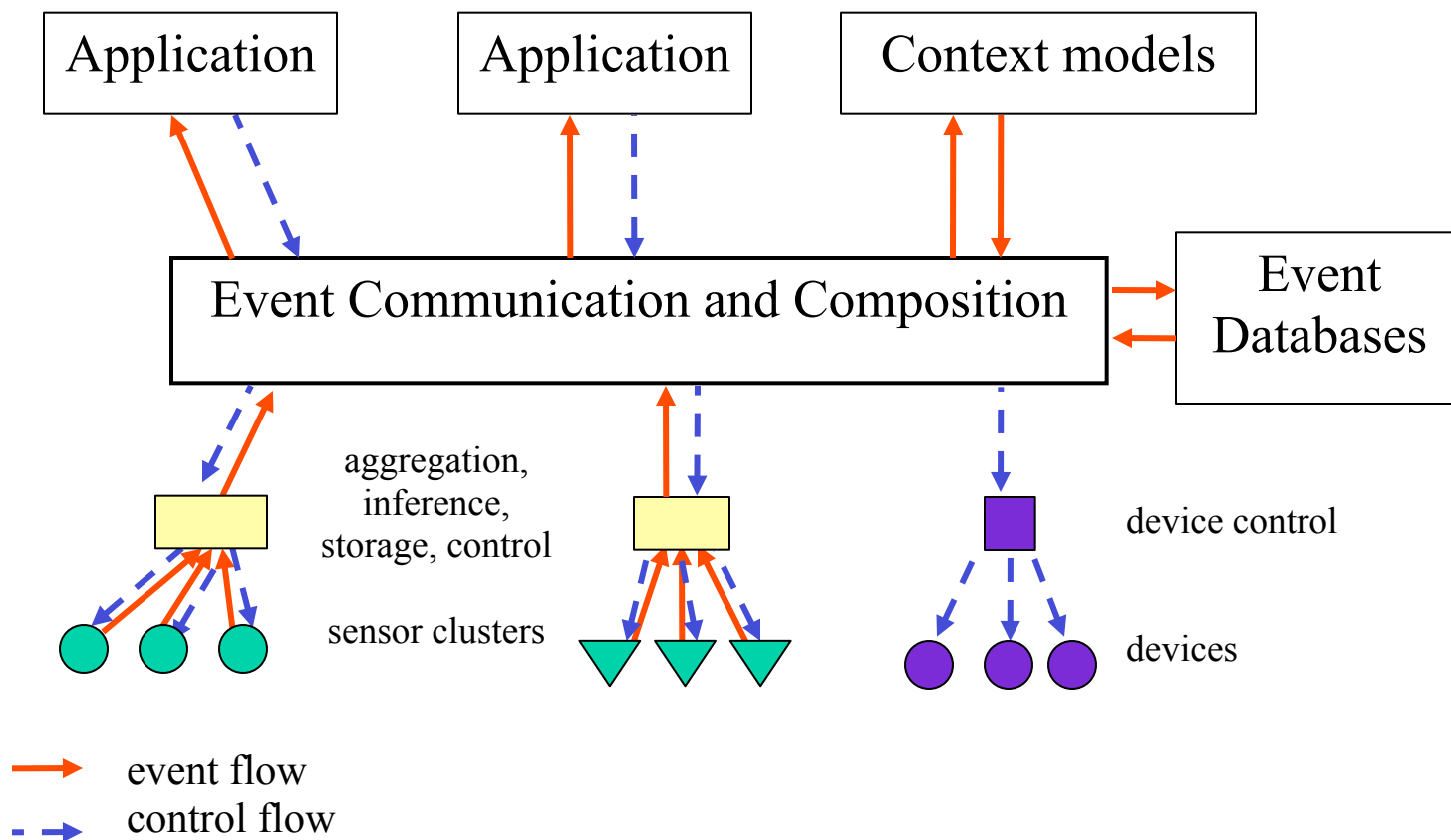
- Event composition (correlation)
 - Pietzuch, Shand, Bacon, Middleware 2003,
and IEEE Network, Jan/Feb 2004
- composite event service above event brokers
- service instances placed to optimise communication
- FSM recognisers – parallel evaluation
- events have source-specific interval timestamps
- simulations of large-scale systems...

Bottom-up and/or Top-Down?

- Can we express all we require by bottom-up composition of primitive events?
- Can we take advantage of **high-level models of context**?
 - e.g. maps, plans, mathematical models, GIS
- What can users be expected to express?
- How is the *top-down, bottom-up gap* bridged and high-level requirements converted into event subscriptions?

“nearest empty meeting room?”, “turn off the lights if the room is empty”, “quickest way to get to Stansted airport?”

Integrating sensor networks (1)



Integrating sensor networks (2)

- ***Data:***
 - sensor-ID, data value, timestamp, location
 - value aggregation from densely deployed sensors
 - inaccuracies masked – data cleansing
 - heterogeneous sensor data correlated (fused)
- ***Information/semantics:***
 - **events defined**, to present sensor data to applications including context models
 - events correlated, **higher-level events generated**
 - real-time delivery may be required
 - level of data logging required (keep all sensor data?)

Traffic monitoring applications

- **sensors:**
 - SCOOT loops for counting,
 - video cameras – extract and transmit anonymised data
 - thermal imaging (infra-red detectors),
 - acoustic detectors
 - bus location data
 - car-park occupancy detection
 - ANPR (automatic number-plate recognition)
- I subscribe to
 - *bus-seen-event (busID=uni4.*, location=MadingleyP&R)*and my desktop is pinged when the bus is detected.

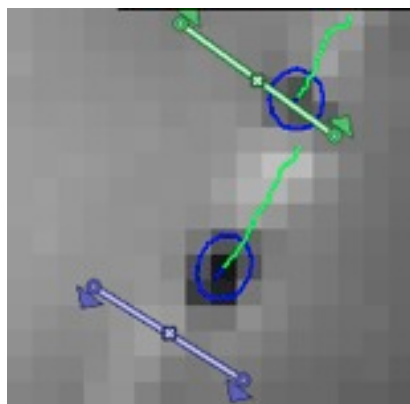
Traffic monitoring applications (cont'd)

- **Route-advice service:** on entering my car I indicate my destination on a map – route is shown, car monitored and route updated dynamically as conditions change.
 - Research on sharing of data within SatNav systems.
- Easy to do with bespoke systems and/or coupling applications with sensors in “vertical silos” e.g. car park data only sent to displays on radial roads into cities
- **Sharing of (public) data:** application developers (and public) can build services by subscribing to advertised events

Work-in-Progress: TIME-EACM project

Irisys infra-red detectors

- combined with video for validation
 - (and banal tasks like positioning the lines)
- privacy-preserving
- Testing carried out: Dept. Eng. roof, Fen Causeway, 2006
~ 90% accuracy cf. video
- wired communication via Engineering Dept.



Irisys – mirroring annual manual count

- carried out on Cambridge radial roads annually
 - we did Huntingdon Rd, 9th Oct 2006, 8am – 7pm
 - using one of DTG’s sentient vans
 - amateur positioning (by us)
 - incoming and outgoing traffic
 - validated against video
 - over 90% accuracy cf. video if cycles excluded



Irisys – ongoing monitoring

- mounted on a lamp post on Madingley Road
- connection to CL via Wifi



Stagecoach/ACIS bus monitoring

- GPS location of buses on most Stagecoach routes
 - radio transfers data back to base
 - some GPRS, some custom
 - bus-stop displays
 - timetables and expected arrival times
 - **Minibus** project – mobile phone selection of bus stops and display of information
 - live and historical data from ACIS since Aug 2007
 - for project use - under a NDA
 - this data allows **journey times** and **congestion** to be analysed and predicted
- ⇒ a very natural fit for events!

Healthcare monitoring application

sensors: **body sensors** for blood-pressure, blood-sugar, etc.

cameras / thermal imaging in smart homes, **tag** objects

- Emergency detection based on sensor values and image analysis – how to decide when to summon help?
- Smart homes: monitoring for falls, visitors, ...
 - (guide-dogs–vs–people?) (visitors–vs–burglars?)
- Tagging objects: “where did I leave ...?” (*pull* model)
or to build a world model for navigation & avoiding obstacles
- Economic model? cost of technology–vs–more people? risks/
costs of false positives and false negatives?
- **Work-in-Progress: CareGrid and PAL projects**

Integrating databases with pub/sub

- **DB world:** continuous queries require recording of individual queries and individual response, one-to-one.
- instead, **Event-Based world:** databases advertise events
 - *event type (<attribute-type>)*
e.g. “cars-for-sale(maker, model, colour, automatic?, ...)”
advertised by many databases e.g. in the Cambridge area
- clients **subscribe** and are **notified** of occurrences
- the pub/sub service does the filtering – not the database
- we have used PostgreSQL – active predicate store

DB Motivating Example – Police IT

Bill Hayden is suspected of masterminding a nationwide terrorist organisation.

- As well as looking up his past database records, the investigators (special terrorism unit) **subscribe**, in all 43 (or whatever) police counties, to **advertised** database update events specifying his name as an attribute.
 - *Note inter-domain naming and access control.*
- Triggers are set in the databases so that any future records that are made, relating to his movements and activities, will be **published** and **notified** automatically and immediately to *those authorised* to investigate him.

Securing pub/sub using RBAC

- At the event client level – use RBAC
 - domain-level authorisation policy indicates, for event types and attributes, the **roles** that can **advertise/publish** and **subscribe**
 - inter-domain subscription is negotiated, as for any other service
 - note that spamming is prevented – only authenticated principals can use the pub/sub service to advertise/publish
- At the event-broker level – use encryption
 - are all the event brokers **trusted**?
 - if not, some may not be allowed to see (decrypt) some (attributes of) some messages.
 - this affects content-based routing

Work-in-Progress – SmartFlow project