Cross-layer Routing for Peer Database Querying over Mobile Ad Hoc Networks

Evangelos Papapetrou*, Panos Vassiliadis, Efthymia Rova, Apostolos Zarras

Department of Computer Science, University of Ioannina, 45110, Ioannina, Greece

Abstract

The widespread of mobile ad-hoc networking calls for a careful design of network functions in order to meet the application requirements and economize on the limited resources. In this paper we address the problem of distributing query messages among peers in mobile ad hoc networks. We assume that peers are organized in classes. Each peer possesses a local database and can answer queries posed by other peers. Each peer can also pose queries to all the peers belonging to a certain class or classes. Contrary to traditional p2p lookup queries, we are interested in collecting answers from as many peers as possible. We propose a query routing protocol, called CL-QF, which is based on a novel cross-layer design. The purpose of this design is to incorporate application layer specifics (e.g., class information) into the network layer in order to reduce transmissions therefore economize on resources. CL-QF coexists with traditional routing. This synergy minimizes the complexity and signaling of CL-QF while the network is able to seamlessly provide legacy unicast communication. CL-QF manages a reduction of up to ∼78% compared to non cross-layer approaches, such as probabilistic forwarding, without compromising the ability to effectively collect replies.

Keywords: mobile communication systems, routing protocols, wireless communication, distributed applications, databases

1. Introduction

The scientific community has witnessed a significant rise of *data-centric networking applications* over the last years. This type of applications introduces a new communication paradigm. Instead of communicating one-to-one based on their identity, users generally communicate in groups depending on the data they possess and need to exchange at the time. Take for example p2p file sharing applications [1], where users communicate spontaneously depending on

*Corresponding author

Email addresses: epap@cs.uoi.gr (Evangelos Papapetrou), pvassil@cs.uoi.gr (Panos Vassiliadis), efirova@cs.uoi.gr (Efthymia Rova), zarras@cs.uoi.gr (Apostolos Zarras)

their content. Another example is the concept of peer databases [2], where each peer carries a database and can pose queries to the databases of other peers. Furthermore, the advent of mobile devices has created an increased interest for data-centric applications in *wireless mobile environments* such as DHTs over mobile ad hoc networks [3] and data management systems for querying data collected over large communities of sensor-enabled cars [4].

In this paper, we are concerned with the design of networking protocols for supporting the operation of a *peer database application* over a mobile ad-hoc network. Various applications of this type have been proposed [4], [5], [6] in the context of mobile computing. More specifically, we are interested in situations where mobile peers are organized in groups, called classes, based on the data they carry (in the form of relational databases) and use to answer queries. The queries posed by a peer are directed to members of specific groups and aim to collect as many answers as possible (rather than highlighting a single peer that can answer). The communication scenario of peer databases requires query packets to be forwarded to all network nodes able to respond to the query, leading to high signalling overhead. In the context of a MANET, such a requirement calls for the design of customized networking procedures in order to alleviate the involved overhead. As it will be explained in the following, traditional routing protocols fail to efficiently support such an application. On the other hand, although significant work has been performed on data-centric applications over MANETs, it cannot be applied to the case of peer databases. The reason is that the design of data-centric networking protocols is strongly related to the characteristics of the application to which the network has to provide its services. For example, in the case of a p2p file sharing application, a protocol that identifies a single node having a copy of the requested file may be sufficient. On the other hand, this is not suitable for peer databases since the network needs to determine as many as possible peers able to respond to the posed query.

In this paper we propose a lightweight mechanism, called *cross-layer query forwarding (CL-QF)*, for forwarding the queries posed by each peer in the mobile ad hoc network. The goal of CL-QF is to (a) maximize query delivery to the peers that should answer a query, (b) avoid flooding the network with packets and minimize the signaling overhead, and (c) maximize the number of responses collected by peers. To this end, CL-QF introduces the following innovative approaches:

- *cross-layer design*: the design incorporates application layer specifics (e.g. class information) into the network layer so that it can be exploited for minimizing the number of nodes that need to transmit in order for a query to reach all the eligible nodes

- *synergy with traditional routing protocols*: unlike other data-centric protocols, CL-QF builds on top of traditional routing protocols. The advantage of this approach is twofold; on one hand the complexity and signaling of CL-QF are minimized by exploiting existing routing information while on the other hand the network is able to seamlessly provide traditional as well

as data-centric communication services without excessively increasing the overall signaling cost.

The simulation results prove the efficiency of CL-QF compared to non cross-layer approaches. More specifically, CL-QF, without sacrificing the ability to collect queries, accomplishes a massive reduction of transmissions that reaches up to ∼85% and ∼78% when compared to plain flooding and probabilistic forwarding respectively.

The rest of the paper is organized as follows. In Section 2, the problem of query routing and peer database communication in mobile ad hoc networks is described in detail. Then, in Section 3, the proposed scheme is delineated, while in Section 4 an analysis of its performance is presented. Section 5 presents the simulation model used for the assessment of the proposed protocol, while in Section 6 simulation results are presented and discussed. Finally, related work is presented in Section 7 and concluding remarks are drawn in Section 8.

2. Problem Description

2.1. Communication Scenario and Basic Concepts

As explained in the previous section, the motivation for this work stems from the implementation of *peer databases* over a mobile ad hoc network and the communication services required to support such a family of applications. To understand the concept of peer databases over a MANET, let us introduce the following example concerning an emergency wireless network set up in the area where a wildfire blasts. There are various kinds of mobile users roaming in the area: firefighter vehicles, both individual and squads of firefighters, sensors for the measurement of temperature, humidity, and wind deployed in the area in an ad-hoc fashion, unmanned aeroplanes, also equipped with sensors, flying over the area, and, finally, manned firefighting helicopters bombarding the forest with water or fire-resistant chemicals. All these mobile "actors" are equipped with some kind of computational apparatus-like, for example, embedded CPU's, special-purpose PDA's or mini laptops. Each such actor stores data in a lightweight database that operates on his computing facility and serves two purposes: (a) it allows the user to pose queries to the network formed by the other users and (b) it provides partial answers to the queries posed by the other users to the network. In other words, query processing follows a simple mechanism: (i) a user poses a query to the network, (ii) the query is forwarded to the rest of the peers in the network, who, in turns (iii) compute their partial, local answer to the query and send it back to the querying peer, who (iv) collects as many answers as possible until a timeout, a threshold on the number of answers or responding peers, or any combination of the above is reached and then (v) performs query processing over the collected data as usual.

Since a querying peer collects data from several other peers, a typical database integration problem must be resolved for the structure of the involved databases: how does the information stored in the database of a certain peer p_1 match the

information of the database of another peer p_2? The two extremes in the setting involve the possibility (a) for each peer to follow its very own structure and (b) all peers to conform to the same structure exactly. For obvious reasons of scalability and usability, we avoid both extremes and assume a middle path: users are grouped in classes, with each class exporting a specific structure for its members' data to the network. We will abstract the database querying complexities and refer to this common structure as the interface of the class. Two classes may share some properties and differentiate themselves in others: for example, both ground sensors and unmanned planes may have a way to report temperature, but only the unmanned aeroplanes may export also a report on smoke percentage in the atmosphere. As a second example, all flying machinery may export the amount of available fuel they have as part of their interface, but only helicopters may export the amount of load they carry since their last charge or drop.

Overall, a class involves a *class name*, an $interface$ (i.e., the set of querying facilities over relational tables that it exports to the network) and a *population* (i.e., a set of nodes that implement the interface of the class and are part of the network at a given time point). For the purpose of this paper, we will simplify the terminology and use the term class equivalently with the term *class population*, unless otherwise stated.

Definition 1. Class. A set of nodes exporting the same interface to the peer network.

Querying may involve one, several, or all the classes of the network, depending on the type of the desired information. For example, average smoke density around a specific location involves only the members of the class $UnmannedPlane$, whereas a query for the average temperature in a specific area, requires an access to members of different classes. It is worth pointing out that collecting as many as possible replies is essential in the aforementioned examples in order to obtain an estimation close to reality. Collecting as many replies as possible would also be of interest in the case that we wish to obtain a mapping of the temperature or the smoke percentage in the entire area. Furthermore, note that a user (e.g., the chief of operations in our example) should be allowed of limiting the range of a query. To motivate the discussion, assume a peer network of the topology of Fig. 1, consisting of 9 peers, each representing a helicopter, a sensor or an unmanned aeroplane in the vicinity of the current position of the chief's helicopter. Obviously, some of these peers are mobile and some of them are static. Assume now that the chief (peer p_1) needs to find the average smoke concentration in the area near his helicopter. To avoid flooding the network with unnecessary load and achieve this goal, the peer needs to perform two actions (not necessarily in the following order): First, it needs to identify which of the peers are the unmanned planes (that report smoke concentration). Second, it needs to identify which of these peers are close to its current location. Observe a dichotomy between the two kinds of properties: the former property ('being a unmanned plane') is a fixed property that lives with peer from the

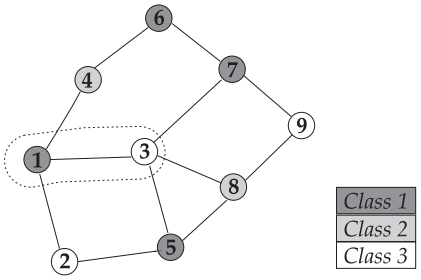


Figure 1: An example network with three classes of nodes

```
SELECT AVG(SMOKE) FROM U_PLANES
WITH HORIZON COMMUNITY Distance_Under_3km AND
RESPONSE_TIME < 4.0 AND CLASS = 'UnmannedPlane'
```

Figure 2: Example query in SQL^P

moment of its birth, whereas the latter has to do with its current state ('close to peer p_1'). As mentioned above, we make the reasonable assumption that all planes abide by the same interface and are thus classified in the same class. At the same time, we allow the peer to define a *community of interest* [7],[8],[9] for peers satisfying a certain -possibly temporal- property like, for example, location. The routing protocol can be used to convey information about these two aspects; nevertheless the paper discusses only the case of classes, for reasons of simplicity.

The peer p_1 utilizes the following constructs:

- Three classes, *Helicopter* (denoted as *Class 1* in Fig. 1), *Sensor* (denoted as *Class 2*), and *UnmannedPlane* (denoted as *Class 3*) that are commonly accepted classes for the environment that we discuss
- Two communities, the community of close peers (*Distance_Under_3km*) and the community of distant peers (*Distance_Over_3km*). Assume that peers p_2, p_3 and p_4 belong to the former category and the rest of the peers to the latter.
- A database, held locally by p_1, holding information about querable properties of peers. Assume that the relation $U\_PLANE(\underline{ID}, X-COORD, Y-COORD, Z-COORD, FUEL-LEFT, SMOKE, HEAT, HUMIDITY)$ is part of this database.

In [10] we have formally introduced SQL^P, an extension of SQL, to allow peers perform this kind of queries. The language allows the user to restrict the range of interest to classes, communities and other properties of peers. For example, in the query of Fig. 2, apart from constraining the class and community of the interesting planes, we also require that the expected response time will be less than 4 seconds. The main idea behind SQL^P is that the required data will be collected at the local database of the requesting peer at runtime (i.e., in our example, the relation U_PLANE will be populated at runtime), and subsequently processed there. The details for the query language can be

found in [10]. In [11] we discuss a lightweight middleware infrastructure for the execution of this kind of operations. In this paper, however, we focus on the exploitation of the routing protocol for the efficient dissemination of these kinds of queries and the collection of answers.

In the described scenario, the network has to provide users with the following services:

- Routing of queries
- Delivery of reply messages

Routing of queries is the most important service and its main objective is to deliver a query to the recipients relevant with the query. The key characteristic of the communication scenario in peer databases is that the network is required to deliver the query to *as many as possible eligible nodes*, so that as much information as possible is collected. At the same time, an efficient implementation is required in order to economize on the limited resources of a MANET. Such an implementation is not straightforward with traditional networking protocols. For example, using a traditional routing protocol would require to unicast the query to each of the network nodes, leading to the waste of valuable bandwidth. Another more elaborate solution would be to use multicast routing protocols [12]. However, the implementation of such a scenario would require the construction of a multicast tree for each different query. Bearing in mind the overhead involved in building and maintaining multicast trees and the plethora of possible queries, the inefficiency of the approach is clear. Finally, another option would be to use a flood-based technique in order to reach all possible query recipients. However, in this case, the incurred signalling overhead is also considered significant since all network nodes are involved in the process even if they are not appropriate query recipients.

Finally, as far as delivery of reply messages is concerned, the network should be capable of routing replies to the originator of the query that produced them. Since this is a classical one-to-one communication, traditional routing protocols could be used. However, the cost of implementing both of the aforementioned services independently may be significant. Therefore, an efficient protocol should strive for enhancing their synergy.

2.2. Problem formulation

A snapshot of the network presented in the previous example, under the assumption that links are bidirectional, can be modeled as an undirected graph $G(V, E)$ comprising of a set of nodes V and a set of edges E. Each edge between two vertices u and v denotes that node u lies in v's transmission range and vice versa. The size of V is the size of the network N ($N = |V|$). A number CL_{cnt} of classes may exist in the network.

As explained previously, a node may pose a query, which according to its structure may be considered to refer to one or more classes. In order for the network layer to provide the appropriate service to a peer database application, it should deliver the query message to all the target nodes of the query. To this

end, some network nodes should forward the query. Throughout the paper, the following terminology will be used frequently:

Definition 2. Target Class of a query. The class or the set of classes that a query refers to.

Definition 3. Target Node of a query. A node belonging to the target class of a query.

Definition 4. Forwarding Node for a query towards class i. A node forwarding a query to the target class i.

Assume a query q. We employ the following notation[1]:

· C_{tg} denotes the set of target nodes of the query q
· F_{tg} denotes the set of forwarding nodes for the target class of the query
· R_{tg} denotes the set of nodes that actually receive the query
· $N(v)$ denotes the set of nodes, including v, lying at most one hop from v

Since each forwarding node broadcasts the query to its neighbors, the number of nodes that receive the query message are:

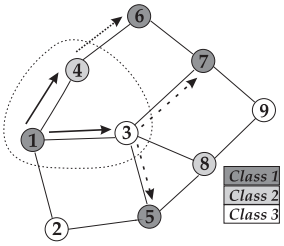
$$R_{tg} = \bigcup_{\forall v \in F_{tg}} N(v)$$

Also, the number of packets that carry the original query message is $|F_{tg}|$. *We wish to minimize the cost of routing the query, i.e., the number of these packets and, at the same time, reach all the members of the class that should receive the query.*
Equivalently, $|F_{tg}|$ should be minimized subject to the condition $C_{tg} \subseteq R_{tg}$, or alternatively

$$C_{tg} \subseteq \bigcup_{\forall v \in F_{tg}} N(v)$$

The above indicate that the information regarding the class of each node is crucial and can be exploited for minimizing the cost of query routing. *The crux of our approach lies in silencing forwarding nodes belonging to F_{tg} if their transmission is redundant in reaching target nodes for the query.* To understand this consider the case in fig. 1 where node 1 wishes to perform a query for discovering data related to class 2. Without any information about C_{tg} all nodes in the network should forward the query in order to ensure that $C_{tg} \subseteq R_{tg}$.

[1]to avoid crowding the notation unnecessarily, we do not annotate the terms with a superscript q that indicates the query to which a parameter refers to



RT:Node 1

Dst	NH	Hop Cnt	Class
1	-	0	1
2	2	1	3
3	3	1	3
4	4	1	2
5	3	2	1
6	4	2	1
7	3	2	1
8	3	2	2
9	3	3	3

RT:Node 3

Dst	NH	Hop Cnt	Class
1	1	1	1
2	1	2	3
3	-	0	3
4	1	2	2
5	5	1	1
6	7	2	1
7	7	1	1
8	8	1	2
9	8	2	3

RT:Node 4

Dst	NH	Hop Cnt	Class
1	1	1	1
2	1	2	3
3	1	2	3
4	-	0	2
5	1	3	1
6	6	1	1
7	6	2	1
8	1	3	2
9	1	4	3

Figure 3: Example of query forwarding

On the contrary, by exploiting class information we can minimize the set of forwarding nodes (indicated by the dashed line). In this way both nodes 4 and 8 that belong to class 2 will receive the query. At the same time, node 2, which is a neighbor of the querying node 1, will not unnecessarily be involved in the forwarding of the query. The same scheme is repeated in the subsequent forwarding of the query.

3. Cross Layer Query Forwarding (CL-QF)

Before delineating the mechanisms of the proposed networking approach, we first describe some of our basic design choices.

3.1. Design Choices

Following the discussion in the previous section it is clear that the basic objectives of this work are:

i) the minimization of the cost involved in routing a query
ii) the synergy with traditional routing protocols

3.1.1. The synergy with traditional routing

As explained in the previous paragraph, providing networking services to data-centric applications requires customized protocols. Since data-centric applications are expected to thrive, the cost of running a specialized protocol per application in the context of a MANET is prohibitively high. In this context, we are not interested in building a special purpose network. On the contrary, we wish to support the coexistence of data-centric applications such as a peer database with other types of applications. Therefore, one of our major design choices has been not to build a new routing protocol rather than propose generally applicable modifications so that the described functionality is incorporated into existing routing protocols. To this end, CL-QF works on top of traditional routing protocols. However, we must make clear at this point that the proposed solution aims at proactive routing protocols [13] due to the nature of the application. Keep in mind that when a node performs a query, data must be collected by as many network nodes as possible. The only way to achieve this without performing a network-wide search is to use a proactive routing protocol. This

is because in proactive routing protocols each node maintains a routing table containing an entry for each discovered node in the network. The entry contains data, such as the next hop, indicating how to reach the respective node. Each node and its neighbors periodically exchange routing updates (distance vector routing), called HELLO messages, containing all or part of their routing table. By this mechanism, in each algorithm repetition, a node populates its routing table with nodes that are one hop further. In an ideal network, the aforementioned procedure would lead to the discovery of all network nodes. However, in mobile ad hoc networks, the actual number of discovered network nodes is limited by mobility, joining and leaving the network and by network partitions. By working on top of a proactive routing protocol, CL-QF is capable of exploiting available routing information in order to minimize the cost of routing a query.

3.1.2. Cross-Layer Design

The discussion in Section 2.2 has highlighted the need for exploiting application layer specifics such as the organization of nodes into classes. To this end, our second design choice has been to opt for cross-layer design. Since CL-QF works on top of any proactive routing protocol, in order to be able to make decisions based on application information at the networking layer, we must devise an efficient mechanism for making this information (the class that a node belongs to) available to the routing protocol. To this end, we introduce the name of the node's class to the routing updates that the node sends in the context of the proactive routing protocol. Furthermore, each node receiving such updates from other nodes should store class information to the entry of its routing table that is related to the node that sent the update. In this way, each node becomes aware of the class that every node in its routing table belongs to. The proposed procedure increases somewhat the signalling overhead due to the larger size of update messages. However, the increase in the size of each update message is rather small (only a couple of bytes). Furthermore, bear in mind that the information regarding the class of a node is in most cases fixed. Therefore its propagation in the network by means of update messages may be limited to only some of the updates. Finally, we should point out that the proposed procedure does not depend on the routing protocol that is implemented.

3.2. Basic Mechanisms

CL-QF consists of three basic mechanisms, namely: i) identification of target nodes, ii) query routing, and iii) reply forwarding.

3.2.1. Identification of target nodes

The cornerstone for implementing an efficient query routing mechanism is to make forwarding decisions for query messages based on the application layer information, already incorporated into the routing tables as described previously. In our approach, when a node submits a query, in order to avoid flooding it, the first step is to identify all the nodes that are eligible for receiving it. This is done by searching the local routing table for nodes belonging to the target

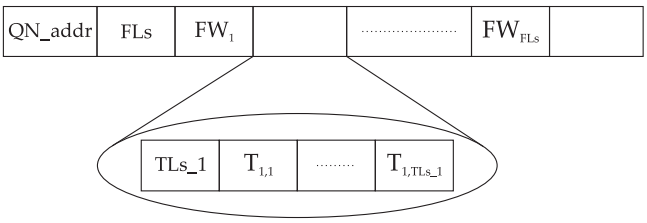


Figure 4: Structure of query header

class. To understand the procedure, consider the example depicted in fig. 3. Node 1, based on its view of the network, issues a query towards nodes of class 1. By consulting its routing table (RT:$Node1$), node 1 can determine the target nodes, i.e., the nodes that the query must reach. In our example, the target nodes are nodes 5,6 and 7. The rationale behind this approach is straightforward: we should not be concerned with other than target nodes. Therefore, only forwarding nodes for the target class should be elected.

3.2.2. Query Routing

After identifying the target nodes, the originator of the query should determine the forwarding nodes for the target class. Since each node has partial knowledge of the paths leading to nodes in the target class, it cannot determine the forwarding nodes for the entire path. Instead, it determines the forwarding nodes in the first hop of communication and relies on those forwarding nodes to repeat the procedure until all forwarding nodes are determined. Clearly, the forwarding nodes in the first hop of communication are the next hops for reaching the target nodes. This information can be retrieved from the routing table. Coming back to the example of fig. 3, nodes 3 and 4 are identified by node 1 as forwarding nodes since they are the next hops for reaching nodes 5,7 and 6 respectively.

In order to minimize the number of query packets, a node, based on its routing table, groups the target nodes according to the next hop used to reach them. The rationale behind this approach is to construct only one message per next hop and not per target node. This grouping of nodes has been proposed for use in the Internet [14]. However, we expand this approach to take advantage also of the wireless transmission properties. To this end, each forwarding node needs to transmit only one message since all next hop nodes are able to receive the message. It is clear however that the message should contain information regarding all forwarding nodes. Therefore, the node constructs a header of the structure depicted in fig. 4.

The fields $FW_1, FW_2, \ldots$ are used to store the forwarding nodes identified by the current node. Each forwarding node FW_i is followed by a set of target nodes $\{T_{i,1}, \ldots, T_{i,TLs\_i}\}$ that can be reached by FW_i. At this point we should note that a forwarding node may be a target node as well. However, as it will be made clear later, its address does not need to be included in this set. For easiness in accessing the data in the header, the list of target nodes for each FW_i

is also accompanied by its size (TLs_i). Furthermore, the number of forwarding nodes contained in the header is stored in the field FLs. Finally, the address of the originator of the query is stored in the field QN_Addr. This information is necessary in order for the target nodes to reply to the query. In the example of fig. 3, node 1 will construct a header containing 3 and 4 as forwarding nodes while nodes 5 and 7 are grouped under node 3 and node 6 under node 4.

After its construction the header is appended to the query message and the packet is transmitted using layer-2 broadcast. Each node that receives the packet forwards it to its application layer. Furthermore, it checks if its address appears in the list of forwarding nodes. If this is not the case, the node drops the packet. However, if the node is a forwarding node then it constructs a new header by following the procedure described before and using as target nodes only the nodes included in its target list. To make this clear, let us consider again the example of fig. 3. In this example, node 3 is a forwarding node. When receiving the query from node 1, it constructs a new header following the same procedure as node 1 and using as targets only the nodes 5 and 7 (node 6 is the responsibility of node 4). The described procedure ends when a node that receives the query discovers that its list of target nodes is empty. At this point, all target nodes have received the query.

The result of the proposed procedure is to minimize the number of transmissions within the network since the retransmission of query packets is strictly controlled. Consider again the example of fig. 3. The total number of transmissions used to deliver the query to all recipients is three (each transmission is depicted with a different line style). In the case that flooding is used, the total number of transmissions for the same example is nine. The disadvantage of the method is that it relies on the accuracy of the routing protocol used to deliver queries. However, this is a tradeoff that we are willing to pay in order to save resources and to achieve the compatibility with traditional routing protocols.

3.2.3. Reply Forwarding

Whenever a target node receives a query, it computes the answer and forwards it back to the originator of the query through traditional routing mechanisms. This can be done by using the address stored in the QN_Addr field of the query header and the underlying routing protocol. The rationale behind this choice has been to enhance the protocol's flexibility, since, in this way, CL-QF does not produce additional overhead for setting up the reverse path. Furthermore, the choice of utilizing a proactive routing protocol minimizes the delay in collecting replies since reverse paths are available and do not need to be discovered at the time that the query is executed.

The querying node collects answers until either a timeout occurs, or a certain amount of answers arrives, or a certain number of peers respond, or any combination of the above. The stopping condition can be part of the query; see [10], for the SQL extensions that deal with these issues. Every answer that arrives at the querying node is locally cached; once the stopping condition is activated, the querying node performs the appropriate query processing to the collected answers. The details of this query processing are explained in [10]; furthermore,

any optimizations of this task with the simultaneous collection and processing of answers, or the delegation of parts of the query processing to other nodes is well beyond the scope of this paper.

3.3. Processing and power requirements

As mentioned before, the objective of CL-QF is to deliver a query to as many as possible target nodes and at the same time minimize the number of transmissions. To accomplish this, CL-QF requires increased processing for reading and constructing the header of a query. In other words CL-QF takes the approach to trade bandwidth with CPU. This approach is suitable for the context of a MANET since the limited resource is clearly bandwidth. Although the nominal bandwidth has increased up to 54 Mbps or more, the effective bandwidth is limited by contention-based access protocols. On the contrary, powerful and power efficient mobile processors already exist [15]. Furthermore, it should be pointed out that the reduction of transmissions also alleviates processing since the processor is not required to transfer a packet to/from the wireless interface.

As far as the power consumption is concerned, researchers have provided evidence [16],[17] proving that the energy consumption in a wireless interface is, in the best case, in the order of tens of μW not only for transmitting but also for receiving or even discarding packets. On the other hand, state of the art mobile processors can provide up to $8 \cdot 10^8$ processing cycles with just 0.5 W, that is 0.625 nW per processing cycle [15].

4. Query Routing Analysis

In order to evaluate the impact of the cross-layer design of CL-QF we analyze the cost involved in query routing compared to the case that class information is not taken into account. In the following, we will refer to the latter case as *simple query forwarding (SQF)*. In other words, SQF is similar to CL-QF except that the class information is not available. Therefore, in SQF, all network nodes are considered as target nodes. Note that SQF only serves as a reference protocol that will help us to clearly outline the benefits related to the proposed cross-layer design.

The execution of the routing mechanism for a query q in each of the forwarding nodes in the network can be considered, on an abstraction level, to form a tree rooted at the originator of the network. Fig. 5 presents the formed tree for the example of query routing illustrated in fig. 3. In general when SQF is used, the size of the tree is N and its depth H, which represents the maximum distance (in hops) of a node from the originator of the query. All internal nodes of the tree are query forwarding nodes and vice versa. In the case of CL-QF, some parts of the tree are truncated since the set of forwarding nodes is minimized. This is indicated in the example of fig. 5 (dashed line) where only nodes 3 and 4 need to forward the query in order to reach all the nodes belonging to class 1.

We refer to the formed tree as T and denote by N_i the set of its nodes with depth i. In other words, N_i is the set of nodes that, according to the underlying

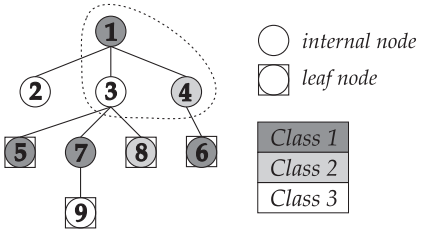


Figure 5: Example tree formed during query forwarding

routing protocol, reside i hops away from the originator of the query[2]. Similarly, F_i represents the set of internal nodes with depth i, while L_i denotes the set of leaf nodes. Clearly, $|F_i| + |L_i| = |N_i|$. As mentioned before, for the case of SQF, all internal nodes are forwarding nodes and vice-versa, therefore the average number of transmitted packets is:

$$D^{SQF} = \sum_{i=0}^{H-1} \sum_{\forall v \in F_i} 1 = \sum_{i=0}^{H-1} |F_i| \tag{1}$$

Clearly $D^{SQF} \leq \sum_{i=0}^{H} |N_i| = N$, with the latter being the number of packets when plain flooding is used (since all nodes transmit the query message). As far as CL-QF is concerned, the forwarding nodes are a subset of $\bigcup_{i=0}^{H-1} F_i$. Another way to express this is to say there is a probability that a node $v \in F_i$ will be a forwarder, i.e. it will receive and finally forward the query. In the following we employ the notation:

- FW^v denotes the event that a node $v \in F_i$ is a forwarder, i.e. it both receives and forwards the query
- $upstr(v, j)$ denotes the upstream node of v with depth j in the path from the originator of the query (root of the tree) to node v; clearly $0 \leq j < i$
- R^v is the event that a node v receives the query from its upstream node $upstr(v, i-1)$
- Tg^v is the event indicating that there is at least one target node in the subtree T_i^v rooted at node v, i.e., Tg^v = at least one target node in T_i^v

Clearly, the average number of transmitted packets for the case of CL-QF is:

$$D^{CL} = \sum_{i=0}^{H-1} \sum_{\forall v \in F_i} P\{FW^v\} \tag{2}$$

The probability of node v being a forwarder equals the probability that v receives the query and transmits it because there is at least one target node in the subtree

[2]again, we do not annotate the terms with a superscript q that indicates the query to which a parameter refers to, to avoid crowding the notation unnecessarily.

T_i^v rooted at node v, i.e.:

$$FW^v = R^v \cap Tg^v \tag{3}$$

Furthermore, in order for a node v to receive a message, its upstream node $upstr(v, i-1)$ should be a forwarder, i.e.:

$$R^v = FW^{upstr(v,i-1)} \tag{4}$$

To continue with the calculation of D^{CL} we introduce the following theorem:

Theorem 1. *The probability that a node $v \in F_i$ is a forwarder is equal to the probability that there is at least one target node in the subtree T_i^v. In other words, $P\{FW^v\} = P\{Tg^v\}$.*

Proof. By combining equations 3 and 4 we can show that:

$$\begin{aligned} P\{FW^v\} =& P\{FW^{upstr(v,i-1)} \cap Tg^v\} \\ =& P\{R^{upstr(v,i-1)} \cap Tg^{upstr(v,i-1)} \cap Tg^v\} \end{aligned} \tag{5}$$

However, it is clear that:

$$Tg^{upstr(v,i-1)} \cap Tg^v = Tg^v \tag{6}$$

because $T_i^v \subset T_{i-1}^{upstr(v,i-1)}$, therefore if a target node exists in T_i^v then it also exists in $T_{i-1}^{upstr(v,i-1)}$. As a result Eq. 5 results in:

$$P\{FW^v\} = P\{R^{upstr(v,i-1)} \cap Tg^v\} \tag{7}$$

and since node $upstr(v, i-1)$ receives the query only if node $upstr(v, i-2)$ is a forwarder, then:

$$P\{FW^v\} = P\{FW^{upstr(v,i-2)} \cap Tg^v\} \tag{8}$$

The comparison of Eqs. 5 and 8 indicate that by repeating the same procedure recursively we can show that:

$$P\{FW^v\} = P\{FW^{upstr(v,0)} \cap Tg^v\} \tag{9}$$

Since node $upstr(v, 0)$ is the root of the query, by definition $P\{FW^{upstr(v,0)}\} = 1$. Therefore, $P\{FW^v\} = P\{Tg^v\}$.

According to theorem 1 it is sufficient to calculate $P\{Tg^v\}$ in order to evaluate D^{CL}. To this end, let us denote by p_c the probability that a node is a target node, i.e., belongs to the target class of the query. Since nodes are independent, the probability $P_{tg}(k)$ of having k target nodes out of M nodes is given by the binomial distribution:

$$P_{tg}(k) = \binom{M}{k} p_c^k (1-p_c)^{M-k}$$

while the probability of having at least one target node is:

$$P_{tg}(k \geq 1) = 1 - P_{tg}(k = 0) = 1 - (1 - p_c)^M \tag{10}$$

Therefore, the probability that there is at least one target node in the subtree T_i^v rooted at node v is:

$$P\{Tg^v\} = 1 - (1 - p_c)^{|T_i^v|-1} \tag{11}$$

Finally, theorem 1 allows us to combine the latter equation with Eq. 2:

$$\begin{aligned} D^{CL} &= \sum_{i=0}^{H-1} \sum_{\forall v \in F_i} (1 - (1 - p_c)^{|T_i^v|-1}) \\ &= D^{SQF} - \sum_{i=0}^{H-1} \sum_{\forall v \in F_i} (1 - p_c)^{|T_i^v|-1} \end{aligned} \tag{12}$$

In this equation the size of any tree $T_i^v,\ \forall v \in F_i$ is constrained by the total network size. Indeed, the summation of the size of those trees, excluding the root node, equals the number of nodes with greater depth in the tree:

$$\sum_{\forall v \in F_i} (|T_i^v| - 1) = \sum_{j=i+1}^{H} |N_i| = N - \sum_{j=0}^{i} |N_i| \tag{13}$$

Eq. 12 and 13 quantify the improvement of CL-QF compared to SQF and are useful for understanding the gains of the cross-layer approach. In a network of fixed size, CL-QF is favored when $p_c \to 0$ since $D^{CL} \xrightarrow{p_c \to 0} 0$. That is, the gains of CL-QF are greater when the population of the target class is small – or, equivalently, the number of the classes in the network is high. On the contrary, when $p_c \to 1$, i.e., there is only one class in the network, $D^{CL} \xrightarrow{p_c \to 1} D^{SQF}$. This is reasonable since the existence of one class actually provides no information that CL-QF can take advantage of. Another useful result is that CL-QF is favored in networks that $|F_i|$ is maximized compared to $|L_i|$. This is because forwarding nodes represent occasions where a transmission may be canceled. An easy way to understand this is to consider the two extremes; a tree where each node has only one child and a tree where all nodes are children of the root. CL-QF is also favored as the network size increases if the ratio $|F_i|/|L_i|$ is high. Again, the reason for this property of CL-QF is the fact that the opportunities for canceling transmissions are maximized. All the aforementioned qualitative results are fully confirmed by the simulation study presented in Section 6.

Providing analytical results by using Eq. 12 and 13 is not possible unless information regarding the tree organization is known. Although there exist some works [18],[19] discussing properties of a MANET topology such as node density and degree, connectivity, etc, they do not address the properties of a minimum distance tree rooted at a given node. Therefore, Eq. 13 can be used

to produce limits on the size of T_i^v:

$$1 \leq |T_i^v| - 1 \leq N - \sum_{j=0}^{i} |N_i| - |F_i| \tag{14}$$

According to this equation the lower limit of the size of the tree rooted at v is 2 (node v itself and a child node since v is a forwarding node). The upper limit results in the case that $|T_i^u| = 2,\ \forall u \in F_i,\ u \neq v$, i.e., all other forwarding nodes with the same depth are the roots of minimum sized trees. In this case, the nodes that may belong in T_i^v are those with greater depth ($N - \sum_{j=0}^{i} |N_i|$) after excluding one child per forwarding node ($|F_i|$ in total). Eq. 14 can be combined with Eq. 12 to produce limits on the performance of CL-QF.

5. Simulation Framework

5.1. Methodology

To assess the performance of cross-layer query forwarding (CL-QF) we compare it, through extensive simulations, to the simple query forwarding scheme (SQF). Again, we should point out that SQF is a downgraded version of CL-QF in the sense that it does not take into account class information. Its purpose is to help us highlight the performance gains related to the cross-layer design as opposed to the gains related to the forwarding technique. Furthermore, we compare CL-QF with Plain flooding (PF) and with a probabilistic flooding scheme in which each node forwards a message with probability p. Probabilistic schemes are known to present a bimodal behavior [20]. That is, given a suitable p, a flooded message will reach all nodes in the network. In this work, we have implemented the scheme described in [20] that has been proposed for wireless ad hoc networks. According to this scheme a message is forwarded with probability 1 in the first k hops (excluding the first one) and after that each node forwards the message with probability p. Clearly the performance of this scheme depends on the choice of k and p. In the following experiments, we present the results for the probabilistic scheme with parameters k and p set to the values that yield the best performance in terms of the scheme's ability to deliver the query. More specifically, we have chosen the values that provide the same delivery efficiency as the other schemes and minimize at the same time the number of nodes that need to forward a message. The optimal values for k and p where determined through exhaustive parameter testing. More specifically, we tested $k \in [0, 3]$ and $p \in [0.05, 0.7]$ with a step of 0.05. The range of the tested values was adequate for capturing the bimodal behavior of the scheme. In other words, the optimal values for k and p were well inside the aforementioned ranges. Using any value greater than the optimal ones resulted in more transmissions without improving the delivery ratio. In the rest of the paper we will refer to the probabilistic scheme as *PROB*.

The presented results have been obtained by using the CMU extensions of ns2 [21] for MANETs, after performing appropriate modifications in order to

implement CL-QF, SQF as well as PROB. All aforementioned techniques have been built on top of DSDV [22], which is one of the most well-known proactive routing protocols, proposed for MANETs. The existing implementation of DSDV in ns2 has been used. At this point, it must be noted that the implementation of CL-QF does not rely and is not limited by the underlying routing protocol. CL-QF can smoothly operate with any proactive routing protocol that uses routing tables. To illustrate this, in some experiments, we use an implementation of SQF and CL-QF over the well-known OLSR protocol [23]. To this end, the UM-OLSR implementation [24] for ns2 has been used.

The evaluation of algorithms in real-life network conditions is of great importance. Therefore, in our simulations we used the medium access control of IEEE 802.11, where delay and/or packet losses may occur frequently due to congestion and/or collisions. Note that when MAC layer broadcasts are used to transmit packets (which is the case for all the investigated techniques), packet collisions result in packet drops, since no acknowledgement mechanism is used. This fact, which is rarely addressed in the literature, severely affects the performance of the algorithms in conditions of MAC layer congestion. Collisions also have an impact on broadcasted periodic messages created by the underlying routing protocol. Therefore congestion may affect the accuracy of the routing protocol itself. To further approximate real-life conditions, we also used configurations of high mobility in order to assess the resilience of the algorithms to stale routing information. Finally, it should be noted that, in order to minimize stochastic artifacts, all presented results were obtained as average values over 10 independent simulation runs.

5.2. Simulation Model and Parameters

The simulation model used, consists of N nodes that roam in a rectangle area of $1000x1000$ m^2 or $1500x300$ m^2. For the simulation of node movement, the well-known Random Waypoint (RW) Algorithm has been used. The mobility scenarios were produced using the perfect simulation model proposed in [25] in order to avoid transient artifacts in nodes' movement. Furthermore, we also used node movement scenarios produced from real maps, in order to test the algorithms in more realistic scenarios. Those scenarios, which are also suitable for modeling *vehicular ad hoc networks*, were produced according to the procedure described in [26], using the provided map of a section of a major US city. The code for producing such scenarios is available online [27]. The roaming area, used with this methodology, was $1000x1000$ m^2. The provided scenario generation code sets automatically the speed of nodes according to the speed limit (maximum 35 miles/hour) of the roads included in the map. In the following, the scenarios produced from real maps will be referred as *city-section (CS) scenarios*.

Nodes are randomly organized into CL_{cnt} classes of equal size. A number $Q(Q < N)$ of nodes performs queries using a Poisson distribution with mean rate λ. Queries are propagated to a maximum number of hops (H_{max}). Each node chooses uniformly the class to which it addresses its queries. Finally, each node receiving a query replies with a packet of 1000 bytes in size. As far as the

Table 1: Simulation parameters

Number of nodes (N)	100 or 50
Region Size	1000x1000 or 1500x300
Maximum speed in RW algorithm	20 m/sec
RW pause time	0 secs
Node's Transmission Range (R)	250 m
Number of Classes (C_{cnt})	5
Number of querying nodes (Q)	0.2N
Query generation rate (λ)	0.1 query/sec
Maximum number of hops (H_{max})	3
Simulation Time	900 sec

size of the query header is concerned, we used 4 bytes for each field of the header although smaller sizes could be used for some fields. This has been done in order to capture the worst case performance for CL-QF. The transmission range of each node is R. The total simulation time was 900 seconds. The simulation parameters are summarized in Table 1. Unless stated otherwise, these values are valid throughout all experiments.

5.3. Evaluation metrics

Five different metrics have been used for the evaluation: a) *average number of packets per query*, i.e., the mean number of packets transmitted during the routing of a query (equivalently, the average number of nodes involved in the forwarding of the query), b) *average number of bytes per query*, i.e., the mean number of bytes transmitted during the routing of a query, c) *mean number of received replies*, i.e., the average number of replies to a query, received by the querying node, d) *delivery ratio*, i.e., the percentage of the generated replies that reached the querying node, and e) *mean delay*, i.e., the average delay for reply messages received by the querying node. The first two metrics evaluate the ability of the algorithms to utilize efficiently the limited system resources and provides an indication of the algorithm's scalability. Specifically, the first metric is used to capture the impact of the algorithm performance on network aspects such as power consumption, network access delay, etc. Such aspects are of great importance in mobile wireless networks that use contention mechanisms. The second metric captures the consumed bandwidth. The third metric evaluates the ability of the algorithms to collect replies from the target nodes. Finally, the fourth and fifth metric identify the impact of the query forwarding schemes on the ability of the underlying routing protocol to deliver data and minimize the involved delay.

5.4. Experiments

In order to evaluate the algorithms, we conducted four experiments, namely: a) the impact of classes, b) performing queries with horizon, c) the impact of network size and d) resilience to mobility. In the first experiment we evaluate the impact of having different number of classes in the network, i.e., we vary

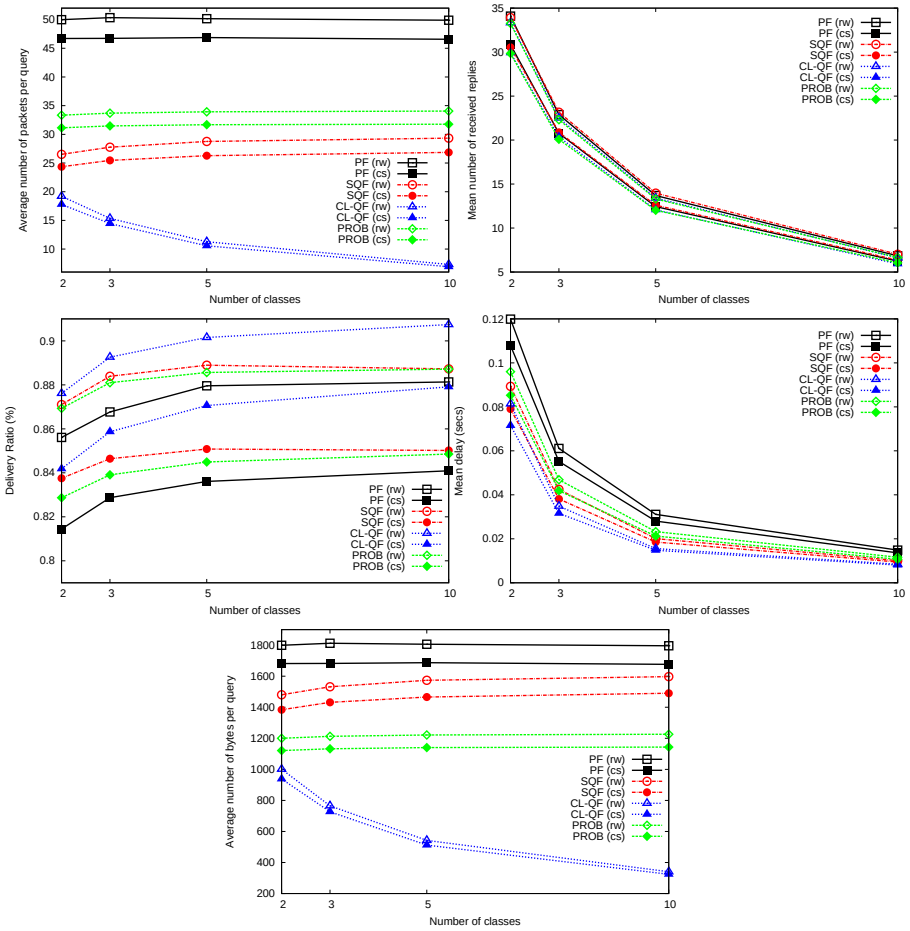


Figure 6: Performance with respect to the number of classes in the network

C_{cnt}. In the second experiment we vary H_{max}, thus we perform queries with different propagation limit in order to highlight the performance of the algorithms over longer and therefore more volatile paths. Then, in the third experiment we assess the algorithms' performance for different number of network nodes. Network density affects the number of neighbors that a node has as well as the MAC layer congestion conditions that may be occur. Finally, in the fourth experiment the mobility of nodes is varied in the RW mobility model. Mobility determines the accuracy of the underlying routing protocol, therefore affects the performance of algorithms.

6. Simulation Results

6.1. The Impact of Classes

In this experiment, we vary the number of the classes into which network nodes are organized. The purpose of this experiment is to directly capture the

impact of using application layer information on the ability of each scheme to scale with the number and the size of classes. Since the number of network nodes is fixed, the size of a class is reduced with the number of classes in the network. Fig. 6 presents the performance of all schemes with respect to the number of classes for a network of 100 nodes in a $1000x1000$ m^2 area, for both RW and CS mobility models. Regarding the PROB scheme, we present its performance for $k = 1$ and $p = 0.35$. Extensive simulations have identified that these values yield the best performance for both the RW and CS scenarios. Clearly, CL-QF outperforms all schemes with respect to the average number of packets per query. As actually expected, the performance of PF does not change as the number of classes increases, since it always asks all the peers of the network. The same applies to SQF as well as PROB. The improvement in the case of SQF is ascribed to its ability to take advantage of routing information in order to group target nodes and minimize transmissions. On the other hand, although PROB does not exploit routing information, it manages to perform close to SQF. However, the disadvantage of PROB lies in determining the suitable parameter values. Contrary to the constant performance of all other schemes, CL-QF improves its performance when the number of classes increases. This is due to the fact, that the proposed method takes advantage of class information and tries to reach only the peers of the target class. This trend can also be confirmed by the analysis of Section 4. Note that as the number of classes increases, p_c decreases. Consequently, $(1 - p_c)^{|T_i^v|-1}$ increases and according to Eq. 12, D^{CL} decreases. It is worth noticing that even when only two classes exist, i.e., the query concerns half of network nodes, CL-QF manages a reduction of approximately 62%, 42% and 28% compared to PF, PROB and SQF respectively. As far as the number of bytes per query is concerned, the performance of the schemes remains qualitatively similar. Although CL-QF appends the query header in each query message, its performance is still better than that of the other schemes because CL-QF achieves a massive reduction on the transmitted packets. The relation of the average number of packets per query and the average number of bytes per query is similar in all of the experiments, therefore the latter will be omitted in the following for presentation purposes.

At the same time, the ability of CL-QF to discover and collect replies is practically equivalent to that of the other schemes, as indicated by the mean number of received replies. The small difference (less than 3%) compared to the best performance (PF) can be attributed to the fact that in PF each node has the opportunity to receive multiple copies of the query since all nodes forward it. Therefore, the resilience in mobility is higher. However, the advantage of PF is compensated since packets are transmitted using MAC layer broadcast which results in packet losses due to collisions and consequently in performance degradation. On the contrary, although CL-QF uses a single path to deliver the packet to a target node, and therefore is more prone to mobility, it manages to preserve its effectiveness since packet collisions are minimized. The reduction of received replies as the number of classes increases, for all schemes, is a result of the smaller number of nodes belonging to each class. The increased number of transmissions in the case of PF, PROB and SQF has also a negative impact on

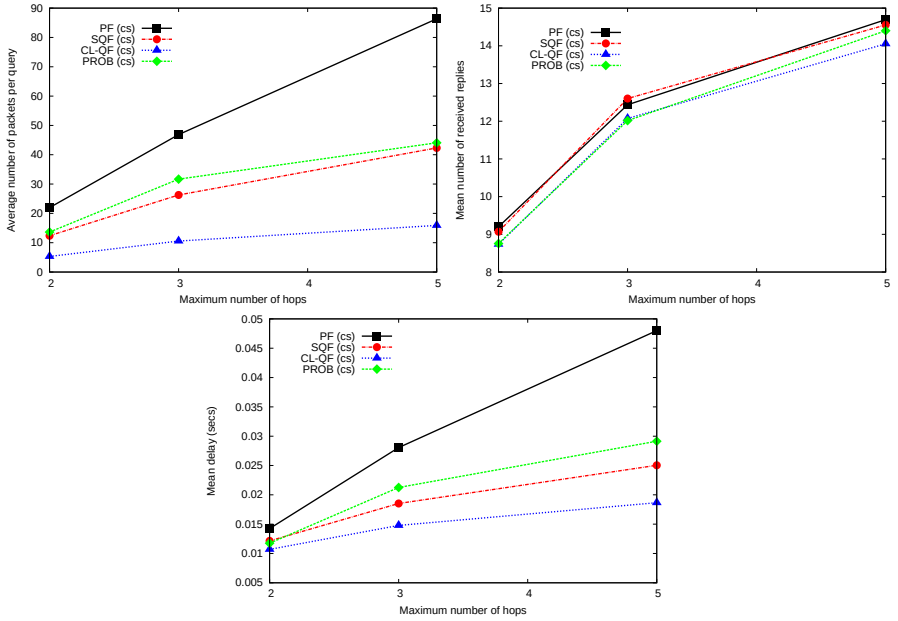


Figure 7: Performance vs maximum number of hops that a query is allowed to propagate

the performance of the underlying routing protocol in terms of delivery ratio and mean delay. This is the result of collisions between query packets and topology update messages used by all proactive routing protocols. The slight increase of delivery ratio and the decrease of mean delay, as the number of classes increases, is the result of less reply messages (due to the smaller class size), which reduces congestion. As explained previously, the impact of the query routing scheme in the operation of the underlying protocol is of paramount importance for supporting multiple data-centric applications over MANETs.

A final observation regarding the illustrated results is that they are qualitatively similar in both RW and CS mobility models even though in the CS scenarios the performance is slightly degraded for all schemes as a result of the increased mobility of nodes (maximum speed is 35 miles/hour). This very interesting result has been also be evident in all of the experiments. Therefore, for simplicity, in the following we will provide only results for the more challenging CS scenarios.

6.2. Performing Queries with Horizon

In this experiment, queries are limited within a certain network distance (horizon), measured in hops. This can be done by using the hop count information available by the underlying routing protocol. The objective of this experiment is to illustrate the performance of the proposed schemes on progressively longer and therefore more volatile paths. Another motivation for this experiment has been to investigate the impact of the schemes with respect to the size of the network area that a query affects. In this experiment we provide results for a network of 100 nodes in a $1000x1000$ m^2 area for the CS mobility

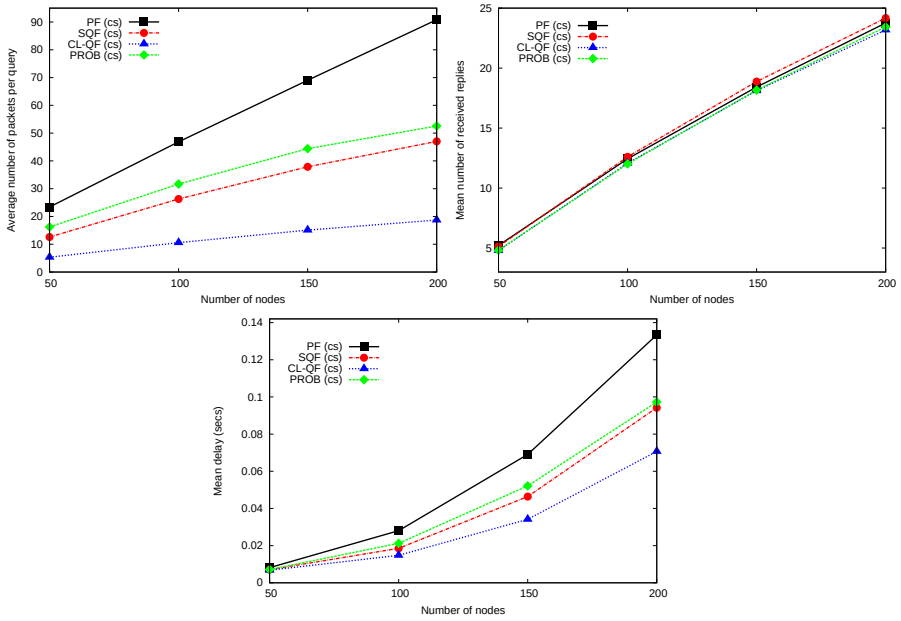


Figure 8: Performance in networks of different size

model (fig. 7). In the presented results we illustrate always the best performance of PROB. It turns out that this can be accomplished for $k = 0$ and $p = 0.6$ when the query is limited to two hops and for $k = 1$ and $p = 0.35$ when the maximum hops are either 3 or 5. It should be noted that when only one hop is allowed, PROB is equivalent to PF. CL-QF again outperforms all schemes in terms of the average number of packets per query. Specifically, CL-QF achieves an improvement of up to $\sim 82\%$, $\sim 65\%$ and $\sim 62\%$ compared to PF, PROB and SQF respectively. The main observation is that the gain of CL-QF increases with the number of maximum hops. This can be easily explained by the following; forwarding nodes away from the originator are probable to be assigned with less target nodes because the part of the network that is still uncovered is smaller. Therefore, there is a high probability that they will cancel the query forwarding. This also complies with the analysis in Section 4 (Eq. 12 and 13); $|T_i^v|$ tends to be smaller as i increases, which is the case in our experiment, therefore $(1 - p_c)^{|T_i^v|-1}$ increases since $1 - p_c \leq 1$. Consequently, the gain of CL-QF increases with i. Similar conclusions apply when comparing with PF and PROB. Although PROB performs similar to SQF, the cost of PF, as expected, increases with the number of hops. As far as the mean number of received replies is concerned, CL-QF manages to keep up with the other schemes although it is more vulnerable to mobility. This is because CL-QF produces less congestion and less packet drops due to collisions. Consequently, the losses due to mobility are compensated. It is interesting that this is true even for long paths which are more prone to failures due to mobility than shorter ones. Finally, as far as the mean delay for delivering replies is concerned, the superiority of CL-QF is confirmed. The explanation of this result is twofold. On one hand, CL-QF minimizes the

number of transmitted packets and therefore reduces the collisions with routing control messages. As a result, the underlying routing protocol maintains an up-to-date view of the network and uses the shortest paths. On the other hand, less transmissions produce less contention in the medium and less congestion in nodes, resulting in the minimization of end-to-end delay. When more hops are involved in the propagation of a query, more distant nodes are reached and therefore produce a reply. As a result, the mean delay increases for all schemes since end-to-end delay is dominated by the number of hops that reply packets travel.

6.3. The impact of Network size

The purpose of this experiment is to evaluate the ability of the schemes to scale in large networks. Therefore, we vary the number of nodes that the network consists of. At the same time, since the roaming area is fixed, the network size affects its density and therefore aggravates the congestion conditions that may occur. Fig. 8 presents the performance of all schemes with respect to the network size. Results are presented for a roaming area of $1000x1000$ m^2. Regarding PROB, its best performance results for $k = 1$ and $p = 0.35$ when the network consists of either 50 or 100 nodes, for $k = 1$ and $p = 0.3$ when the network size is 150 nodes and for $k = 1$ and $p = 0.2$ in a 200-node network. CL-QF outperforms all schemes with respect to the average number of packets per query. Moreover, it exhibits a remarkable scaling ability with respect to the network size. As the number of nodes increases, the difference between CL-QF and the other schemes increases. Clearly, this is an attribute of the cross-layer design that allows CL-QF to eliminate nodes not eligible for receiving the query. The increase in the average number of packets per query with the increase of network size for the CL-QF case is due to the increase of target nodes. This is also evident in the figure illustrating the mean number of received replies. More replies are received since the fixed number of classes results in more nodes per class, therefore more target nodes. Again all schemes illustrate approximately the same ability to discover and collect replies for both the mobility models.

As far as the mean delay is concerned, it is clear that the lightweight nature of CL-QF supports the unhindered operation of the underlying routing protocol. When CL-QF is employed, the performance improvement can reach up to an impressive 47% compared to PF, 34% compared to PROB and up to $\sim 25\%$ compared to SQF. It is more important that the maximum improvement is achieved in bigger networks where delay is higher. This is reasonable since delay is mainly affected by increased contention and/or congestion. Such conditions are more frequent in bigger networks. This is why in networks of 50 nodes all schemes perform closely while in large networks CL-QF, due to its resilience to congestion, dominates the other schemes.

6.4. Resilience to mobility

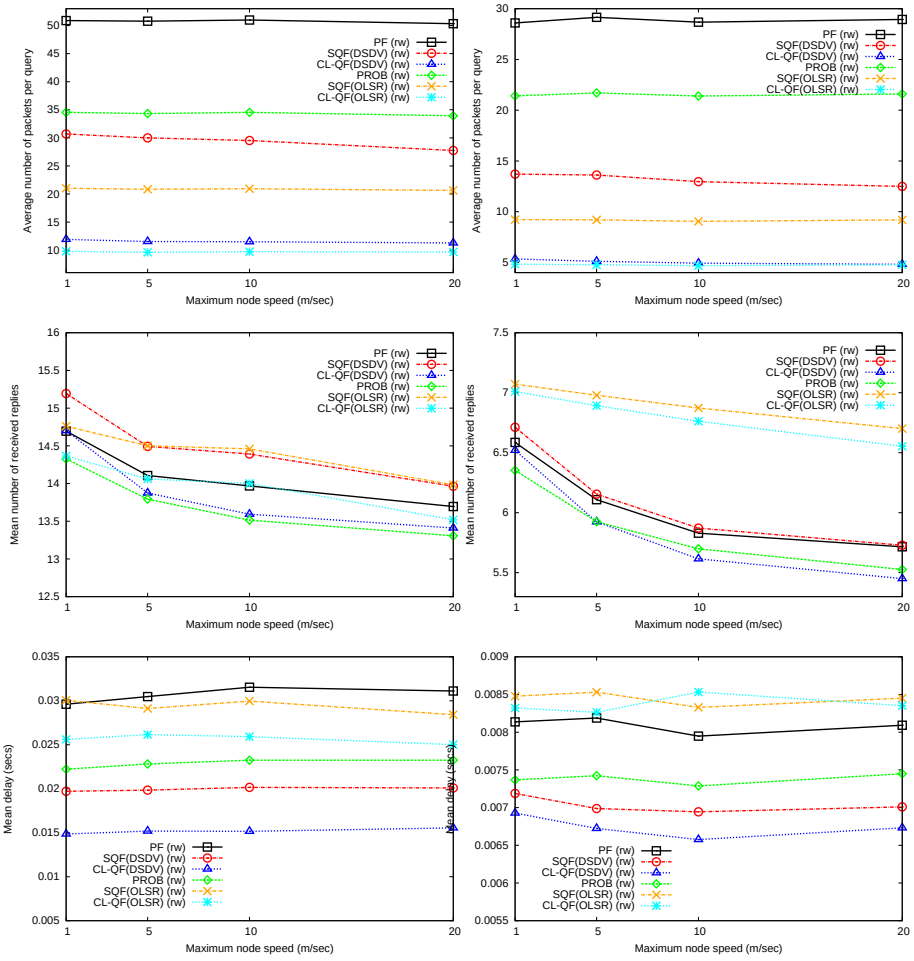
In this last experiment we vary the maximum speed at which nodes are roaming. The purpose of this experiment is to evaluate the system's perfor-

mance under various levels of mobility that affect the underlying routing protocol and therefore the query routing schemes built on top. Furthermore, in this experiment we also evaluate an implementation of SQF and CL-QF that is based on the well-known OLSR protocol. The purpose is to assess the way routing protocols that exhibit different levels of resilience to mobility affect the performance of the proposed techniques.

At this point it must be noted that in this experiment only the RW mobility model is used since the CS model does not allow the definition of speed by the user. Specifically, we use a network of 100 nodes roaming in a $1000x1000$ m^2 area and a network of 50 nodes roaming in a $1500x300$ m^2 area. The two settings are chosen to represent approximately the same node density. However, the second setting is more prone to mobility since, in this case, paths are in general longer. The best performance for PROB is achieved for $k = 1$ and $p = 0.35$ in the $1000x1000$ m^2 case, while in the $1500x300$ m^2 case the respective values are $k = 1$ and $p = 0.3$.

Fig. 9(a) illustrates results obtained for the scenario of $1000x1000$ m^2, while fig. 9(b) depicts the results for the $1500x300$ m^2 case. With respect to the average number of packets per query, again CL-QF manages a massive improvement compared to the other schemes regardless of whether the underlying routing protocol is DSDV or OLSR. The improvement in the CL-QF(DSDV) case compared to PF is $\sim 78\%$ for the $1000x1000$ m^2 area and $\sim 82\%$ for the $1500x300$ m^2 area. Compared to PROB, the improvement is $\sim 66\%$ and $\sim 77\%$ respectively. Similar performance is witnessed for CL-QF(OLSR). However, the improvement over SQF is smaller in the case that OLSR is used. This is a result of the nature of OLSR. Due to the use of multi-point relays, more destinations are served by the same next hop. As a result, transmissions are reduced even without the use of class information. However, it can be seen that even in this case the improvement of CL-QF reaches up to $\sim 50\%$ in both scenarios.

The average number of packets per query is practically stable over the entire mobility range. Although this is somewhat expected for PF and PROB, for CL-QF and SQF this is a very interesting result since it indicates that query routing is not significantly affected by link failures. This is confirmed by the mean number of received replies where all schemes present similar performances for all the settings with the notable exception of the OLSR-based schemes that perform significantly better. The limited hysteresis of CL-QF(DSDV) compared to both PF and SQF(DSDV) is the result of the increased failure rate of DSDV due to mobility. Indeed, when the more reliable OLSR is used as the underlying protocol, CL-QF improves its performance in all aspects, outperforms all other schemes in the $1500x300$ m^2 scenario and performs similar to PF in the $1000x1000$ m^2 case. It is clear that SQF always delivers more replies compared to CL-QF (remember that SQF ignores class information when forwarding a query). However, the performance hysteresis is small (approx. 3% in the worst case) and it is the price paid for minimizing the number of transmitted packets (by reducing them to roughly 50% of SQF's cost) during the routing of a query. The slight decrease in the performance of all schemes can be ascribed to the increasing rate of link failures due to mobility.



(a) 100 nodes in $1000x1000$ m^2 area (b) 50 nodes in $1500x300$ m^2 area

Figure 9: Performance under variable user's mobility.

Finally, the lightweight nature of CL-QF has a beneficial effect on the performance of the underlying routing protocol in terms of mean delay. Again, this is a result of minimizing congestion when CL-QF is utilized since smaller queues are formed in each node therefore smaller delays are incurred. This is evident especially for CL-QF(DSDV) which achieves the lowest delays in both scenarios. It is interesting that the delays in the case of CL-QF(OLSR) are relatively high. However, this is a result of its ability to deliver more replies and not a result of increased congestion. Since more distant nodes are harder to reach, other schemes collect replies from closer nodes. On the contrary, the increased ability of CL-QF(OLSR) to deliver replies means that more replies from distant nodes are collected. This has an impact on the average delay since delay is dominated by the number of hops that a reply travels.

7. Related Work

Data-centric communication depends on the application layer. Therefore, the key for providing such communication services is to incorporate application layer specifics into networking functions. This particular task is not straightforward considering existing wired infrastructures such as the Internet, and gets even more complicated in the case of mobile ad-hoc networks. Recently, many researchers have focused on the implementation of efficient data-centric protocols in mobile ad-hoc networks in order to cope with the problem of limited network resources.

Most of the reported research concerns p2p file and/or resource sharing applications [3], [28],[29], [30]. In this scenario, resources/files are available in the network. Users produce lookups in order to locate one node that possess the requested resource/file. Several solutions have been proposed for efficiently routing lookup messages in wireless mobile ad-hoc networks [3], [28], [29]. Furthermore, in the same context, scientists have proposed ancillary procedures for the dissemination and replication of resources [29], [30]. The communication paradigm introduced by this kind of applications is not compatible to the one of peer databases. The objective of routing in the case of peer databases is to deliver the query to *as many eligible nodes as possible*, so that as much information as possible is collected. On the contrary, the routing of a lookup in file/resource sharing applications aims at locating a single user that possesses the requested data. Furthermore, ancillary mechanisms such as data dissemination and replication can not be implemented efficiently in the sense that in peer databases each peer carries its personal data that are probably highly volatile and time variant (e.g., a car's speed, a pedestrian's position, etc).

Another example of data-centric communication comes from sensor networks. Several routing protocols have been proposed to efficiently collect data from sensor nodes [31]. Furthermore, application layer functions such as data fusion and data aggregation are introduced at the network layer in order to optimize the overall system operation. Although collecting data from sensor nodes bears some similarities to the routing of queries in peer databases, there are also

considerable differences. In sensor networks the queries are sink-initiated, meaning that only sink nodes are allowed to collect data contrary to peer databases where all nodes may spontaneously pose queries. Furthermore, in sensor networks, collected data are not volatile therefore allowing the routing protocol to establish paths toward specific sensor nodes. This is also accommodated by the fact that sensor nodes are not mobile. On the contrary, a peer database carries volatile data and is located in a mobile node. Therefore, the routing protocol must constantly locate target nodes and route efficiently the query. Finally, sensor networks are special purpose networks that allow for specialized routing protocols. On the other hand, our approach aims at providing a routing mechanism able to efficiently coexist with other applications and protocols.

Concerning the domain of peer to peer databases, there have been several interesting approaches that aim towards facilitating queries that collect as many data as possible from the peer network (in constrast to single lookup queries that try to locate a certain resource inside a peer network). Mobility is not really applicable to the systems in the related literature since, in all of them, data-centric communication takes place on overlay networks that rely on conventional wired infrastructures. In particular, Hyperion [2] and Piazza [32] assume unstructured overlay support, while Mercury [33] and PIER [34] are based on structured overlays. The two former systems place a particular emphasis to the issue of schema mappings; despite the theoretical elegance of the LAV,GAV or GLAV mappings that these systems employ, we believe that composing a quadratic number of mappings with respect to the number of involved peers is impractical for mobile peers (thus, we resort to a set of commonly agreed classes for peers in this paper). The later two approaches further rely on a full indexing technique. PISCES [35] also assumes a structured overlay, while proposing a partial indexing approach towards dealing with the scale up of the amount of available data. The approach proposed in BestPeer [36], which is thematically targeted to skyline queries, may be applied on top of structured or unstructured overlays. None of the aforementioned studies addresses the problem of cross-layer routing of queries in the context of a MANET.

Regarding peer to peer databases on top of mobile ad-hoc networks, [6] presents an excellent survey. However, it is mainly focused on publish-subscribe works on the assumption that a peer may not know the identities of other peers in the network and the data they store. Therefore, routing in the traditional MANET sense is not considered. [37] is a good example of an approach customized for resource discovery. Moreover, [37] operates in an environment where no routing is performed, but rather the application layer is responsible for the communication of peers. In this context, peers broadcast both their queries and their internal reports; at the same time they cache other peers's reports that they receive for future use. In our work, we operate on a significantly different assumption and assume that routing between peers in a MANET (or a mesh network) is feasible. Although operating without routing infrastructure offers the advantage of avoiding the involved overheads, we believe that operating on top of existing routing protocols, especially in the case of MANET/mesh networks is both feasible and useful. We also need to point out that instead of

working with simple lookup queries (where the publish subscribe mechanisms discussed in [6] might be useful) we are interested in maximum result collection queries, with an emphasis on volatile peers. This makes data broadcasting and caching expensive and not so useful as compared to exploiting well-known proactive routing protocols for economizing on the bandwidth used.

In [38], a first discussion of our cross-layer approach for routing queries has been introduced. In this work, we extend the work of [38] in several respects. First, we introduce a analytical model for analyzing the performance of CL-QF. The results of this analysis are in coherence with the provided simulation study. Second, another contribution of this work is the introduction of the simple query forwarding (SQF) scheme as a means of highlighting the gains of the cross-layer design. To this end, SQF is included for comparison in the provided simulation results as well as in the proposed analysis in order to differentiate between the gains related to the cross-layer design and the improvements caused by efficiently utilizing the wireless medium. Third, we extend the simulation study in order to: i) support more evaluation metrics (i.e., the delay of delivering data, the cost of each scheme in bytes), ii) evaluate the proposed schemes with respect to more system parameters, and iii) emphasize on realistic simulations by including mobility models that use real maps.

8. Conclusions

In this paper we have proposed a novel networking mechanism for mobile ad-hoc networks, named CL-QF. The new protocol incorporates application layer specifics in order to optimize the utilization of scarce network resources by suppressing redundant transmissions. Moreover, CL-QF exploits the synergy with traditional routing protocols in order to minimize both complexity and signaling and allow the network to seamlessly support both traditional and data-centric communications. CL-QF has been evaluated through extensive simulations which proved its ability to minimize the number of transmitted packets during the routing of a query without sacrificing the effectiveness of the algorithm to collect as many replies as possible. More specifically, CL-QF achieved a massive reduction of up to $\sim$85% and $\sim$78% compared to plain flooding and probabilistic forwarding respectively in a network of only ten classes. Even when only two classes were present in the network, the improvement reached up to $\sim$62% and $\sim$42% respectively. Similar improvements were obtained in networks of increasing size and when queries with increasing horizon were used. Finally, the lightweight implementation of CL-QF did not hinder the efficient operation of the underlying routing protocol with respect to the delivery ratio while it systematically improved the mean delay.

References

[1] S. Androutsellis-Theotokis, D. Spinellis, A survey of peer-to-peer content distribution technologies., ACM Comput. Surv. 36 (4) (2004) 335–371.

[2] M. Arenas, V. Kantere, A. Kementsietsidis, I. Kiringa, R. J. Miller, J. Mylopoulos, The Hyperion project: from data integration to data coordination, ACM SIGMOD Record 32 (3) (2003) 53–58. doi:http://doi.acm.org/10.1145/945721.945733.

[3] H. Pucha, S. Das, Y. Hu, Ekta: An efficient DHT substrate for distributed applications in mobile ad hoc networks, in: In Proc. of the 6th IEEE Workshop on Mobile Computing Systems and Applications (WMCSA 2004)., 2004.

[4] V. Bychkovsky, K. Chen, M. Goraczko, H. Hu, B. Hull, A. Miu, E. Shih, Y. Zhang, H. Balakrishnan, S. Madden, Data management in the CarTel mobile sensor computing system., in: SIGMOD Conference, 2006, pp. 730–732.

[5] B. Xu, O. Wolfson, Data management in mobile peer-to-peer networks, in: DBISP2P, 2004, pp. 1–15.

[6] Y. Luo, O. Wolfson, Mobile p2p databases, in: Encyclopedia of GIS, 2008, pp. 671–677.

[7] F. Giunchiglia, I. Zaihrayeu, Making peer databases interact - a vision for an architecture supporting data coordination, in: Cooperative Information Agents VI, 6th International Workshop, CIA 2002, Madrid, Spain, September 18-20, 2002, 2002, pp. 18–35.

[8] V. Kantere, I. Kiringa, J. Mylopoulos, A. Kementsietsidis, M. Arenas, Coordinating peer databases using ECA rules, in: Databases, Information Systems, and Peer-to-Peer Computing, First International Workshop, DBISP2P, Berlin Germany, September 7-8, 2003, 2003, pp. 108–122.

[9] M. Khambatti, K. Ryu, P. Dasgupta, Efficient discovery of implicitly formed peer-to-peer communities, International Journal of Parallel and Distributed Systems and Networks 5 (4) (2002) 155–164.

[10] N. Folinas, P. Vassiliadis, E. Pitoura, E. Papapetrou, A. Zarras, Context-aware query processing in ad-hoc environments of peers, Journal of Electronic Commerce in Organizations 6 (1) (2008) 38–62.

[11] D. Athanasopoulos, A. Zarras, V. Issarny, E. Pitoura, P. Vassiliadis, Cowsami: Interface-aware context gathering in ambient intelligence environments, Pervasive and Mobile Computing 4 (3) (2008) 360–389.

[12] J. Jetcheva, D. B. Johnson, Adaptive demand-driven multicast routing in multi-hop wireless ad hoc networks., in: MobiHoc, 2001, pp. 33–44.

[13] M. Abolhasan, T. Wysocki, E. Dutkiewicz, A review of routing protocols for mobile ad hoc networks, Ad Hoc Networks 2 (2004) 1–22.

[14] R. H. Boivie, N. K. Feldman, C. Metz, On the wire - small group multicast: A new solution for multicasting on the internet, IEEE Internet Computing 4 (3) (2000) 75–79.

[15] Cortex-a9 processor (2011).
URL http://www.arm.com/products/processors/cortex-a/cortex-a9.php

[16] L. M. Feeney, M. Nilsson, Investigating the energy consumption of a wireless network interface in an ad hoc networking environment, in: INFOCOM, 2001, pp. 1548–1557.

[17] L. Wang, J. Manner, Energy consumption analysis of wlan, 2g and 3g interfaces, in: Proceedings of the 2010 IEEE/ACM Int'l Conference on Green Computing and Communications & Int'l Conference on Cyber, Physical and Social Computing, GREENCOM-CPSCOM '10, IEEE Computer Society, Washington, DC, USA, 2010, pp. 300–307. doi:http://dx.doi.org/10.1109/GreenCom-CPSCom.2010.81.
URL http://dx.doi.org/10.1109/GreenCom-CPSCom.2010.81

[18] C. Bettstetter, G. Resta, P. Santi, The node distribution of the random waypoint mobility model for wireless ad hoc networks, IEEE Transactions on Mobile Computing 2 (3) (2003) 257–269. doi:http://dx.doi.org/10.1109/TMC.2003.1233531.

[19] C. Bettstetter, Topology properties of ad hoc networks with random waypoint mobility, SIGMOBILE Mob. Comput. Commun. Rev. 7 (3) (2003) 50–52. doi:http://doi.acm.org/10.1145/961268.961287.

[20] Z. J. Haas, J. Y. Halpern, L. Li, Gossip-based ad hoc routing, IEEE/ACM Trans. Netw. 14 (3) (2006) 479–491. doi:http://dx.doi.org/10.1109/TNET.2006.876186.

[21] K. Fall, K. Varadhan, The ns manual, VINT Project, Univ. California, Berkeley, CA (2001).
URL http://www.isi.edu/nsnam/ns/ns-documentation.html

[22] C. E. Perkins, P. Bhagwat, Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers., in: SIGCOMM, 1994, pp. 234–244.

[23] T. Clausen, P. Jacquet, Optimized link state routing protocol (olsr), rfc3626.

[24] Um-olsr implementation, http://masimum.dif.um.es/?Software:UM-OLSR.

[25] J.-Y. L. Boudec, M. Vojnovic, Perfect simulation and stationarity of a class of mobility models, in: INFOCOM '05: Proceedings of 24th Annual Joint Conference of the IEEE Computer and Communications Societies, IEEE, 2005. doi:10.1109/INFCOM.2005.1498557.

[26] A. K. Saha, D. B. Johnson, Modeling mobility for vehicular ad-hoc networks, in: VANET '04: Proceedings of the 1st ACM international workshop on Vehicular ad hoc networks, ACM, New York, NY, USA, 2004, pp. 91–92. doi:http://doi.acm.org/10.1145/1023875.1023892.

[27] A. K. Saha, D. B. Johnson, Realistic mobility modelling for mobile ad hoc networks, Available online at http://www.cs.rice.edu/~amsaha/Research/MobilityModel/ (2007).

[28] Y. C. Hu, S. M. Das, H. Pucha, Exploiting the synergy between peer-to-peer and mobile ad hoc networks, in: HOTOS'03: Proceedings of the 9th conference on Hot Topics in Operating Systems, USENIX Association, Berkeley, CA, USA, 2003, pp. 7–7.

[29] D. Chakraborty, A. Joshi, Y. Yesha, Integrating service discovery with routing and session management for ad-hoc networks., Ad Hoc Networks 4 (2) (2006) 204–224.

[30] A. Datta, S. Quarteroni, K. Aberer, Autonomous gossiping: A self-organizing epidemic algorithm for selective information dissemination in wireless mobile ad-hoc networks., in: ICSNW, 2004, pp. 126–143.

[31] K. Akkaya, M. F. Younis, A survey on routing protocols for wireless sensor networks., Ad Hoc Networks 3 (3) (2005) 325–349.

[32] I. Tatarinov, Z. Ives, J. Madhavan, A. Halevy, D. Suciu, N. Dalvi, X. Dong, Y. Kadiyska, G. Miklau, P. Mork, The piazza peer data management project, SIGMOD Record 32 (3) (2003) 47–52.

[33] A. Bharambe, M. Agrawal, S. Seshan, Mercury: Supporting scalable multi-attribute range queries, in: ACM SIGCOMM Applications, Technologies, Architectures, and Protocols for Computer Communication, 2004, pp. 353–366.

[34] R. Huebsch, J. M. Hellerstein, N. Lanham, B. T. Loo, S. Shenker, I. Stoica, Querying the internet with PIER, in: 29th International Conference on Very Large Data Bases (VLDB), 2003, pp. 321–332.

[35] S. Wu, J. Li, B. Ooi, K.-L. Tan, Just-in-time query retrieval over partially indexed data on structured p2p overlays, in: ACM SIGMOD International Conference on Management of Data, 2008, pp. 279–290.

[36] S. Wang, B. Ooi, A. Tung, L. Xu, Efficient skyline query processing on peer-to-peer networks, in: IEEE 23rd International Conference on Data Engineering (ICDE), 2007, pp. 1126–1135.

[37] O. Wolfson, B. Xu, H. Yin, H. Cao, Search-and-discover in mobile p2p network databases, in: ICDCS, 2006, p. 65.

[38] E. Papapetrou, E. Rova, A. Zarras, P. Vassiliadis, Cross-layer networking for peer databases over wireless ad-hoc communities, in: IEEE International Conference on Communications, ICC, IEEE, 2007, pp. 3443–3448.