Integrated Data Location in Multihop Wireless Networks

Irfan Sheriff, Prashanth Aravinda Kumar Acharya, Ashwin Sampath,
Ben Y. Zhao and Elizabeth M. Belding
Department of Computer Science, UC Santa Barbara
{isheriff, acharya, ashwins, ravenben, ebelding}@cs.ucsb.edu

Abstract— **Multihop wireless networks are ideal as infrastructures for location-aware network applications, particularly for disaster recovery operations. However, one missing component is an efficient and scalable distributed data location service. Existing approaches impose significant communication overhead on the underlying wireless layer and generally limit the total number of locatable objects in a network. To address this problem, we present the Integrated Data Location Protocol (IDLP), which provides scalable location of a large number of objects by integrating compressed summaries of object signatures into the routing layer. We evaluate our approach using extensive simulations in Qualnet, as well as detailed measurements from a deployed AODV-implementation on the UCSB MeshNet testbed. Results show that IDLP maintains low communication overhead while efficiently locating up to a hundred objects per node.**

I. Introduction

Spontaneous wireless ad hoc networks are a flexible extension to wired infrastructure. Wireless routers can be quickly deployed over poles, buildings or even transport vehicles, and require minimal existing infrastructure. In the wake of Hurricane Katrina, multihop wireless networks have emerged as critical communication infrastructure during emergency disaster recovery. With wireline phone service and cellular networks disabled by Katrina, a wireless network designed for video surveillance was the only functional communication system remaining. During Katrina's recovery, the VoIP traffic carried by the mesh network represented a lifeline for struggling local businesses. The wireless mesh network is generally credited for the population growth from 50,000 to 250,000 from November 2005 to February 2006[1].

Timely communication and information sharing is critical to the success of any emergency or disaster recovery operation. While years of research have focused on reliable and efficient wireless routing techniques, much less attention has been paid to wireless infrastructure support for information sharing such as data location and distributed search. In addition to location and access to mobile data, these techniques can also be applied to locate objects and resources based on their proximity to wireless nodes or access points. For example, emergency workers might search for the location of resources such as food, medical supplies and clothing, rescue equipment such as ambulances and flashlights, or individual people such as fire-fighters or military personnel. Nodes in an ad hoc wireless network can track these resources by their relative proximity, and make their location available to users in the network.

[1]http://www.infoworld.nl

While data location is a well known problem, it faces new challenges in the context of multihop wireless networks. First, multihop wireless networks present a dynamic environment due to intermittent device connectivity, unreliable wireless links and mobility in the network. Hence, solutions based on centralized service agents or conceptually static entities such as directory agents will provide highly unreliable results. Second, without a wired infrastructure and network structure, object location on multihop ad hoc networks can incur unacceptable control traffic overheads with application layer techniques. Third, the bandwidth constrained nature of multihop wireless networks implies a trade-off between control protocol overhead and search accuracy. An ideal solution, thus, should adopt a concise object representation technique that supports unique object identification while imposing minimal control protocol overhead.

In this work, we propose Integrated Data Location Protocol (IDLP), a light-weight distributed object location solution for multihop wireless networks. The IDLP architecture attempts to minimize control protocol overhead; handles the constant node churn that is characteristic of multihop wireless networks; and addresses the bandwidth limitations and unreliable links of a wireless environment. In IDLP, each node creates a data signature of its objects and advertises these signatures throughout the network by leveraging routing control packets. Similar advertisements from other nodes are associated with routing entries and stored in the routing table. Object queries generated by applications are searched locally within the routing table amongst data signatures before invoking a network-wide search for a provider node. This technique utilizes the partial topology information available at the nodes and minimizes network-wide search operations. Together, these techniques allow us to locate objects within a decentralized wireless network while maintaining low communication overhead.

The rest of the paper is organized as follows. First, we provide background on existing content location schemes in Section II. We then present the design of an Integrated Data Location Protocol (IDLP) in Section III. We evaluate the performance of IDLP through Qualnet simulations in Section IV, followed by results obtained from a deployment of IDLP on the UCSB MeshNet testbed in Section V. Finally, we summarize and conclude in Section VI.

II. Related Work

Significant work has gone into the design of scalable service discovery systems for the wired Internet. Some solutions, such as LDAP [1], JINI [2] and SLP [3], rely more on centralized client-server architectures. Others, such as DNS [4], Globe [5] and the Berkeley Service Discovery Service (SDS) [6], leverage hierarchical architectures to scale to large numbers of records and across network domains. In particular, the Berkeley SDS uses Bloom filters [7] to summarize service descriptions across network domains in order to route queries to the appropriate local servers. Similarly, Bloom filters are also used to locate data objects in both the Summary Cache [8] and the Probabilistic Data Location [9] projects. Unlike these projects, we cannot rely on a stable underlying network topology to build state for query forwarding. In addition, interference on wireless links and the resulting bandwidth constraints limit the amount of acceptable communication overhead for data or service location.

A number of service location systems for wireless networks work at the application layer. Some use a decentralized approach based on peer-to-peer (P2P) caching [10]–[13], while others use a hybrid of P2P and directory-based architectures [14]–[16]. Still others push service discovery down the network stack in order to work more closely with the routing layer [17]–[20].

In [16], the authors adopt a two phase hybrid architecture similar to the Kazaa file-sharing network [21]. Local directories (supernodes) aggregate service records and resolve local queries. When local resolution fails, a directory forwards the query towards other directories likely to have relevant information. Query forwarding is guided by aggregate summaries of service records using Bloom filters, identical to the way query forwarding is performed in the Berkeley SDS [6]. However, mobility patterns of the underlying nodes require deleting records from Bloom filters, an expensive and difficult task.

Ekta [15] performs data location on wireless networks by layering a distributed hash table (DHT) on top of the Pastry [22] P2P network. To locate an object, it must first be moved to the wireless node whose Pastry identifier matches closest with the object. If metadata is stored instead of the object itself, a client node must use P2P routing to locate the metadata, and then redirect to the object. Despite the integration of DSR [23] source routes, the resulting query redirection, multihop overlay routing, and DSR route maintenance traffic can impose significant overhead on the wireless nodes.

The approaches in [20] and [17] embed service records inside ZRP [24] route advertisements and ODMRP [25] multicast advertisements, respectively. Nodes are required to know the service to UUID mapping or the service to multicast group mapping, a priori. These mechanisms work for discovering small sets of well-defined services but are ill-suited for the general problem of data location, where a priori mappings are not possible. In addition, the UUID approach limits the number of data objects searchable (at most 16 in [20]), and the proactive nature of local ZRP advertisements increases the cost of distributing service records. In contrast, our work specifically addresses the challenge of representing a large collection of data objects, while our integration with AODV means service summaries are distributed on-demand. The choice of an on-demand routing protocol works well for mobile wireless networks [26].

Finally, the Geography-based Content Location Protocol (GCLP) [27] is a push-based technique where nodes periodically publish object information by forwarding the information in four directions across the network. Nodes on the publish path cache these records. Other nodes perform object location by forwarding a query in all four directions. Nodes at the intersection of the publish and query paths use their local caches to resolve queries. We compare our work against GCLP because it is one of the most popular and efficient of the existing approaches.

III. Design

We now describe the Integrated Data Location Protocol (IDLP). To support data location while minimizing communication overhead, IDLP embeds location information compressed as *data signatures* into routing control messages in the network. Nodes store data signatures along with routing entries in the routing table and use them to resolve data location queries.

A. Data Location and Routing Integration

Multihop wireless networks present a dynamic environment characterized by nodes joining/leaving the network, node mobility, changes in the topology and variable link quality. Note that this is a fundamentally different problem from data location on static networks, where approaches such as directed diffusion [28] can optimize routes to data over time. These properties make wireless routing an inherently expensive process. Popular reactive routing protocols such as AODV [29] and DSR [23] require a network-wide flood of a *route request* message to discover routes to an unknown node.

As a baseline strawman approach to data location in a multihop wireless network, we consider the simple solution of broadcasting the query across the network (similar to a *route request* message). With this *query flooding* approach, we can simultaneously build a route and resolve a query. This simple solution is guaranteed to work and has the same cost as a single route discovery operation. Thus, any new solution must resolve queries at a lower cost.

This is a difficult challenge for application-level solutions. A single extra application-level hop could require a new route discovery, resulting in a network-wide broadcast. The resulting traffic would impose more overhead than the strawman solution described above. For example, a client using a directory-based approach must send a message to the directory server, then possibly contact another server for the actual object or information. Each of these steps may result in a route discovery cycle and thus may be more expensive than a complete query using query flooding. Even with caching enabled at the directory server, client mobility is likely to change routes to the directory server, triggering further route discovery events. Similarly, query resolution using structured peer-to-peer overlays [22], [30] involves multiple hops on the overlay network. Each overlay hop may result in a new route discovery operation and thus may be more expensive than a complete query using query flooding. In general, all application level approaches that assume stable network routing will likely incur much higher messaging overhead compared to the naïve query flooding approach.

Therefore, an efficient data location system must be aware of the availability of routing state at the network level. This calls for a cross-layer approach that integrates data location and wireless routing. IDLP integrates a data location component into the routing protocol by piggybacking data location information onto routing control messages. Routing control messages are used as vehicles to disseminate the data location information of each node into the network, eliminating the need for explicit *publish* messages. Client nodes resolve queries by accessing the data location information in their routing tables.

B. Compact Data Signatures

The integration of data location into the wireless routing level requires a compact representation of data objects that imposes the minimum amount of storage and communication overhead. The data representation must be able to efficiently encode the presence of multiple data objects per node, and not impose a significant overhead as the number of objects grow in the wireless network.

Given the stringent storage and communication constraints of wireless networks, we choose to support only data location via a unique name, rather than more multi-field queries such as those supported by LDAP [1] or the Berkeley SDS [6]. For application of IDLP to service discovery, we assume the presence of a canonical ontology that maps generic terms into a single name. For example, queries for "fire fighter" or "fireman" would map into "fireman." Such an ontology mapping can be provided by the wireless service provider at the application level. In addition, location-based modifiers such as "find the nearest X" can be resolved by augmenting query results with GPS positioning information.

The key challenge is to find a scalable, compact representation of data object names. In a network where each wireless node can offer multiple data objects or locate multiple services, we need a mechanism whose overhead increases in size with the number of nodes in the network, not the number of objects per node. Each node can then serve a number of objects or services using metadata of a constant size, or a unique *data signature* that represents the contents of the directory listing.

Bloom filters [7] provide an ideal solution for our needs. They compress a set of elements into a fixed size string and support set membership queries on the string. Generating and querying Bloom filters are efficient operations, and the resulting string can be extremely compact. Bloom filters allow lossy compression, and provides a tunable knob in the tradeoff between filter size and query resolution accuracy. In IDLP, we encode multiple objects on a single node into one Bloom filter, and piggyback it onto routing control messages for dissemination.

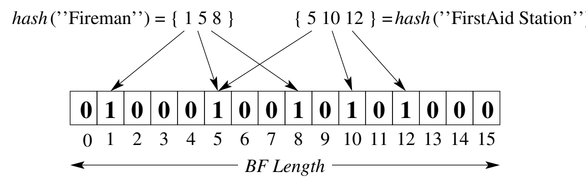


Fig. 1. Hashing multiple object names into a single Bloom filter.

Before we discuss the use of Bloom filters in IDLP, we first outline their operation. A Bloom filter [7], [8] is a method of representing a set $A = \{a_1, a_2, \ldots, a_n\}$ of n elements to support set membership queries, *i.e.*, to check whether an element a_x is a member of A. The Bloom filter consists of a vector v of m bits and a set of k independent hash functions, $h_1, h_2, \ldots, h_k$, each with an output range $\{1, \ldots, m\}$.

The construction of a Bloom filter that represents the set A is done as follows: For each element $a \in A$, the bits at positions $h_1(a), h_2(a), \ldots, h_k(a)$ in v are set to one. Thus a bit at position n in the Bloom filter is set if there is at least one element a_x in A such that $\exists i, 1 \leq i \leq k,\ h_i(a_x) = n$. All other bit positions are set to zero. We show an example of Bloom filter construction in Figure 1 where $k = 3$, $m = 16$.

Given a Bloom filter B, we can check for the existence of an object b by examining the value of the bits in the Bloom filter at positions $h_1(b), h_2(b), \ldots, h_k(b)$. If any of the bits is set to zero, then b is definitely not present in the set represented by B. On the other hand, if all the bits are set to one, then it is probable that the object b exists in the set represented by B. However, it is possible that even with all the respective bits set, the object may not be present in the set, an occurrence of a *false positive*. The false positive rate can be controlled as per the requirements of the application by choosing the appropriate number of hash functions k, the size of the Bloom filter m, and taking into account the average number of objects used in the construction of each Bloom filter.

Note that we are choosing larger Bloom filter sizes to minimize the occurrence of false positives in data signatures. With an appropriate choice of k and m, Bloom filters can provide "near-lossless" compression. Therefore, our compression gain from Bloom filters is not from lossy compression, instead we exploit Bloom filters mainly as a compact representation of a dynamic dataset. Since it is not possible *a priori* to know the number of objects in the network or to assign them sequential identifiers, the Bloom filter signatures act as a compact naming scheme to accommodate a dynamically changing set of objects.

In Section IV, we analyze the effects of the Bloom filter parameters on the performance of IDLP and list the values of k and m used in our implementation of the protocol. We now describe the use of Bloom filters in IDLP and the propagation of the Bloom filters in the network.

C. IDLP Operation

IDLP resides above the routing protocol in the protocol stack of each node in the network. The protocol accepts and handles object location queries from a querying application. Queries from the application are assumed to be a unique string identifier for the object, *i.e.* a file name, URL or service name. IDLP is also responsible for responding to object location queries from peer nodes in the network.

a) Data Signature Initialization: At each node, the initialization phase of the protocol involves the construction of the data signature that represents a summary of the set of objects shared by the node. The data signature is constructed using the object identifiers of all the shared objects. The identifiers of the shared objects can be filenames, URLs, or service names. These identifiers form the set elements $a_1, a_2, \ldots, a_n$ of the Bloom filter construction described in Section III-B. This data signature is then passed to the underlying routing protocol. The routing protocol incorporates the data signature in its control messages, thus propagating it throughout the network. While IDLP is compatible with any reactive wireless routing protocol, we chose AODV [29] as the underlying protocol for our experiments in this paper.

Figure 1 shows an example of the construction of a data signature of a node that has two objects "Fireman" and "FirstAid Station." In this example the node uses a 16 bit Bloom filter and three hash functions. The bit positions for each object are calculated and then combined to produce the data signature for the node.

b) Data Signature Propagation: To illustrate the propagation of IDLP data signatures, we assume a scenario using AODV as the wireless routing protocol. A typical AODV routing table entry has the following structure: <*destination addr, next hop addr, route metric (hop count), sequence #, interface ID*>. We augment the entry with an additional field associated with each entry: the destination node's IDLP signature.

Route discovery in AODV is based on a network-wide broadcast of the Route Request (RREQ) message, initiated by the source node. In IDLP, the RREQ message includes the data signature of the originating node. All the nodes in the network receive this message and cache the routing information and data signature for the source node. When the destination receives this RREQ message, it responds with a Route Reply (RREP) message that includes its own IDLP data signature. This RREP message is unicast along the reverse path of the route. All nodes along this path record the routing information and the destination node's IDLP data signature. Thus, in a network with many active flows, each node has several routing entries with the corresponding data signatures for those destinations. In addition, if AODV's HELLO messages are enabled, each node's routing table contains route entries and IDLP signatures for all of its neighbors. This process of route discovery through the RREQ, RREP, and HELLO messages facilitates the propagation of the IDLP data signature in the network without the need for explicit control messaging.

c) Location Query Resolution: We now outline the sequence of events and actions associated with resolving an *Object Location Query*. Assume that a node is trying to locate an object with identifier O. IDLP then calculates the bit positions $p_1 = h_1(O)$, $p_2 = h_2(O), \ldots, p_k = h_k(O)$ in a Bloom filter for this object. Note that hash functions $h_1, h_2, \ldots, h_k$ are the same globally known hash functions used to generate the data signature during initialization. IDLP then searches the current AODV routing table and selects all the nodes whose data signatures have the bit positions p_1,$p_2, \ldots, p_k$ set to 1. These nodes form the *global candidate list* and represent the set of all the nodes that may have the object O.

From this global candidate list, we designate a subset of nodes as *candidate nodes*. We describe the criteria for candidate node selection in Section III-D. The local node then sends a unicast query message for object O simultaneously to each candidate node. Upon receiving the query message, each node sends a response to the querying node about the availability of the object. In other words, nodes that have the requested object respond positively and others respond negatively.

If no positive response is received at the querying node within a timeout period, a network-wide broadcast query is initiated. On receiving this broadcast query message for the first time, a node that has the requested object sends a unicast response to the originating node as described above. If it does not have the object, the node forwards the message to its neighbors. Duplicate copies of the broadcast query message are discarded. A querying node that receives multiple positive responses, during either the unicast query phase or the broadcast query phase, can choose among the responses to select the closest node.

Figure 2 depicts a scenario that illustrates the query process. Consider a disaster area with an ad hoc network that uses IDLP to provide a location-aware search facility. As shown in the figure, the ad hoc network is comprised of several geographically distributed wireless routers. Assume that the wireless routers employ IDLP with an eight bit-two hash function Bloom filter. A rescue worker, connected to router M, wishes to locate a medicine kit closest to him and sends the corresponding query to M. On receiving this query, M computes the bit positions for "Medicine Kit" and examines its routing table to select the candidate nodes. Router M determines that routers G, P and B are the closest routers that may have a medicine kit nearby. It then unicasts queries to each of these nodes. Routers G and P respond positively with information about the medicine kits close to them. Router G, with the smaller hop count, is chosen as the closest router[2]. Note that B is an example of a false positive. In addition, since there is an upper bound on the number of unicast queries sent, some matching entries (such as H) will be omitted.

[2]In this example, we assume that the network distance (hop count) is a good indicator of the geographic distance. In a practical deployment, additional information such as GPS coordinates of individual routers can be used to select the closest router.

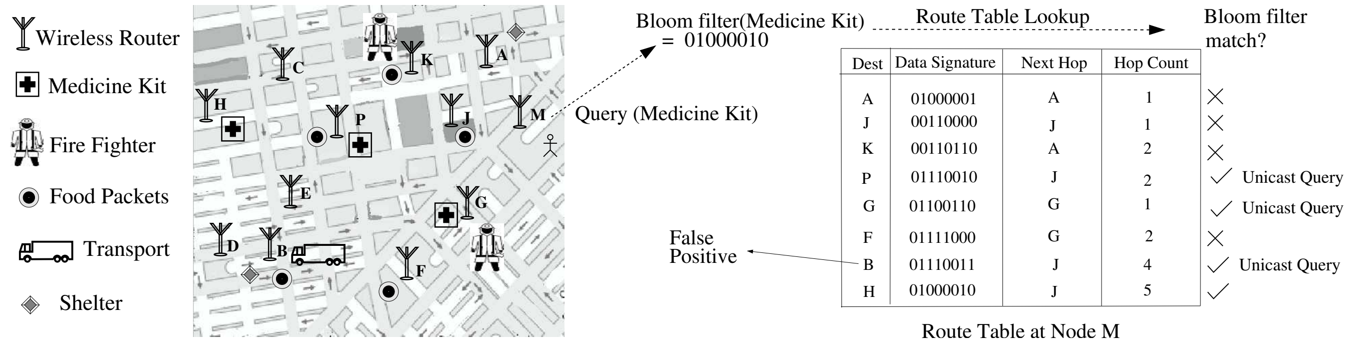


Fig. 2. A rescue worker connected to wireless node M contacts M with a query for the nearest medicine kit. M uses IDLP to resolve the query, finds four matching entries, and issues unicast messages to three of them. Map courtesy of http://maps.google.com.

D. Candidate Node Selection Metrics

Candidate nodes are selected from the routing table to receive the simultaneous unicast queries. The purpose of sending unicast queries is to minimize the network overhead associated with a network-wide broadcast. If a route to a node that has the object is already known, that node is queried, thereby preventing a new broadcast route discovery. Additionally, the selection of more candidate nodes helps reduce the effect of false positives on the query resolution process. By sending unicast queries to more nodes, the probability of failure of the unicast query phase is significantly reduced. However, a large number of unicast messages may cause the same overhead as a single broadcast message. In Section IV, we investigate the the choice of number of simultaneous unicasts and its impact on the overhead in the network.

The selection of the candidate nodes from the global candidate list is based on the following parameters: hop count of the routing entry, and the timestamp of the routing entry. To ensure the locality property of the search, the candidate nodes are selected based on their distance from the querying node. Thus, the hop-count field in the routing table entry is used to select candidate nodes with the least hop-count values. The timestamp on the routing entry is used as the secondary key to select the more recent routing entry when two nodes are at the same hop count distance from the querying node.

E. Node and Object Churn

Node churn refers to the phenomenon of nodes joining and leaving the network. IDLP leverages the behavior of the routing protocol to handle the node churn in the network. The routing protocol is aware of nodes that leave/join the network and correspondingly updates the routing table. This continuous process of table updates maintains the latest view of the network and up-to-date data signatures. IDLP is thus able to function well even in a dynamic network.

Object churn refers to a change in the set of shared objects. When the set of objects shared at a node changes, its IDLP data signature also changes. IDLP pushes updated data signatures to the routing protocol for inclusion with future messages. In addition, the IDLP node proactively sends the updated data signature to other nodes in the network that have a route to it. The forwarding mechanism for this message is similar to the propagation of *link break* (Route Error) messages in AODV and has smaller overhead compared to a network-wide flood of the update message. We limit the rate of proactive publishing in order to aggregate multiple objects churns at a node and reduce the network overhead.

IV. Simulation Performance

We evaluate IDLP on the Qualnet simulation platform [31] under a variety of system parameters and environmental conditions. We begin this section by describing the parameters and metrics chosen to evaluate IDLP. Next, we discuss the exploration of tradeoffs to determine our system parameters, and we evaluate the impact of environmental factors on IDLP. Finally, we compare the performance and overhead of IDLP with the Geography-based Content Location Protocol (GCLP) [27] and the simple query and flood approaches.

A. Parameters and Metrics

Simulations were carried out with 100 nodes in a network area of 2000m X 2000m with IEEE 802.11b as the MAC layer. The transmit power, receiver sensitivity and thermal noise factor were chosen so as to achieve a reliable packet reception range of about 200 meters. The network was static except during the experiments that explicitly consider mobility. AODV was used as the routing protocol. The Bloom filter hash functions chosen were MD5, SHA1 and SHA256[3]. Tunable system parameters include:

Object distribution: Object names are chosen from a list of 1,000,000 unique Internet URLs. Randomly selected objects are distributed across nodes. The number of objects per node varies from 10 to 300, and the replication factor per object (number of replicas in the network) varies from 1 to 20.

Maximum number of simultaneous unicasts (η): As the number of simultaneous unicasts increases, redundancy is introduced to accommodate the false positives generated by

[3]The optimal number of hash functions, k, to minimize false positives is $\approx \ln 2 * \frac{m}{n}$, where m is the number of Bloom filter bits and n the number of inserted elements [8]. The parameter n is not known in advance. We therefore fix the number of hash functions to be three for our analysis. We choose these hash functions to achieve an even distribution of output across the result space. The functions do not have to be cryptographically secure.

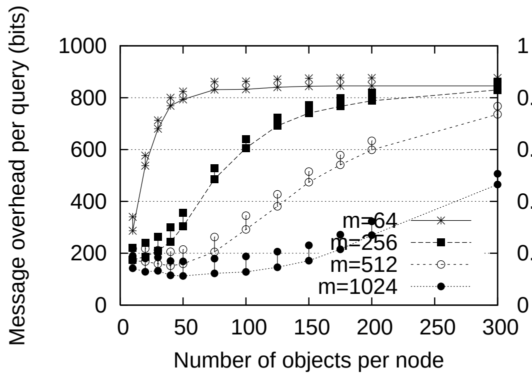


Fig. 3. Impact of Bloom filter length m on per-query communication overhead.

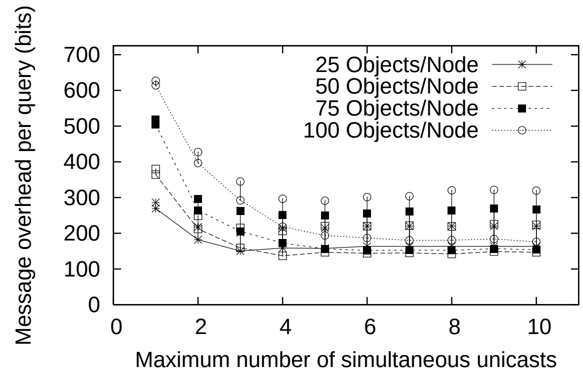


Fig. 4. Impact of the maximum number of simultaneous unicasts (η).

the Bloom filter. We refer to the maximum number of allowed simultaneous unicasts as η. Increasing η increases the unicast overhead and reduces the fraction of queries that require a broadcast for resolution.

Bloom filter length (m): The length of the Bloom filter, m, determines the false positive rate. We must choose an appropriate value of m to support a specific number of objects per node. A large Bloom filter length results in higher unicast success rates but incurs greater store and dissemination overheads.

Routing state: IDLP disseminates data signatures through route control messages. The amount of background traffic in the system affects the amount of routing state present on the nodes, and consequently the likelihood of finding a match from local routing entries. For our analysis, we assume a traffic model where at any given time, 50% of the nodes have an active flow to another random node in the network.

We evaluate IDLP by examining both its performance and its overhead on the system. The metrics for evaluation include:

Query overhead: Queries are generated for all the objects in the network, and query source nodes are chosen randomly in the network. *Message overhead per query* is the average per node overhead introduced by a query. The per node overhead represents the network-wide overhead averaged out per node. The message overhead includes the overhead caused by unicasts and broadcasts. *Broadcast fraction per query* represents the fraction of the total queries that result in a broadcast.

Publish overhead: Publish overhead is the amount of messaging overhead in bits per second required to distribute object information in the network. We evaluate the publish overhead generated by IDLP.

End-to-end delay: End-to-end delay is the amount of delay encountered from the time of query initiation to the time of indication of a success or failure. The end-to-end delay should be within acceptable limits for a data location system to be deployable.

B. Performance Tradeoffs

The choice of the Bloom filter length, m, and the maximum number of simultaneous unicasts, η, impact the performance of IDLP. We first examine the impact of m and η, keeping all other system parameters fixed. Simulations were conducted with the system parameters specified in Section IV-A. The object replication factor is five.

1) Bloom Filter Length: Figure 3 shows a plot of the message overhead per query with the number of objects per node for different Bloom filter lengths. The connected data points on the curve represent the average broadcast overhead incurred per node per query. The vertical line for each data point on the curve represents the corresponding average per node unicast overhead per query. The top of each line thus represents the total per node overhead introduced by a query. The y-axis on the right shows the broadcast fraction per query corresponding to each data point on the curve.

We observe from the graph that the unicast overhead forms a small fraction of the total overhead per query. Total message overhead depends largely on the fraction of queries that result in broadcast. The broadcast fraction remains below 20% in the region of low false positive rate. This region can be seen in the Figure 3 to be up to 50 objects with a 512 bit Bloom filter length and up to 150 objects with a 1024 bit Bloom filter length.

It is interesting to observe that the broadcast fraction per query does not reach zero even when the false positive rate is close to zero. The 16% minimum broadcast is due to the absence of complete routing information at the nodes. This minimum broadcast fraction depends on the amount of routing information in the network. Additionally, the maximum broadcast fraction per query does not reach 100% even when the false positive rate is close to 100%. This is because the selection of nodes for unicast in this region is random and a fraction of the unicast queries succeed with random selection of nodes.

The publish overhead is independent of the number of objects in the network and depends on the Bloom filter length (m) and the traffic in the network. For the traffic model used in the simulations, and assuming the average length of a flow is roughly one minute, we determined that a 512 bit Bloom filter results in a publish overhead of roughly 2 kbps across the network and can support 50–75 objects with 80–90% of the queries successfully resolved through unicasts. A 1024 bit

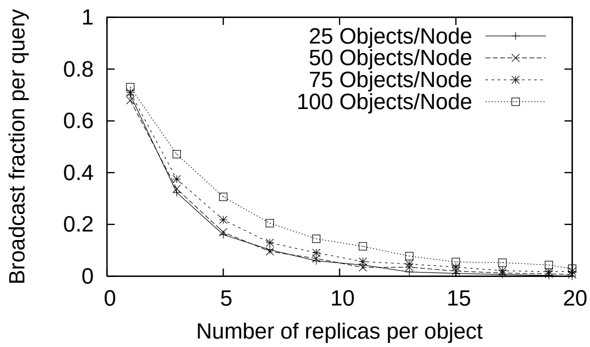


Fig. 5. Impact of the object replication factor on fraction of queries requiring broadcasts.

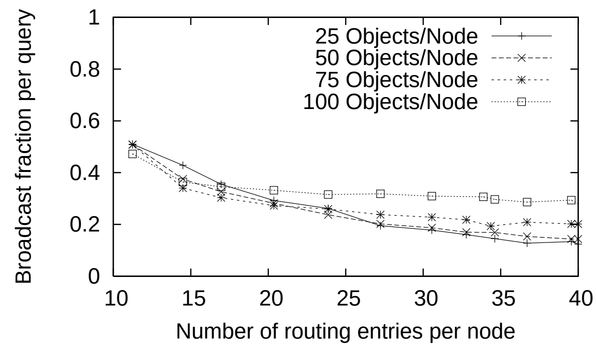


Fig. 6. Impact of per-node routing state.

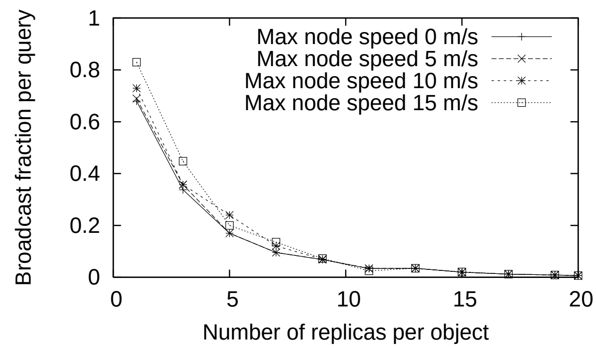


Fig. 7. Impact of mobility.

Bloom filter results in a publish overhead of about 4 kbps and can support 150–175 objects per node with 80–90% of the queries resolved with unicasts. Clearly there is a trade-off between the overhead and the supportable number of objects per node in the network. We expect about 50–75 objects for our target application scenario and hence choose m=512 for the remaining analysis.

2) Maximum Number of Unicasts: We now examine the impact of the maximum number of simultaneous unicasts (η). Figure 4 plots the message overhead per query with η. The data points represent the average broadcast overhead per query while the vertical line represents unicast overhead per query. It is seen that the broadcast overhead per query decreases as η increases. When η is low, the fraction of queries that result in broadcast is high. Additionally, when the number of objects per node is low, the false positive rate is lower and hence the broadcast overhead for 25 objects/node is less than for larger number of objects. As we increase the maximum number of simultaneous unicasts, the false positives of individual unicasts are covered with redundant unicasts and hence all the curves reach a common low broadcast overhead per query.

The unicast overhead shown by the length of the vertical lines indicates that the unicast overhead increases as η increases. However, note that when the false positive rate is higher (100 objects/node), the increase in the unicast overhead is much larger as η increases. This shows the importance of appropriate selection of Bloom filter length. The maximum unicast overhead is limited by the amount of routing information available at the nodes.

It is thus clear that a low value of η (<2) can cause a large fraction of the queries to result in broadcast. We observe from the graph that about three unicasts can compensate for the false positives created by the Bloom filter. Hence we choose $\eta = 3$ for further analysis of IDLP.

C. External Factors

We now examine the impact of the environment driven factors such as the object replication, user traffic and mobility on the performance of IDLP.

1) Impact of Object Replication: Figure 5 shows that there is an exponential decrease in the broadcast query fraction with increased replication of objects in the network. More replicas in the network result in an increased chance of a routing entry to a node containing the object. In addition, the fraction of entries in the routing table that contain the object increases. When the replication is sufficiently large (>10), the false positive rate has little effect on the fraction of queries that result in broadcast. A large fraction of the queries ($\approx$ 95%) succeed during the unicast phase.

2) Impact of User Traffic: Since the amount of user traffic determines the routing information available at nodes in the network, we evaluate IDLP with the routing state per node. Figure 6 represents the effect of routing state per node on the fraction of the queries resulting in broadcasts. When the false positive rate is low (25 and 50 objects per node in the graph), additional routes at a node result in a greater fraction of the queries succeeding during the unicast phase. However, at a higher false positive rate (100 objects per node), the probability of choosing the false positive entries increases and hence the existence of more routing information has little effect on the fraction of queries that succeed during the unicast phase. This shows the importance of choosing an appropriate value of m to suit the application requirements. A 512 bit Bloom filter can support about 50-75 objects per node.

3) Impact of Mobility: We performed mobility experiments using the random waypoint model in Qualnet with different

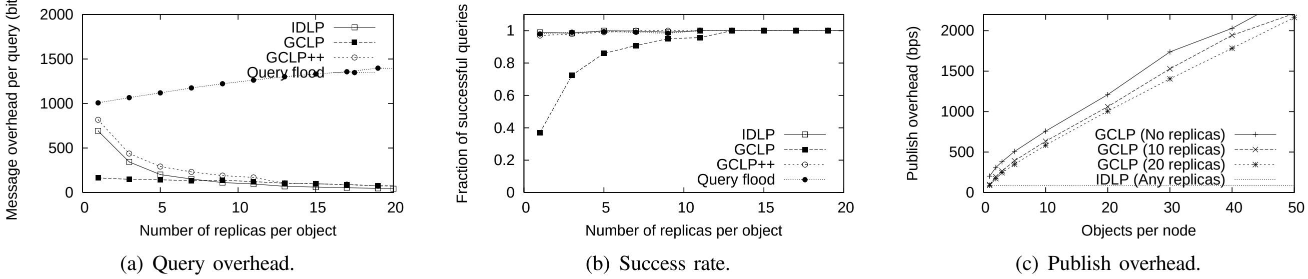


Fig. 8. Comparing IDLP against GCLP and Query flooding.

maximum node speeds. While the random waypoint model is not ideal, it serves our purpose because we are only concerned with the impact of broken links and routes on the performance of IDLP.

Figure 7 shows the broadcast fraction per query with different maximum node speeds. We observe that mobility degrades performance at lower object replication factors. With more replicas, redundancy from multiple simultaneous unicasts mask the impact of broken routes at certain nodes, ensuring that they do not negatively impact the overall unicast performance.

Since IDLP leverages routing control messages for data signature propagation, the publish overhead increases proportionally with control messages and has been excluded from discussion here. We found the overall success rate in the presence of mobility to be close to 100%.

D. Comparing IDLP to GCLP and Query Flooding

We now compare IDLP with the Geography-based Content Location Protocol (GCLP) [27] described in Section II and a simple flood technique. GCLP is a push-based technique proposed for multihop networks, where data records and queries are forwarded across the network in each of the four compass directions. Nodes at the intersection of the publish and query paths use cached data to resolve queries. Because IDLP and GCLP share the same objectives, we compare their performance. Note that GCLP only returns the location of the object while IDLP forwards messages to the object's location. To compare the overhead fairly between the protocols, we assume that GCLP uses an idealized routing protocol, and hence only incurs an additional message overhead to traverse the shortest path between the client node and the object location. Since GCLP does not ensure a query success rate close to 100%, we extend GCLP to resort to a broadcast search upon failure to locate an object. This is represented as GCLP++ in our performance comparison results.

Figure 8(a) shows a comparison of per node message overhead per query between IDLP, GCLP, GCLP++ and Query flood approaches. Query flood invokes a network-wide flood for each object query and hence is likely to generate higher overhead. It is interesting to observe that the performance of Query flood degrades as the number of object replicas increases. This is due to a surge in the reply overhead. GCLP shows a low query overhead even with few object replicas.

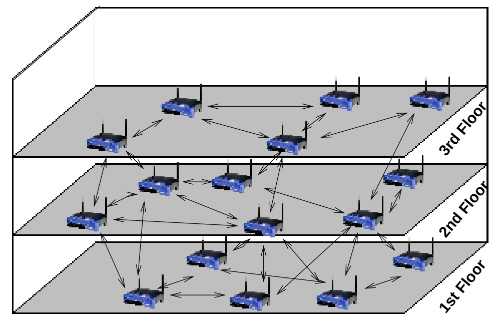


Fig. 9. UCSB MeshNet Testbed.

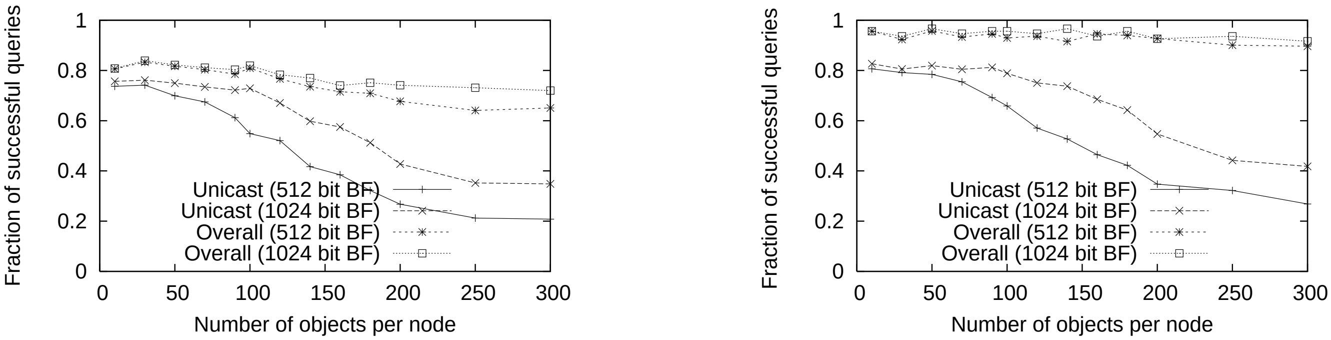
However, Figure 8(b) shows that GCLP has a relatively low success rate when there are few object replicas in the network. GCLP++, IDLP and Query flood all achieve a success rate of close to 100%. IDLP achieves query success rate comparable to GCLP++ without periodically publishing object availability data across the network.

Finally, Figure 8(c) compares object publication overhead for GCLP and IDLP on a per-node basis. IDLP's overhead is independent of the number of objects per node. When the number of objects per node is 50, IDLP provides a factor of 25 improvement in publish overhead over GCLP. The object publish overhead with GCLP increases linearly with the number of objects. GCLP thus works best when the number of objects per node is small. IDLP can support more objects per node with an appropriate selection of Bloom filter length. It is thus suitable for both service discovery and data location with a greater number of objects per node in the network.

V. Evaluation on the UCSB MeshNet

To understand IDLP performance in a real network, we implement IDLP and deploy it over the UCSB MeshNet Testbed[4]. The testbed consists of 25 nodes distributed over five floors of the Engineering I building on the UC Santa Barbara campus. Figure 9 shows an approximate representation of the physical layout of the 16 nodes used for evaluating IDLP. There are two types of nodes in the network: Linksys WRT54G routers and small form-factor Intel Celeron-based X86 boxes running Linux. Each WRT54G node consists of two Linksys WRT54G wireless devices connected together.

[4]http://moment.cs.ucsb.edu/meshnet



(a) Without retries.

(b) With retries.

Fig. 10. Unicast success rate and overall success rate of queries with number of objects in the network.

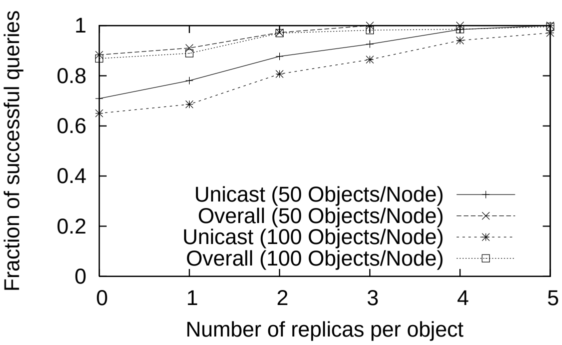


Fig. 11. Unicast success rate and overall success rate with object replication.

One device serves as a mesh node running AODV, while the other is used for out-of-band management of the node. Each X86 box is equipped with a PCMCIA 802.11b radio and a wired Ethernet interface. The IEEE 802.11b radio is used for the AODV mesh and the Ethernet interface is used for out-of-band management.

The link quality between each pair of connected nodes was measured using the ETX [32] metric. The link delivery ratio within the network varies from 10% to 95%. The diameter of the testbed is six hops.

A. Deployment Challenges

We first evaluate IDLP over Kernel AODV [33] deployed on the MeshNet testbed. The initial results obtained were not promising. The unicast success rate of IDLP with 10 objects per node and 512 bit Bloom filter was 43% and the overall success rate was 52%. Detailed investigation showed that the data delivery rate averaged across all node pairs was $\approx$ 50%. This poor performance had two primary causes: the presence of unidirectional links in the testbed and the choice of shortest hop paths by AODV. While previous research has shown the existence of unidirectional links in real testbeds [34], the presence of different node types exacerbates this problem in our testbed. 70% of all unidirectional links were links between Linksys WRT54G routers and X86 boxes[5]. The Kernel AODV implementation does not incorporate the *black list* feature [29] for handling unidirectional links. Thus unidirectional links had a serious impact on the performance of IDLP. This problem manifests during data signature publication in one direction, because the publishing node may be unreachable in the other direction. The other cause of poor performance was the use of hop count as the routing metric by AODV. Researchers have shown the shortest hop count metric to perform poorly for data delivery [35]. This adversely affected the query success rate with IDLP.

Once our investigation revealed these performance factors, we decided to use an ETX-based enhancement proposed for AODV called AODV-ST [36]. AODV-ST maintains bidirectional link connectivity information at each node. Route selection relies on this information and the most reliable path is chosen for data delivery. AODV-ST thus addresses both our issues of unidirectional links and poor path reliability and permits us to accurately evaluate IDLP.

B. Evaluation of IDLP

We now evaluate IDLP over AODV-ST in the MeshNet testbed. We first examine the impact of the number of objects in the network on the overhead and the query success rate. The system parameters chosen are as follows: $\eta = 2$, replication factor is one, and 50% of the nodes have a flow to another random node in the network.

We first look at the success rate of IDLP. Figure 10(a) shows the unicast success rate and the overall success rate obtained with 512 bit and 1024 bit Bloom filters. The unicast success rate represents the fraction of the total queries that were successfully resolved through unicasts alone. The unicast success rate with both 512 bit and 1024 bit Bloom filters in the region of low false positives ($\approx$ 0%) is close to 75%. The overall success rate is about 80%. In contrast, simulation results show a unicast success rate of $\approx$ 80% and a overall success rate of $\approx$ 100% in the same region. This disparity is due to simplistic assumptions in the simulator that do not incorporate the lossy variations of the wireless medium.

[5]The transmission power on these devices was set at the same level.

We next add redundancy in the system to isolate the effect of packet loss. Upon failure to locate an object during either the unicast or the broadcast phase of IDLP, we retry the query. The results from these experiments are shown in Figure 10(b). The unicast success rate increases from 75% to 80% and the overall success rate increases from 80% to 95%. The overall success rate remains around 95%, indicating that multiple retries are required to completely isolate the broadcast loss. It is clear redundancy is needed to cope with the packet loss in the wireless medium.

Figure 11 shows the impact of replication on the success rate of the queries. Higher replication adds redundancy in the system and also increases the chance of a routing table match for the data signature. The system achieves close to 100% success with a replication factor of two. The impact of other system parameters were found to follow the trends observed in simulation and have been excluded due to space constraints.

Finally, we evaluated the end-to-end delay per query and found that it is always less than 500 milliseconds on the MeshNet. This is well within tolerable limits for a data location system.

VI. Conclusion

Adoption of location-aware network services such as disaster recovery requires an efficient data location mechanism for multihop wireless networks. We propose IDLP, a lightweight, bandwidth-friendly data location layer for multihop wireless networks. Through integration of data location and wireless routing, IDLP reduces control protocol overhead and efficiently handles topology changes. The usage of Bloom filters as compact data signatures enables IDLP to support data location for a large number of objects. Simulation results show that piggybacking signatures on routing control messages optimizes control protocol overhead and retains a high search accuracy. Measurement results of IDLP on a testbed show that IDLP provides a high query success rate while incurring low communication overhead and end-to-end delay.

Acknowledgements

This work was supported in part by NSF Career award CNS-0347886 and NSF Career award CNS-0546216.

References

[1] T. A. Howes, "The Lightweight Directory Access Protocol: X.500 Lite," U. of Michigan, Tech. Rep. 95-8, Jul 1995.
[2] Sun Microsystems, "Jini (TM) Architecture Specification v2.0," Jun 2003.
[3] E. Guttman, C. Perkins, J. Veizades, and M. Day, "Service Location Protocol, Version 2," *IETF RFC 2608*, Jun 1999.
[4] P. V. Mockapetris and K. Dunlap, "Development of the Domain Name System," in *Proc. of SIGCOMM*, Stanford, CA, Aug 1988.
[5] M. V. Steen, F. J. Hauck, P. Homburg, and A. S. Tanenbaum, "Locating Objects in Wide-Area Systems," *IEEE Communications Magazine*, pp. 104–109, Jan 1998.
[6] S. E. Czerwinski, B. Y. Zhao, T. D. Hodes, A. D. Joseph, and R. H. Katz, "An Architecture for a Secure Service Discovery Service," in *Proc. of MobiCom*, Seattle, WA, Aug 1999.
[7] B. H. Bloom, "Space/Time Trade-offs in Hash Coding with Allowable Errors," *Journal of the ACM*, vol. 13, no. 7, pp. 422–426, Jul 1970.
[8] L. Fan, P. Cao, J. Almeida, and A. Z. Broder, "Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol," *IEEE/ACM Transactions in Networking*, vol. 8, no. 3, pp. 281–293, Jun 2000.
[9] S. C. Rhea and J. Kubiatowicz, "Probabilistic Location and Routing," in *Proc. of INFOCOM*, Hilton, NY, Jun 2002.
[10] D. Chakraborty, A. Joshi, T. Finin, and Y. Yesha, "GSD: A Novel Group-based Service Discovery Protocol for MANETs," in *Proc. of MWCN*, Stockholm, Sweden, Sep 2002.
[11] S. Helal, N. Desai, V. Verma, and C. Lee, "Konark - A Service Discovery and Delivery Protocol for Ad-hoc Networks," in *Proc. of WCNC*, New Orleans, LA, Mar 2003.
[12] M. Nidd, "Service Discovery in DEAPspace," *IEEE Personal Communications*, vol. 8, no. 4, pp. 39–45, Aug 2001.
[13] O. Ratsimor, D. Chakraborty, A. Joshi, and T. Finin, "Allia: Alliance-based Service Discovery for Ad-Hoc Environments," in *Proc. of ACM Mobile Commerce Workshop*, Atlanta, GA, Sep 2002.
[14] U. C. Kozat and L. Tassiulas, "Network Layer Support for Service Discovery in Mobile Ad Hoc Networks," in *Proc. of INFOCOM*, San Francisco, CA, Mar 2003.
[15] H. Pucha, S. M. Das, and Y. C. Hu, "Ekta: An Efficient DHT Substrate for Distributed Applications in Mobile Ad Hoc Networks," in *Proc. of WMCSA*, English Lake District, UK, Dec 2004.
[16] F. Sailhan and V. Issarny, "Scalable Service Discovery in MANET," in *Proc. of IEEE PerCom*, Kauai Island, HI, Mar 2005.
[17] L. Cheng, "Service Advertisement and Discovery in Mobile Ad Hoc Networks," in *Proc. of Workshop on Ad Hoc Communications and Collaboration in Ubiquitous Computing Environments*, New Orleans, LA, Nov 2002.
[18] R. Koodli and C. E. Perkins, "Service Discovery in On-Demand Ad Hoc Networks," *IETF Internet Draft (expired)*, 2002.
[19] A. Varshavsky, B. Reid, and E. de Lara, "A Cross-Layer Approach to Service Discovery and Selection in MANETs," in *Proc. of MASS*, Washington, DC, Nov 2005.
[20] C. N. Ververidis and G. C. Polyzos, "Routing Layer Support for Service Discovery in Mobile Ad Hoc Networks," in *Proc. of IEEE PerCom*, Kauai Island, HI, Mar 2005.
[21] "KaZaA Media Desktop," http://www.kazaa.com, 2005.
[22] A. Rowstron and P. Druschel, "Pastry: Scalable, Distributed Object Location and Routing for Large-scale Peer-to-peer Systems," in *Proc. of Middleware*, Heidelberg, Germany, Nov 2001.
[23] D. B. Johnson and D. A. Maltz, "Dynamic Source Routing in Ad Hoc Wireless Networks," *Mobile Computing*, vol. 353, pp. 153–181, 1996.
[24] Z. J. Haas, "A New Routing Protocol for the Reconfigurable Wireless Networks," in *Proc. of the IEEE ICUPC*, San Diego, CA, Oct 1997.
[25] S. Lee, W. Su, and M. Gerla, "On-Demand Multicast Routing Protocol in Multihop Wireless Mobile Networks," *Mobile Networks and Applications*, vol. 7, no. 6, pp. 441–453, Dec 2002.
[26] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva, "A performance comparison of multi-hop wireless ad hoc network routing protocols," in *Proc. of MobiCom*, Dallas, TX, Oct 1998.
[27] J. B. Tchakarov and N. H. Vaidya, "Efficient Content Location in Wireless Ad Hoc Networks," in *Proc. of MDM*, Berkeley, CA, Jan 2004.
[28] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks," in *Proc. of MobiCom*, Boston, MA, Aug 2000.
[29] C. Perkins, E. M. Belding-Royer, and S. Das, "Ad Hoc On-Demand Distance Vector (AODV) Routing," *IETF RFC 3561*, Jul 2003.
[30] B. Y. Zhao, L. Huang, S. C. Rhea, J. Stribling, A. D. Joseph, and J. D. Kubiatowicz, "Tapestry: A Global-scale Overlay for Rapid Service Deployment," *IEEE JSAC*, vol. 22, no. 1, pp. 41–53, Jan. 2004.
[31] Scalable Network Technologies. (2005) Qualnet Network Simulator, version 3.8. http://www.scalable-networks.com.
[32] D. S. J. De Couto, D. Aguayo, J. Bicket, and R. Morris, "A High-Throughput Path Metric for Multi-hop Wireless Routing," in *Proc. of MobiCom*, San Diego, CA, Oct 2003.
[33] L. Klein-Berndt, "NIST Kernel AODV Implementation," http://w3.antd.nist.gov/wctg/aodv_kernel/, 2005.
[34] Y.-J. Kim, R. Govindan, B. Karp, and S. Shenker, "Geographic Routing Made Practical," in *Proc. of NSDI*, Boston, MA, May 2005.
[35] R. Draves, J. Padhye, and B. Zill, "Routing in Multi-Radio, Multi-Hop Wireless Mesh Networks," in *Proc. of MobiCom*, Philadelphia, PA, Oct 2004.
[36] K. Ramachandran, M. Buddhikot, G. Chandranmenon, S. Miller, E. Belding-Royer, and K. Almeroth, "On the Design and Implementation of Infrastructure Mesh Networks," in *Proc. of WiMesh*, Santa Clara, CA, Sep 2005.