Distributed Multicast Grouping for Publish/Subscribe over Mobile Ad Hoc Networks

Eiko Yoneki and Jean Bacon
University of Cambridge Computer Laboratory
J J Thomson Avenue, Cambridge CB3 0FD, UK
Email: {eiko.yoneki, jean.bacon}@cl.cam.ac.uk

***Abstract*— Event broker grids that deploy the publish/subscribe communication paradigm extend the capabilities of seamless messaging in heterogeneous network environments. In order to support such mixed network environments, we integrate publish/subscribe semantics with multicast routing in mobile ad hoc networks. Dynamic construction of an event dissemination structure to route events from event publishers to subscribers is important, especially to support highly dynamic, self-organizing, mobile peer-to-peer systems. The proposed approach aggregates content-based subscriptions in a compact data format (Bloom filters), and ODMRP (On-Demand Multicast Routing Protocol) is extended to use this context from the middleware-tier to construct an optimized dynamic dissemination mesh. Thus, dynamic multicast groups are created by aggregating subscriptions, applying K-means clustering and other methods. Cooperation between the middleware-tier and network components allows fine-grained subscriptions to be represented.**

I. Introduction

Event-based middleware, based on the publish/subscribe communication paradigm, has become popular because asynchronous, many-to-many communication is well suited to constructing reactive distributed systems. Most distributed event-based middleware contains three main elements: a producer who publishes events (messages), a consumer who subscribes his interests to the system, and an event broker to match and deliver the events to the corresponding consumers. Usually event brokers are connected in an arbitrary topology. There is a diversity of large scale network environments, and messages may flow from tiny sensor networks to Internet-scale peer-to-peer (P2P) systems. Mobile ad hoc networks (MANET) lie between these extremes. In such hybrid and mixed network environments, the event-based middleware takes an important role in providing efficient communication mechanisms. MANET is a dynamic collection of nodes with rapidly changing multi-hop topologies that are composed of wireless links. The combination of mobile devices and ad hoc networks is best managed by the creation of highly dynamic, self-organizing, mobile P2P systems. Especially with the recent evolution of distributed event-based middleware over P2P overlay network environments, the construction of event broker grids will extend a seamless messaging capability over heterogeneous network environments.

Most early event-based middleware systems were based on the concepts of group (channel) or subject (topic) communication. These systems categorize events into pre-defined groups. Some early research projects defined content-based notification systems, for example, the Cambridge Event Architecture (CEA) was content-based and used a programming language type-system for events [1]. The content-based model [4] allows subscribers to use flexible subscription languages to declare their interests with respect to event contents. Events are routed based on their content and consumer subscriptions. A key challenge when building large-scale event-based middleware is efficient propagation of subscriptions among brokers. This is particularly difficult in systems that support mobility.

There have been efforts to create efficient multicast communication for MANET (see Section II). When the scale of group communication is small, multicast can be efficient. However, if the group gets larger and a multicast group must define a specific topic for each receiver (e.g., a consumer alert with a fine-grained subscription), it requires a more efficient communication mechanism such as publish/subscribe. Publish/subscribe becomes powerful in MANET environments, when not all the nodes are in an exclusively ad hoc topology. In this case, 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. Broker nodes that offer data filtering services can coordinate data flow efficiently. Thus, to achieve improved asynchronous communication, the semantics of publish/subscribe must be introduced. Maintaining group membership and efficient delivery of events to all members is challenging. Nodes that are mobile have a limited transmission range; a source-to-destination path could pass through several intermediate nodes, leading to a dramatic increase in complexity. Context-awareness is important for improving the accuracy of data dissemination and performance in such ubiquitous environments. Many mobile ad hoc routing protocols take account of contexts such as location, topology, and mobility patterns to construct optimized routing. Integration of contexts from application and middleware to network components is becoming important.

In this paper, we describe an integrated approach to a content-based message dissemination mechanism, that extends ODMRP (On-Demand Multicast Routing Protocol) [5]. ODMRP supports optimized data dissemination mechanisms with context awareness, including location, network topology, network capabilities (e.g., bandwidth and stability), and mobility. To support publish/subscribe systems it is crucial to construct the event dissemination structure dynamically. We optimize the construction by defining an interface to apply the context from a publish/subscribe system to ODMRP. The context consists of subscriptions and message advertisements. We further extend the definition of context to conditions set by the middleware. The interface is generic; to supply data to

be attached to ODMRP packets and indicate how to process them. Our proposal is conceptually similar to Active Networks, which allow users to inject customized programs into the nodes of the network. Content-based subscriptions at a broker node are aggregated and summarized into a compact data format in Bloom filters [3], and the event publisher broker node operates multicast grouping by examining the propagated subscriptions, applying K-means clustering and other methods. This dynamic group construction is one of the difficult challenges. Context-awareness allows both middleware and network layer components to exploit information, providing an efficient and dynamic event routing mechanism for better performance.

This paper proceeds as follows: section II describes multicast protocols, section III describes the publish/subscribe system, section IV reports experimental results, section V discusses related work, and we conclude with section VI.

II. Multicast Protocols

Multicast has traditionally been used as the transport for messaging systems. However today's IP-based multicast schemes are not scalable to support large groups, and a good infrastructure to support multicast is not available in wide area networks. Thus, Application-Level Multicast Routing Protocols (ALMRPs) are replacing group communication over wide area networks. A typical design is as two layered protocols: a protocol that maintains routing, above which a multicast tree is constructed. Most ALMRPs use tree routing for logarithmic scaling with respect to receiver numbers: as the node connectivity changes, the tree structure changes accordingly. Non tree routing examples are Narada [12], Bayeux/Tapestry [21] and CAN [13]. However, the multicast service model is less powerful than that of a content-based network model, and there is currently no optimal way of using or adapting the multicast routing infrastructure to provide a content-based service.

For wireless networks, broadcast is the most natural form of communication. The publish/subscribe messaging model maps well onto a decentralized group structure in MANET. In general, there are three types of routing mechanism in MANET. Proactive (table driven) protocols use a traditional, distributed, shortest-path and maintain routes at all times. They impose a high overhead. Reactive (on-demand) protocols are initiated by the source. They have lower traffic compared with proactive protocols, but impose delay in route determination. Hybrid protocols are an adaptive way of combining proactive and reactive protocols. One important difference from static network environments is that routing optimality is not the highest priority, because of the dynamic environment due to mobile nodes. Several multicast routing protocols for MANET have been developed. DVMRP (Distance Vector Multicasting Routing Protocol) builds a source-based tree. AODV (Ad-hoc On-Demand Distance Vector Routing Protocol) builds a core-based tree. CAMP (Core Assisted Mesh Protocol) builds a mesh interconnection of hosts.

ODMRP applies an on-demand routing technique to avoid channel overhead and improve scalability (see Fig. 1). ODMRP results from incorporating FGMP(Forwarding Group Multicast Protocol) with an on-demand scheme. It attempts to create a group of forwarding nodes between the source and the multicast receivers. 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. The concept of a forwarding group implies that only a subset of nodes forwards multicast packets (scoped flooding).

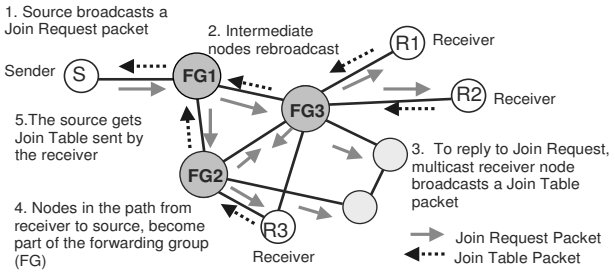

Fig. 1: ODMRP Operation

ODMRP provides a richer connectivity among multicast members using a mesh-based approach. It supplies multiple routes for one particular destination, which helps if the topology changes or nodes fail. ODMRP takes a soft-state approach to maintain multicast group members. Nodes need not send any explicit control message to leave the group. However, there are issues to be considered, for example an increase in senders leads to control overhead. For more detail on ODMRP, see [5].

III. Publish/Subscribe Semantic Multicast

The messaging system in MANET should be self organizing. This is because the topology of a mobile P2P system has to constantly adjust itself, by discovering new communication links, and also needs to be fully decentralized due to the lack of a central access point. In such environments it is best to create a routing table on demand, as in ODMRP. According to simulation studies, [8], ODMRP performs well with regard to throughput and control packet overhead. ODMRP is simple and scalable, through avoiding the drawbacks of multicast trees such as intermittent connectivity and frequent tree reconfiguration. We therefore selected ODMRP for the underlying data dissemination mechanism. It is also an appropriate substrate for various on-demand multicast protocols for MANET. ODMRP improves its performance using mobility and location information. Many contexts are within the network, and are outside the scope of middleware. On the other hand, the semantic contexts from the upper layers should be used to build efficient communication by the network layer component. Here, there is a need to exchange contexts among applications, the middleware tier and the network layer, to build an optimized data dissemination structure.

Subscription Models: One of the key issues in supporting messaging systems is designing subscription models. Topic-based addressing is an abstraction of numeric network addressing schemes. With the content-based subscriptions used in SIENA and Gryphon [6], delivery depends only on message content, extending the capability of event notification with more expressive subscription filters. In Pronto [18], besides

topic-based routing, a filtering function that selects messages based on their content is supported. The content filter language is based on the WHERE syntax of SQL92 over XML messages. Content-based publish/subscribe is essential for better (filtered) data flow in mobile computing. The most advanced and expressive form of subscription language is content-based with pattern matching; such a language is important for event notification. Common topic-based systems arrange topics in hierarchies, but a topic cannot have several super topics. Type-based subscription provides a natural approach to this, if the language offers multiple sub-typing, thus avoiding explicit message classification through topics. This works well with typed languages, but it is complex to deploy this degree of serialization of objects. Moreover, mobile applications may not have the concept of objects or typing. Thus, the combination of hierarchical topics and high speed content filtering could be a more flexible approach for mobile applications. Research is also ongoing to structure complex content-based data models [11] and reflection-based filters. In [17], subscription partitioning is attempted to obtain better system throughput and less network traffic. A similar approach can be used to create multicast groups from aggregated subscriptions.

Our proposed system follows the basic model of an event-based middleware. Publisher, subscriber, and broker are the elements of the system and content-based subscriptions are used when ODMRP builds the routing table to disseminate the events. Content-based routing is computationally more expensive than explicit-address routing or topic-based routing. In static P2P network environments, content-based subscriptions are used to construct the routing itself. However, over dynamic mobile ad hoc network environments, the multicast channel life is ephemeral and subscriptions are expected to be more specific. In such network environments, epidemic-style data dissemination should work better. Channels should be established dynamically, as in our approach, to realize content-based publish/subscribe. Note that it is necessary to develop data structures and algorithms that can introduce efficient subscription propagation and event matching. XML is a good candidate, although it lacks typing.

An event is defined using 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 can be in any range defined by the corresponding type. Events themselves can be carried in a byte stream or in compressed format in ODMRP packets. A subset of XPath [2] is used as a subscription language. Complex XPath expressions will be transformed to simplified and unified formats. The subscriptions are tightly linked to the corresponding event data structure in the current system, and an integration with an ontology-based event data model [20] is in progress. Aggregated subscriptions are kept in a compact data structure within subscriber brokers. For compact encoding, Bloom filters are used. The digests of published events (and event advertisements) are transformed similarly into compact data structures that travel, with the event, in a multicast packet header. On arrival at a potential subscriber broker, a matching operation is applied, to the event digest and the subscription, to determine whether to join the subscription to the particular multicast group, and thus to receive further such events. The publishing broker then creates the multicast group from the propagated subscriptions; the challenge is to do this optimally. Brokers can propagate any changes in subscriptions to ODMRP's periodic membership refreshing mechanism.

A. Compact Encoding in Bloom Filters

Subscriptions are aggregated at brokers and transformed into a compact data format. Advertisements and notifications are transformed into a compact format that uses XPath as an intermediate expression during the transformation. For encoding data structures, Bloom filters are used.

Bloom Filters: are compact data structures for probabilistic representation of a set in order 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 probability of a false positive decreases exponentially with linear increase in the number of hash functions and vector size. A Bloom filter is a bit vector, and items are inserted by setting the bits corresponding to a number of independent hash functions.

Subscription and Notification in Bloom filters: Given the constrained MANET environments, it is mandatory to aggregate the set of subscriptions into a compact set of content specifications. The proposed subscription summary structure in Bloom filters is based on the one described in [16], extended to take advantage of XML events, typed by XML schema and the XPath subscription language (for more detail, see [19].) Currently no hierarchical encoding is used. A subscription summary consists of the five data structures described below, and a broker's subscription summary is an array of these data. (Fig. 2 shows an example).

- **Event Type Name (EN):** contains the hashed value of the root element name in the XML schema for the event type.
- **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 sets of subscriptions. Redundant ELs are eliminated. Each row in an EL includes associated EC (Enumeration Constraint), AC (Arithmetic Constraint) and SC (String Constraint).

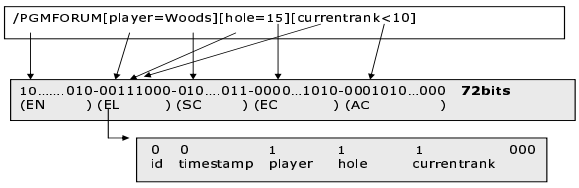

Fig. 2: Encoding Subscriptions in Bloom filters

- **Enumeration Constraint (EC):** holds the constraints of each enumerated string attribute of a subscription.

- **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 sub-ranges of values specified in subscriptions for the specific attribute. The second array is used when an arithmetic constraint has an equality operator for a value that is not in the existing sub-ranges.
- **String Constraint (SC):** contains information about the constraints in the string type. For each different string type element/attribute, that appears in at least one subscription, 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.

B. Global Grouping

Clustering in ad hoc networks often indicates clustering of the nodes, in which case hierarchical routing schemes can manage efficient broadcast. Such clustering schemes are based on the location or topology of the nodes. We attempt clustering of subscriptions based on their semantics. Here, publisher broker nodes define multicast groups from the propagated 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 complexity. Cluster analysis is divided into hierarchical and non-hierarchical methods. K-means [9] 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. In hierarchical methods, the final number of clusters is not known ahead of time. There are two types of hierarchical clustering: *agglomerative* and *divisive*. In agglomerative clustering, each object is initially placed in its own group. The two closest groups are combined into a single group, and so on. In divisive clustering, all objects are initially placed into a single group. The two objects that are in the same group but are farthest apart are used as seed points for two groups. All objects in this group are placed into the new group that has the closest seed. This procedure continues until a threshold distance is reached. Hierarchical clustering may produce relatively good results but it may require maximum $O(N^2)$ calculation time. Established clusters are not easily re-classified, thus hierarchical clustering methods may not be a good choice for incremental conceptual classification.

Our subscription clustering has much in common with subscription partitioning and routing in wide area networks. In [17], there are two approaches to partitioning the overall publish/subscribe operations among multiple servers: either partition the event space or partition the set of subscription filters. In the Event Space Partitioning (ESP) approach, given the number of servers is N_s, the d-dimensional space is partitioned into N_s disjoint subspaces, each assigned to a different server. A subscription is hosted by a server if its filter intersects the server's associated subspace. Filter Set Partitioning (FSP) is a dual partitioning approach that always assigns each subscription to a single server. Similar subscriptions, i.e. subscriptions matched by the same events, are grouped together and assigned to the same server to allow for the construction of compact and effective summary filters. Summary filters from multiple servers may overlap and an event is forwarded to every server whose summary filter contains the event. Fig. 3 depicts the difference between these two approaches for 2-dimensional subscription filters.

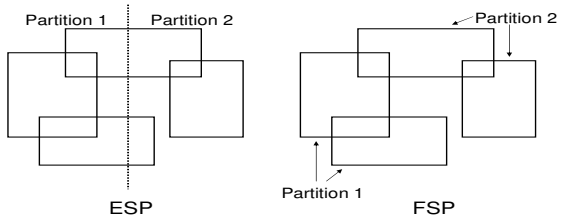

Fig. 3: Event Space Partitioning and Filter Set Partitioning

Our goal is similar to the FSP approach, in that the members of a large set of existing subscriptions are hashed, via their signature strings, into appropriate buckets. A weight is then assigned to each bucket, that is the product of the number of unique strings and the total number of subscriptions falling in that bucket. For content-based publish/subscribe, the FSP approach has demonstrated good load balancing both statically and dynamically, 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 hierarchical clusters, even if the propagated subscription can be part of a constructed hierarchy. Another aspect is that subscriptions may reach the publishing broker nodes over a period of time. 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 the construction of multicast groups more complex. If the groups are to cover coarse-grain subscriptions, more noise will be delivered to the broker, whereas if the multicast groups are to correspond to fine-grain subscriptions, many groups will be created. However, if nodes can have high mobility, using fine-grain subscriptions may reduce the overhead of group membership maintenance. The goal is to define a channel per entity; however 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 optimized methods will depend on the application characteristics. Three different approaches are currently applied in order to obtain heuristic experimental results.

The least constrained subscription: This approach is taken under the assumption that 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. Fig. 4 shows an example of the content subscription graph. When the publisher broker publishes messages it searches all subscriptions in the graph and, if it matches at least one of them, it publishes the message.

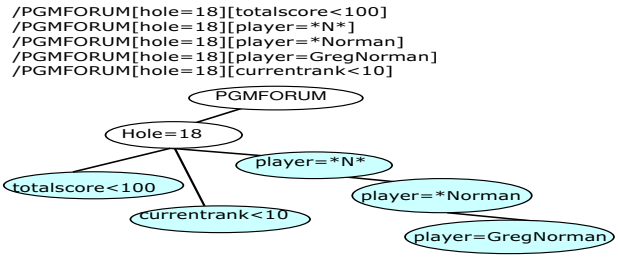

Fig. 4: Content Subscription Graph for Multicast Group

Grouping by K-means Clustering: This approach aims to set effective groups for efficient communication including the splitting/combination of groups resulting from 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. The K-means method is relatively fast, and the calculation time is O(tkn) (n: number of objects, k: number of clusters, and t: number of repeats; normally $t << n$). The ultimate goal is to divide the objects into K clusters such that some metric relative to the centroids of the clusters is minimized. The metric to minimize, and the choice of a distance measure, will determine the optimal cluster structure. Our choice of metric is the sum of the average distance to the centroids over all clusters. We compute the distance measure by selecting the parts of the Bloom filters that represent subscriptions and converting them from XML elements and attributes into real values. We also assign a weight to each object, that can be related to some function of its variance, but we currently assign weight values on an experimental basis. If the weight has approximately the same value for all objects to be clustered, it may not be a valuable indicator and should be small. However, if it varies considerably among the objects, it may be an important discriminator and should be relatively large. The typical operation is:

- Place K points representing initial group centroids among subscriptions.
- Assign each subscription to the group containing the closest centroid.
- When all subscriptions have been assigned, recalculate the positions of the K centroids.
- Repeat Steps 2 and 3 until the centroids no longer move. This produces a separation of the subscriptions into groups from which the metric to be minimized can be calculated.

A basic problem of K-means is that the number of groups has to be known. Thus, some heuristic step is applied. K-means algorithms converge to a local optimum over a number of iterations, and in practice they converge quickly. Nevertheless, the processing can be stopped after any iteration, resulting in a feasible partition into K groups. This also provides an easy way to accommodate changes of grouping, simply running a number of re-balancing iterations when new subscribers arrive or subscriptions change. The EL data and associated SC, EC, and AC data in the Bloom filters (See Section III.A for data types) are used for clustering, and weights are computed for the assigned elements. Note that the event source that triggered the multicast group formation is also an input for K-means clustering. K-means is usually not suitable for non-convex clusters, which makes it difficult to generalize the clustering algorithms. However, one purpose of using the K-means algorithm is to exclude subscriptions that are distant from the main group of subscribers. K is currently computed from the number of elements and attributes in a cluster.

These algorithms perform well for highly heterogeneous subscriptions, and they also scale well. For dynamic subscriptions, iterative clustering algorithms (K-means) seem to be most appropriate. Here, hierarchical clustering algorithms have poorer performance than iterative clustering, but have the advantage of monotone improvement: when more multicast groups become available, the new groups are formed by subdividing the existing ones.

C. Routing by Pub/Sub ODMRP

The routing between a broker and publishers/subscribers can be achieved by unicast protocols. This paper focuses on the routing between the broker publishing an event and the brokers subscribing to it. In general, there are two approaches to 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 preprocessing 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. Our approach is to integrate the latter with an extention of ODMRP that applies scoped flooding.

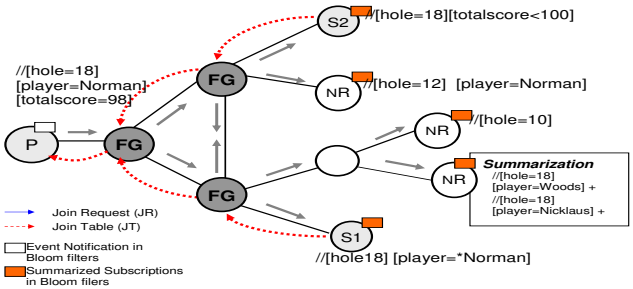

Fig. 5: Pub/Sub ODMRP Operation

The protocol operation of Pub/Sub ODMRP is described below (see Fig. 5). First, a publisher broker node (P) pe-

riodically broadcasts a Join Request (JR) packet over the network. The digest of the event to be sent in the Bloom filter expression is embedded in the JR packet. A node receiving a non-duplicate JR packet stores the upstream node ID and rebroadcasts the packet. The subscriber broker nodes (NR as non-matching subscriber and S as matching subscriber), that keep the subscriptions in Bloom filters, have the correct bits set for the subscription to be recognized at receiver nodes. Once a subscriber broker node decides to join the group, it updates the publisher entry in its member table. A Join Table (JT) packet is broadcast periodically and 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). Optionally the subscription information is kept along with the routing information, and is used as a filter for data forwarding. 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 operates global grouping from aggregated subscriptions.

After the group establishment and route construction process, a publisher broker can transmit 'Data packets' to subscriber brokers via selected routes and forwarding groups. Intermediate nodes relay a 'Data packet' only when it is not a duplicate and its forwarding group membership has not expired. This whole operation minimizes traffic overhead and prevents sending packets along stale routes. See [19] for further details of routing.

IV. Experiments

Evaluation of the proposed system is complex. It is affected by the complexity of subscriptions and events, the efficiency of global subscription aggregation, mobility patterns, network traffic load, the false positive rate of Bloom filters etc. In this paper, we show the basic characteristics of our approach in terms of Pub/Sub ODMRP overhead, clustering effects, and mobility impact (see [19] for more experimental results). We extended a Java based ns-2 simulator adding extended ODMRP. Experimental results have been obtained on the following two data dissemination models:

- our proposed Pub/Sub ODMRP;
- ODMRP (N channels multicast for N subscriptions).

100 nodes with 20 publishers and 50 subscribers are randomly placed over a 1000m x 1000m area. A publisher broker sends 100 messages, that match 60 % of subscriptions. The route refresh interval is set to 5 seconds, and the forward group timeout interval is 25 seconds, chosen as a less influential value for the experimental environment.

A. Conversion Overhead in Pub/Sub ODMRP

The messaging system using Pub/Sub ODMRP requires conversion from messages to XPath expressions. Ultimately both subscriptions and messages in XPath expressions are transformed to Bloom filters. We measured the conversion overhead in relation to the complexity of the XML schema. This was done separately from the network simulation experiments and the result indicates that the conversion overhead stays at a constant value within the range of 200 nesting elements in the XML schema. The comparison process between the message digest and the aggregated subscriptions at the subscriber broker is a fraction of this, since it compares simple bit sets.

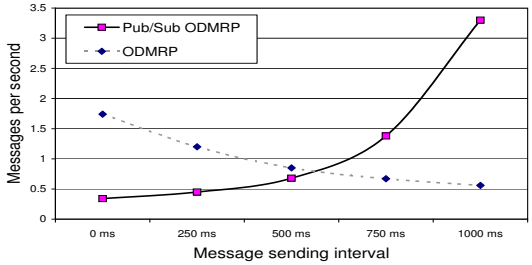

Fig. 6: Throughput Comparison

Fig. 6 shows the throughput of the messaging systems. The throughput is measured when all messages have been delivered to the target subscriber brokers. The conversion time from XPath to Bloom filters is included. Pub/Sub ODMRP improves performance when the message interval cancels out the conversion overhead. ODMRP's maximum throughput is lower than the average throughput of Pub/Sub ODMRP.

B. Clustering Method Comparison

Fig. 7 shows Algorithmic comparisons for global clustering in publisher brokers. K-means creates more channels than the graph form of aggregated subscriptions but the overall traffic over the networks is reduced. This shows that K-means provides efficient grouping.

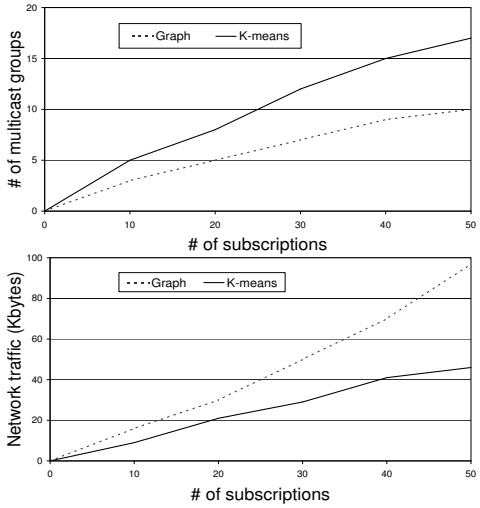

Fig. 7: Cluster Algorithms Comparisons

C. Mobility and Reliability

The proportion of messages delivered correctly is related to node mobility in Fig. 8. Random way-point is used in the simulation for the node mobility pattern. When node mobility is high, loss of messages is inevitable. Note that the delivery ratio is high at human walking speed (5km/hr). The delivery ratio of Pub/Sub ODMRP shows similar values to ODMRP. See [19] for more experiments.

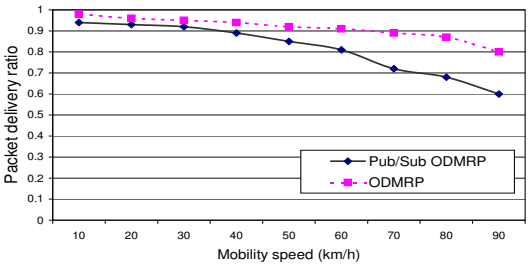

Fig. 8: Message Delivery Ratio

V. Related work

In the past decade many publish/subscribe systems have been reported, including CEA [1], SCRIBE [14], and SIENA [4]. SCRIBE is a topic-centric publish/subscribe messaging system using Distributed Hash Tables (DHT) over Pastry [15]. Several publish/subscribe systems, e.g. SIENA, implement some form of content-based routing. SIENA is a notification service scalable to wide-area networks. Routing strategies in SIENA use two classes of algorithm: advertisement forwarding and subscription forwarding. SIENA has an extension to support mobility by an explicit operation to relocate clients. None of these systems support extremely dynamic mobile environments such as mobile P2P networks.

Several middleware systems have been developed to support wireless ad hoc network environments, e.g. STEAM [10] and IBM WebSphere MQ [7]. STEAM provides a proximity-based group communication. These systems construct publish/subscribe above existing transport protocols. Our approach uses cross layering between middleware-tier and network components. In wireless ad hoc network environments, much research currently focuses on datagram routing in both unicast and multicast routing. However, no definite solution to define publish/subscribe semantics using these protocols has been provided. Given the characteristics of mobile ad hoc networks, the use of distributed hash tables in a messaging system to locate objects may not work in dynamic environments. Thus, our approach is unique in taking advantage of the broadcast nature of wireless ad hoc networks and creating dynamic multicast channels by grouping subscriptions.

VI. Conclusions and Future Work

In this paper, we extended multicast communication to a publish/subscribe semantic cast mechanism. The MANET environment is highly dynamic, and communication optimization needs to be done by means of a cross-layer approach. Our key contribution is the creation of an interface between the middleware tier and network components. Content-based subscriptions in a compact data format (Bloom filters) travel within multicast headers, and ODMRP uses this context to construct an optimized dynamic dissemination mesh. A dynamic multicast group is created and maintained by aggregated subscriptions over the network by applying K-means clustering and other methods. In dynamic MANET environments, the multicast channel lifetime is ephemeral and subscriptions are expected to be more specific. In such network environments, our approach would achieve efficient group communication. Cooperation between middleware-tier and network components provides fine-grained subscriptions. Experimental results highlight that performance is improved by introducing a publish/subscribe scheme; the network traffic is reduced and the reliability is improved by the mesh topology. The evaluation could address other aspects such as more complex subscriptions and the brokers' topology, and more complex experiments are in progress. A complete event-based middleware involves more than supporting asynchronous communication and we are working on issues such as reliability, persistence, fault-tolerance, and security.

Acknowledgment. This research is funded by EPSRC (Engineering and Physical Sciences Research Council) under grant GR/557303.

References

[1] Bacon, J. et al. Using Events to build Distributed Applications. *Proc. IEEE SDNE*, 1995.
[2] Berglund, A. et al. XML Path Language (XPath) 2.0. Working Draft. *W3C, http://www.w3c.org/TR/xpath20/*, 2001.
[3] Bloom, B. Space/time Trade-offs in Hash Coding with Allowable Errors. *CACM, 13(7)*, 1970.
[4] Carzaniga, A. et al. Achieving scalability and expressiveness in an Internet-scale event notification service. *Proc. 19th ACM Symp. on Principles of Distributed Computing*, 2000.
[5] Chiang, C. et al. On-Demand Multicast Routing Protocol. *Proc. WCNC*, 1999.
[6] IBM. Gryphon: Publish/Subscribe over Public Networks. *http://researchweb.watson.ibm.com/gryphon/gryphon.html*.
[7] IBM. WebSphere MQ: Connecting your applications without complex programming. *WebSphere White Papers*, 2003.
[8] Lee, S. et al. A Performance Comparison Study of Ad Hoc Wireless Multicast Protocols. *Proc. IEEE INFOCOM*, 2000.
[9] MacQueen, J. B. Some Methods for classification and analysis of multivariate observations. *Proc. 5th Berkeley symposium observation, Pro-Statistics and Probability*, 1967.
[10] Meier, R. et al. STEAM: Event-based middleware for wireless ad hoc networks. *Proc. DEBS*, 2002.
[11] Muhl, G. et al. Filter Similarities in Content-Based Publish/Subscribe Systems. *Proc. ARCS*, 2002.
[12] The Narada Event Brokering System . *http://grids.ucs.narada.edu/ptliupages/projects/narada.*
[13] Ratnasamy, S. et al. Application-Level Multicast using Content-Addressable Networks. *Proc. NGC*, 2001.
[14] Rowstron, A. et al. SCRIBE: The design of a large-scale event notification infrastructure. *Int'l. Workshop of NGC*, 2001.
[15] Rowstron, A. et al. Pastry: Scalable Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems. *Middleware*, 2001.
[16] Triantafillou, P. et al. Subsciption Summaries for Scalability and Efficiency in Publish/Subscribe Systems. *DEBS*, 2002.
[17] Wang, Y. et al. Subscription Partitioning and Routing in Content-based Publish/Subscribe Systems. *Proc. 16th Symp. on Distributed Computing*, 2002.
[18] Yoneki, E. and Bacon, J. Gateway: a Message Hub with Store-and-forward Messaging in Mobile Networks. *Proc. ICDCS-MCM*, 2003.
[19] Yoneki, E. and Bacon, J. An Adaptive Approach to Content-Based Subscription in Mobile Ad Hoc Networks. *Proc. PerCom-MP2P*, 2004.
[20] Yoneki, E. et al. Towards Peer-to-Peer Event Broker Grid in Hybrid Network Environments. *Proc. DOA-GADA*, 2004.
[21] Zhuang, S. et al. Bayeux: An architecture for scalable and fault-tolerant wide-area data dissemination. *Proc. NOSSDAV*, 2001.