A Gossip-based Distributed News Service for Wireless Mesh Networks

Daniela Gavidia
Faculty of Science
Department of Computer Science
Vrije Universiteit Amsterdam
De Boelelaan 1081a, 1081 HV
Amsterdam, The Netherlands
Email: daniela@cs.vu.nl

Spyros Voulgaris
Faculty of Science
Department of Computer Science
Vrije Universiteit Amsterdam
De Boelelaan 1081a, 1081 HV
Amsterdam, The Netherlands
Email: spyros@cs.vu.nl

Maarten van Steen
Faculty of Science
Department of Computer Science
Vrije Universiteit Amsterdam
De Boelelaan 1081a, 1081 HV
Amsterdam, The Netherlands
Email: steen@cs.vu.nl

***Abstract*— The prospect of having an easily-deployable, self-configuring network for a relatively low investment has made wireless mesh networks an attractive platform to provide wireless services. With the significant attention currently placed on wireless mesh networking, deployment of these mesh networks may be imminent. However, even with the infrastructure in place, development of flexible middleware has yet to reach a level where it can deliver on the promise of mesh networks due to the special characteristics of wireless connectivity. In this paper, we propose a fully decentralized news service based on epidemics. Its simple design makes for a scalable and robust solution, flexible enough to be used as the basis for other more sophisticated applications.**

I. Introduction

As advances in wireless networking continue, we are gradually seeing a shift in which distributed (middleware) systems are moving from wired networks to heterogenous or completely wireless systems. Notably, wireless mesh networks [1] offer the facilities to quickly and cheaply set up a networking infrastructure that can easily span the size of a city. From a distributed systems perspective, the challenge lies in providing services that can hide the inherent unreliable nature of the underlying infrastructure. This unreliability is caused by failing links and a relatively high rate of joining and leaving nodes (purposefully or unintentionally), which continuously affect the topology of the network.

This instability requires that we seek new solutions to well-known problems. As a step in that direction, we are exploring how gossiping protocols can help in the construction of highly robust services. In this paper, we consider the problem of providing a *news service* that runs entirely on a wireless mesh network. This service provides mobile users news items that are of interest to them. In our approach, we assume that a user, by means of a PDA or a similar small device, can connect to an access point (i.e., router) of a wireless mesh network. When connected, the user can read news items as if accessing a central database where all items are stored. Using content-based filtering, for example by means of SQL-like queries, only the items of interest will be delivered.

The problem we address can best be described as setting up a simple, self-configuring news service in a mesh network, under the condition that it be fully decentralized. The reasons for avoiding a centralized implementation are, in a way, related to the nature of mesh networks. Wireless mesh networks are based on the principle of cooperation between routers, most notably exemplified by routers forwarding packets on behalf of other nodes. With that in mind, we want to steer away from a centralized solution where one node is solely responsible for the availability of the service. A decentralized solution effectively divides the workload (and responsibility) among the collection of nodes providing the service, allowing us to sidestep issues that may arise from having a single point of failure and single ownership of the service. We outline the requirements for a successful implementation of our distributed news service as follows:

- **Ease of deployment** A collection of nodes should be able to start providing the service with minimal configuration. Nodes should be able to join the system without going through complicated bootstrapping mechanisms. In essence, we desire to have a decentralized system where nodes can start making a contribution to the service as soon as they are operational.
- **Minimal requirements** Contributing to the service should not be a burden to the nodes in the mesh network. Memory and computational requirements should be small enough to allow any router to be part of the service. No powerful nodes are expected to be in place for high-performance tasks.
- **Robustness** The system should be minimally affected by nodes joining and leaving the network. Moreover, recovery from significant changes in membership should be prompt.
- **Scalability** The service should be able to perform adequately in the face of increasing number of nodes and news items being published.
- **Effectiveness** When an item is published, it should be made available to the interested users in a timely manner.

We expect to meet these requirements by having the routers in the mesh backbone exchange news items using the epidemic protocol we introduce in Section III. Epidemic (or gossip-

based) techniques have proved to be a robust, efficient, and scalable solution for disseminating information in peer-to-peer networks [2], [3], [4]. Aside from the robustness and scalability inherent to gossiping, the protocol we present is characterized by simple, independent one-to-one interactions. The simplicity of this approach allows any router willing to participate in the service to start contributing as soon as they come in contact with a router that is already providing the service. By basing our solution on gossiping, we expect to be less vulnerable to topology changes.

Our main contribution is that we embrace the unpredictable nature of wireless networks and attempt to use this to our advantage by implementing a gossip-based solution. Our approach skews deterministic routing in favor of probabilistic delivery of news. As a result, we can deliver a scalable and robust service with predictable behavior for large-scale deployments.

The remainder of this paper is organized as follows. In the next section, we describe our system model, specifying the assumptions we make and describing an application scenario for the news service. Section III details the implementation of the gossip protocol executed by the network of mesh routers. In Section IV, the performance of the service from the user's perspective is analyzed. Related work is discussed in Section V. Section VI presents a discussion followed by conclusions and final remarks in the last section.

II. System Model

The service we propose is provided by a mesh backbone composed of a large number of wireless routers. Users running the news service are able to publish events, which we call *news items*, of interest to other users. These users carry around *clients*, which are portable devices capable of connecting to the mesh backbone to retrieve news items. Essentially, the clients poll the routers for news items matching the interests of users. By specifying their preferences in advance and using them for filtering, users are able to receive in their portable devices only relevant news items.

When initially contacting a mesh router, clients are expected to send a filter to be used to identify the items of interest to the user. As long as the client maintains a connection to the router, it will receive updates whenever new items that match the users interests are received. Filtering is done at the router to avoid excessive communication with the client devices, which may have limited power supplies. Filters are not propagated through the network.

A. Assumptions

We assume the presence of a large collection of mesh routers forming a mesh backbone. These mesh routers are not mobile and, as a whole, provide coverage for an extensive geographical area. As part of the fixed infrastructure, they do not have strict constraints on power consumption. We expect these routers to have a dedicated amount of memory space to be used for storing news items. These caches will be updated periodically using the gossip protocol described in Section III.

News items are propagated through the network in the form of *news entries*. While a news item is a piece of information, a news entry is the representation of the news item in the network and for each news item several news entries may exist. The dissemination of news entries is done primarily within the mesh backbone. Each router can communicate wirelessly with the routers within its range. These routers are called its *neighbors*. A unique *id* is associated with each router. The entries that a router inserts into the network can be uniquely identified by a combination of the router id and a sequence number. In its most basic form, a news entry contains a unique id, a timestamp and a time-to-live. There may be other fields of information depending on the application. A limited number of these entries can be stored by each router in its local *cache*. In our experiments, the size of the cache is defined by the parameter c, which is the same for all routers. The storage capacity of the network as a whole is then $N \times c$, where N is the number of routers in the network. Routers in the network gossip periodically, exchanging the entries in their caches. We define a *round* as a gossiping interval in which each router initiates an exchange once.

The clients in our system are, for the most part, portable devices, such as phones, laptops or PDAs. These devices have limited power supplies and, for that reason, do not participate actively in the dissemination of news items. They do, however, engage in communication with the routers to be updated on news events.

B. Application Description

To illustrate the usefulness of the service, we propose a possible application scenario: advertising in a shopping center where products on sale need to be promoted. In this scenario, routers could be located at any other shop. Some routers may already be in place for use as hotspots or as part of a store's accounting system. As computers have become prevalent in business environments, we do not expect lack of infrastructure to be a major obstacle for the deployment of the mesh network. With the mesh network in place, news items advertising products would be disseminated through the mesh network and be picked up by the mobile devices that costumers carry.

News entries have a limited lifetime. After this time period expires, the information they carry is no longer valuable to clients and should be flushed from the network. Going back to our example, the lifetime of entries could relate to the time period when a sale is effective (for example, drink at a discount price during lunch time).

At any point in time, a router will have a partial view of the complete set of news items in its cache. We do not expect each router to store all items. Instead, each router will devote a fixed amount of memory to store entries it discovers through communication with other routers. Periodically, this view will be refreshed with different news entries. According to the interests that costumers have expressed when contacting a router, their mobile clients will be updated with relevant advertisements.

```
/*** Active thread ***/
// Runs periodically every T time units

Q = selectPeer()
buff_send = selectItemsToSend()
send buff_send to Q
receive buff_recv from Q
cache = selectItemsToKeep()
```

(a)

```
/*** Passive thread ***/
// Runs when contacted by another router

receive buff_recv from any P
buff_send = selectItemsToSend()
send buff_send to P
cache = selectItemsToKeep()
```

(b)

Fig. 1. Skeleton of an epidemic protocol.

III. Shuffle Protocol

When a router participates in a gossip exchange, it assumes either an *active* or a *passive* role. Each router initiates an exchange once per round. We refer to the router that initiates the exchange as the active one, while the one that is contacted assumes the passive role.

The data exchange between routers follows a predefined structure. Figure 1 shows the skeleton of the push-pull epidemic protocol we use for communication within the mesh backbone. Three methods, `selectPeer()`, `selectItemsToSend()` and `selectItemsToKeep()` represent the core of the protocol. By implementing different policies in these methods, various epidemic protocols, each with its own distinctive characteristics, can be instantiated.

Based on the structure shown in Figure 1, we introduce an epidemic protocol we call *shuffle*. The shuffle protocol is characterized by avoiding the loss of data during an exchange. It achieves this by establishing an agreement between peers that each peer will keep the entries received from the other after the exchange takes place. We will elaborate on the details of the exchange later on.

The shuffle protocol is partly based on a peer-to-peer protocol used for handling flash crowds [5], which we recently enhanced in order to maintain unstructured overlays that share important properties with random graphs [6]. The most important observation to make is that any two nodes that engage in a shuffle essentially *swap* a number of entries. In doing so, they not only preserve the data that are collectively stored in the network, but also "move" these data around in a seemingly random fashion. The underlying idea is that by randomly shuffling data entries between nodes, all nodes will be able to see all news items eventually.

A. Protocol Policies

In the shuffle protocol, each node agrees to keep the entries received from a neighbor for the next round. This might seem trivial, but given the limited storage space available in each node, keeping the entries received during an exchange implies discarding some entries that the node has in its cache. By picking the entries to be discarded from the ones that have been sent to the neighbor, we ensure the conservation of data in the network. The policies are summarized as follows:

Method	Description
`selectPeer()`	Select a neighbor randomly
`selectItemsToSend()`	Randomly select s entries from the local cache. Send a copy of those entries to the selected peer.
`selectItemsToKeep()`	Add received entries to the local cache. Remove repeated items. If the number of entries exceeds c, remove entries among the ones that were previously sent until the cache contains c entries.

B. Simulation Setup

In order to observe the behavior of the protocol in large-scale settings, a series of simulations were conducted. We have learned from earlier studies of other epidemic protocols [7] that the results from emulations running in a cluster of hundreds of nodes yield strikingly similar results to simulation results when observing large-scale behavior. For this reason, we decided to study the behavior of the protocol presented in this paper through extensive simulations. The results presented in this section correspond to a network of 10000 nodes with a cache size of c, which may vary in different experiments. Two types of topologies were used in the experiments:

- *Grid topology* The nodes were set up in a square grid topology, with 100 nodes on each side over an area of 100×100 units. Two cases were explored: (a) the range of each node was set to 1 unit, making communication possible with the node's immediate neighbors to the North, South, East and West. On average, each node had 3.96 neighbors (due to the effect of boundary nodes with less than 4 neighbors); (b) the range of each node was set to 2 units, making communication with 12 immediate neighbors.
- *Random topology* The nodes were placed randomly in a square area of 100×100 units. Nodes were allowed to reach neighbors within a range of 2 units, which was enough to guarantee that each node had at least one neighbor and that a path between any two nodes existed. The average number of neighbors for each node was 12.19.

Both topologies were used to study the behavior of the protocols. The experiments that we conducted focused on two characteristics observed during the execution of each epidemic

protocol (a) the replication of items in the network and (b) the time required to reach all the nodes in the network.

C. Properties

To understand the behavior of the protocol, we focus on the way a single news entry traverses the network. On first instance, a news entry is inserted into the network by a router. Subsequently, the entry takes a step (moves to the cache of another router) whenever the router that currently holds the entry participates in an exchange. For every execution of the protocol, the next step of the entry is chosen randomly. As a consequence, the path followed by an individual entry consists of a series of random steps. This behavior is analogous to a random walk in the space defined by the mesh network.

Additionally, as an entry moves from router to router, there is a chance that it will be replicated in the caches of the routers it has passed through, given that there was space available. It follows that a news item may have several news entries in the network at the same time. For that reason, when referring to an item in the network we are actually referring to all news entries that represent that news item. These entries have the same id. In the next sections we study the way these entries are replicated through the network.

1) Distribution of Storage Capacity: Let us first consider how different news items are distributed through the network. After running the protocol for several rounds, we observe that the storage capacity of the network is evenly divided between the different items. By this, we mean that the slots available to store news entries are used in a balanced way, with each news item being able to place approximately the same number of entries in the network. This behavior is not programmed into the algorithm, but it is an emerging property resulting from its repeated execution.

The value to which the number of entries of an item converges is dictated by the number of different news items in the network. Given a network of size N where all nodes have a cache size of c, the network has a total capacity of $N \times c$. These $N \times c$ available slots have to be filled with d different news items. Because of the randomness introduced when choosing which entries to exchange, the total capacity should eventually be evenly divided between the different items resulting in an average of $\frac{N \times c}{d}$ entries for each of the d news items. Considering that the protocol does not allow more than one news entry representing the same news item in the same cache, this means that c/d of the nodes should have an item of each of the d different ids:

$$\#\text{ entries per item} = \frac{\text{capacity of the network}}{\text{number of news items}} = \frac{N \times c}{d}$$

Figure 2 shows the convergent behavior of the protocol. For the experiment, a collection of 10000 nodes were placed in a grid topology with 4 neighbors per node and 10 nodes were randomly selected to generate different news items. Time is measured in *rounds*, where a round is a gossiping interval in which each node executes the exchange protocol once. After an initial stabilization period, the number of entries in the system for each of the 10 items converges to the same value.

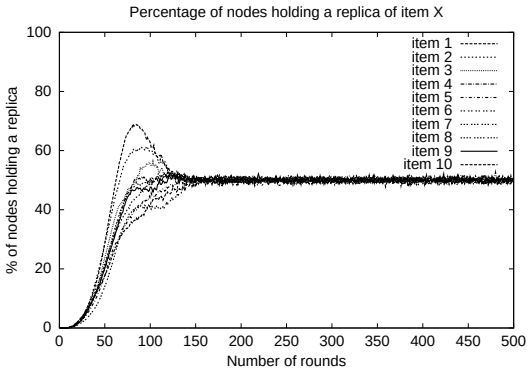

Fig. 2. Convergent behavior illustrated by having 10 nodes that generate news items in a network of 10000 nodes.

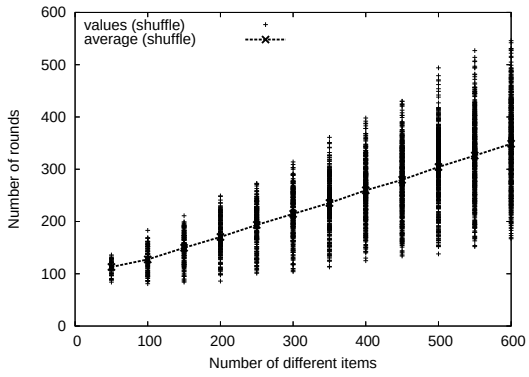

Fig. 3. Number of rounds required for all the routers in the backbone to have seen an item using shuffling on a grid topology.

According to our previous reasoning, this value should be $10000 \times 5/10 = 5000$, meaning that 50% of the nodes in the network have an entry from one of the 10 different news items available, which is confirmed by our experiments. Similar convergent behavior was observed when experimenting with other topologies.

2) Dissemination Speed as a Function of the Diversity of News Items: To demonstrate the effectiveness of shuffling for disseminating information, we have conducted experiments that show the effect of the number of different items on the dissemination speed of the items through the network. In this section, we look at the time needed for the news items to have reached all routers in the network. The results presented here correspond to a mesh backbone of 10000 routers. Unless explicitly stated, the routers were set up in a rectangular grid topology, with 100 routers on each side. For the experiments, we measure the time it takes for the items to reach all the routers in the network.

Figure 3 shows the time, measured in rounds, required for various different items to have passed through the caches of all the routers in the backbone. The cache size for all routers was set to 50 and all items in the cache were shuffled in each round. In each experiment, a different number of distinct items (starting at 50 and up to 600, with increments of 50) were inserted into the backbone by routers located in

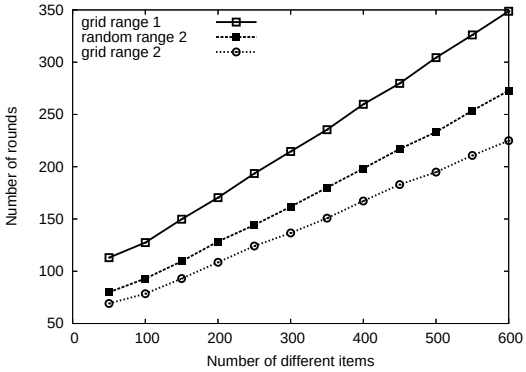

Fig. 4. Number of rounds required for all the routers in the backbone to have seen an item using shuffling on three different topologies.

random locations. For each news item, the time required to traverse the backbone was measured. Due to randomness in the exchanges and the location of the routers inserting the news items, the time measured for an individual news item may vary significantly compared to the measurements for other items. By calculating the average time for a news item to go through the mesh backbone, we can observe that as the diversity of news items in the network increases the average time for a specific news item to reach all routers increases linearly. We observe that this linear behavior is maintained when conducting the experiments with different topologies, as shown in Figure 4.

In our third set of experiments, we focus on the effect of the cache size on the dissemination speed. As before, we look at the average time required for an item to have reached all routers in relation to the number of different items being gossiped. The results, shown in Figure 5, reveal that the slope of the curve of average values is directly related to the number of items being shuffled. There is an inversely proportional relationship between the number of items being exchanged and the slope of the curve. The four curves shown correspond to experiments with a cache size of 30, 40, 50 and 60 items. In all cases, all entries in the cache were exchanged. By doubling the number of entries shuffled from 30 to 60, the average time for news items to pass through every router in the backbone is virtually divided in half. Such a characteristic, as well as the predictable behavior with an increasing number of different items, are important factors to consider when choosing an appropriate value for the cache size c and the number of entries to shuffle.

3) Robustness: In order to test the robustness of our system in the case of node failures, we look at a scenario where the nodes within a limited area go down, not unlike what would happen in case of a power outage. The experiment, performed with 10000 routers arranged in a grid with range 1, consists of observing the number of entries per item in the mesh backbone before, during and after the failure of all routers within a square area. We assume that when a router fails, all the entries in its cache are lost. When the router goes up again, its cache is empty and has to be populated again.

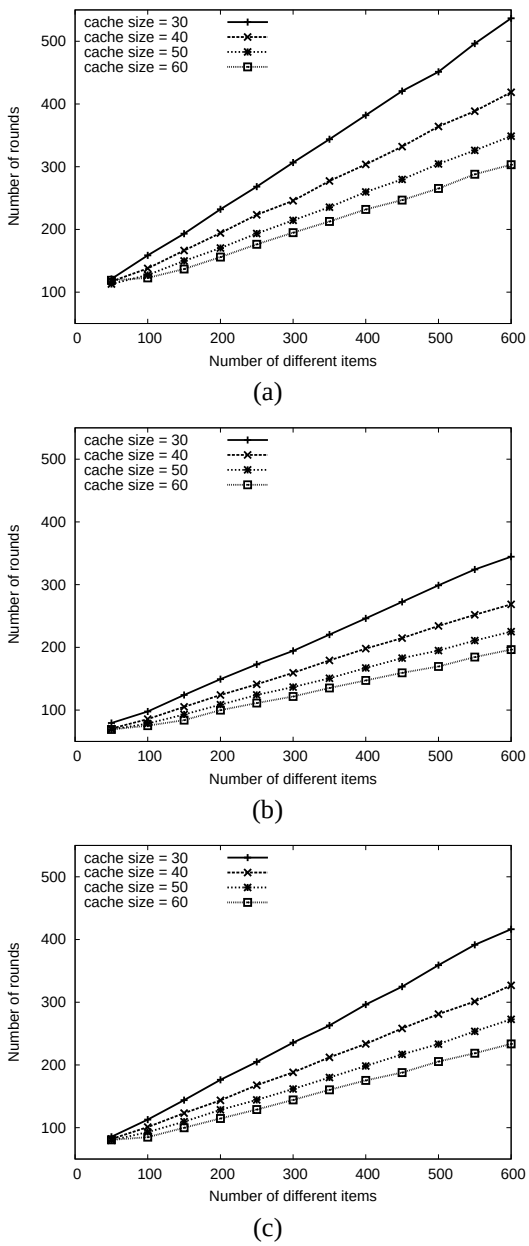

Fig. 5. Number of rounds required for all the nodes in the network to have seen an item for different cache sizes with (a) grid topology (range 1), (b) grid topology (range 2) and (c) random topology (range 2). All entries in the cache are exchanged.

Figure 6 shows the results of the experiment when 49% of the routers experience a failure at the same time and recover 100 rounds later. The routers chosen for failure were arranged in a 70×70 square inside the 100×100 grid. Like the experiment in Figure 2, 10 items are being shuffled in the network and all routers have a cache size of 5. Once the number of entries per news item has converge to the same value, the routers chosen for failure go down. As could be expected, given that the entries were randomly located throughout the network, the number of entries per item is virtually cut by half once the failures occur. When the routers that failed rejoin

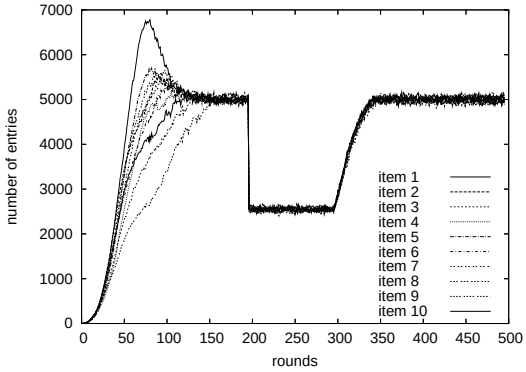

Fig. 6. Number of entries per news item. 49% of the routers go down at round 200 and recover at round 300.

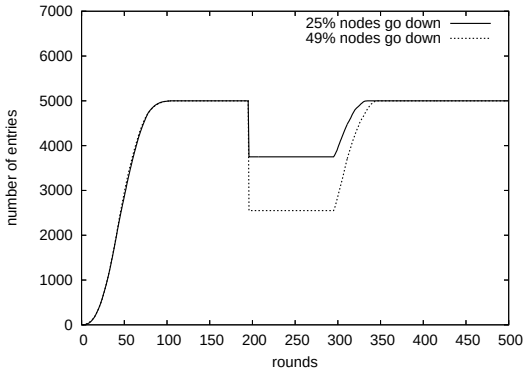

Fig. 7. Average number of entries per item. Routers go down at round 200 and recover at round 300.

the network, we observe a smooth transition to the previous state, with entries quickly populating their caches. Unlike the first rounds, the entries per item are replicated at roughly the same rate. This is due to the fact that the routers surrounding the area of failure already have their caches full of entries and can update the routers that failed as soon as they become operational again.

For a clear view of the recovery time, Figure 7 shows the average number of entries per item. In addition to the experiment presented earlier, we include the case where 25% of the routers fail. These routers are arranged in a 50×50 square. Comparing both curves, we observe the same behavior up to the moment of node failures. At that point, the average number of entries per items falls according to the loss of storage space. The recovery in both cases is quick despite the difference in the number of routers that failed.

The speedy recovery of the affected area can be attributed to information flowing in from multiple sides. For an affected square area, we would expect the recovery time to be proportional to the square root of n, where n is the number of routers that experience a failure. As can be seen in Figure 8, this seems to be the case. The results shown in the figure were obtained using the random topology with range 2. Several experiments where routers within a square area failed were conducted. For each experiment, a square area of a different

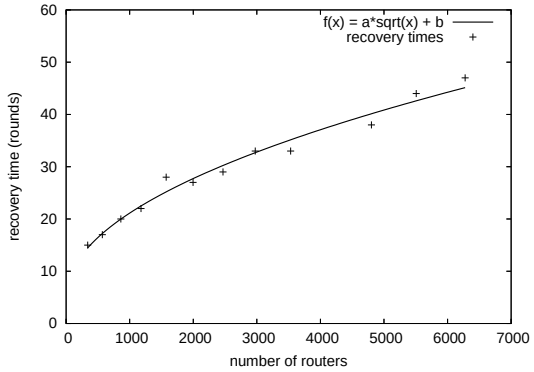

Fig. 8. Recovery times when an increasing number of routers fail.

size was used, ranging from 20×20 up to 80×80 with increments of 5 distance units on each side. Figure 8 shows that, indeed, the recovery times obtained from the simulations tend to be proportional to the square root of the number of routers affected. To verify this, we also plot the curve a*sqrt(n)+b which was obtained through linear regression. The constants have values 0.504244, and 5.18383, respectively.

IV. Performance on the Client Side

In this section, we take a look at the performance of the news service from the user's perspective. Users of the service have access to news items through *clients* in the mesh network. These include portable devices such as laptops, PDAs or other hand-held devices. Due to the variety of news items available in the network, users need to configure their clients to retrieve items matching the users interests. Item retrieval is based on content filtering. Once a connection to a router is established, clients must submit filtering criteria for the router to identify which news items to forward to the client.

A. Recall Rate

To evaluate the effectiveness of the news service from the user's perspective, we observe the *recall rate* of items over time. The recall rate is defined as the number of items of interest to a user that a router delivers to the client device over a period of time versus the total number of relevant items. We test the recall rate through the following experiment:

- A network of 2500 routers is arranged in a 50×50 grid. Each router can communicate with its neighbors to the North, South, East and West.
- 50 users are positioned at random locations.
- 500 news items are being shuffled in the network.
- The interests of the users match 100 news items.

The news items are published at random locations in the network and shuffled until the number of entries for each item converges to roughly the same value (as seen in Section III-C.1). At that point, the clients connect to the nearest router expressing interest in certain kinds of items. A router responds by forwarding the matching news items seen in its cache to the client. Caches are updated with every gossip round prompting the delivery of previously undiscovered items to the client.

As a result, we expect the recall rate to increase as the client spends more time connected to a router.

The results of repeating the experiment with different cache sizes can be seen in Figure 9. The figure shows the average recall rates for the 50 users. In all cases, the recall rates increase rapidly during the initial rounds and slow down when most items have already been discovered. As could be expected, larger caches lead to higher recall rates of items. This is due to a higher storage capacity in the network that allows for more entries to be placed for each news item. Therefore, the probability of finding a particular item in the cache of a router increases.

It should be noted that an increase in the total number of news items would slow down the recall rate, as dissemination speed decreases with the number of news items in the network. This effect can be countered by an increase in cache size. In the remainder of this section, we take a fixed number (500) of news items and explore the effect of modifying other parameters, such as the cache size, the number of items shuffled and the topology of the network, on the recall rate.

B. Probability of seeing an item

Executing the shuffle protocol until the storage capacity of the network is full yields a probability of c/d of finding a particular item when examining the cache of a router picked at random, for $c \leq d$. If we define the success of our experiment as finding a particular item in a random cache and knowing that the probability of success remains constant, the probability of succeeding after performing the experiment $k \geq 1$ times is:

$$p(k, c, d) = 1 - \prod_{i=1}^{k} (1 - prob\_success(i)) = 1 - \left(1 - \frac{c}{d}\right)^k$$

Figure 10 shows the probability of finding an item in a cache selected at random after k attempts for different cache sizes. We observe a similar, although not identical, behavior to the recall rate results presented in the previous section. This is not surprising, as the shuffle protocol ensures that after each round a router refills its cache with entries received from a neighbor chosen at random. However, due to the locality of the gossip exchanges, when looking at the cache of the same router for several rounds, we are bound to discover the items held by our neighbors first. This limits the variety of items we might see as our neighbors are more likely to hold many of the same items as we do in comparison to a randomly chosen router in the network. This accounts for the slightly lower recall rate in the experimental results compared to the probability of seeing an item when selecting a random cache every time.

C. Improving Recall Rate

Shuffling provides a random sample of the collection of items in the network at every round for each router, however, as can be inferred from Figure 9 and 10, there is a correlation in the items seen from one round to the next, which accounts for less than optimal recall rates. In other words, the reason for the recall rate results not being identical to the probability in Figure 10 is due to the results from each round not being independent. This can be attributed to the lack of variety of items in the neighborhood of a router.

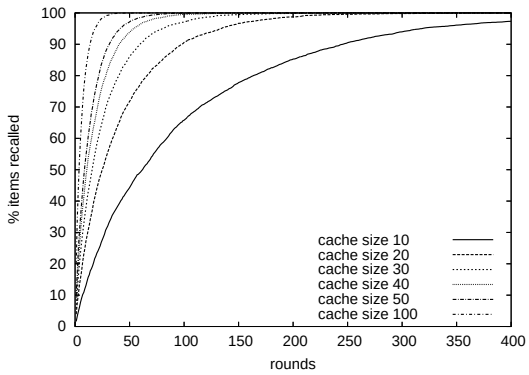

Fig. 9. Recall rate of news items. All entries in the cache are exchanged.

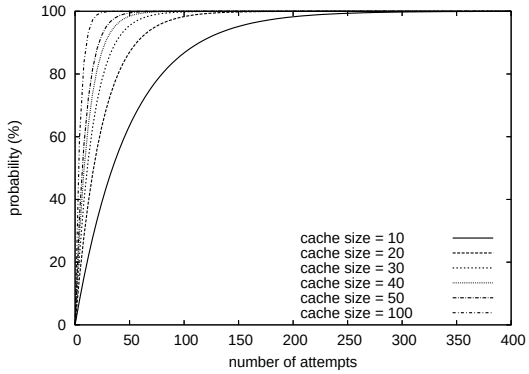

Fig. 10. Probability of recalling an item from a randomly filled cache.

Figure 11 shows the effect of the neighborhood in the recall rate. We confirm that the recall rate was being hampered by each router having a limited neighborhood by showing that the probability of seeing an item when a cache is picked randomly is the same as the recall rate when the range of a router is such that it can reach any other router in the network. In the graph, the results for a network of routers with range 100 and $p(k, 50, 500)$ overlap. We also show the impact in performance of doubling the range from 1 to 2 units, effectively increasing the number of neighbors from 4 to 12. This experiment shows that it is not necessary to be able to reach every node in the network to achieve a close-to-optimal recall rate. Finally, as a worse case scenario, we show what happens if the routers are arranged in a single line, where each router can only reach its neighbors to the left and right. In this case, the recall of items after the first few rounds becomes increasingly slow. We attribute this to new items being hard to come by after the items of interest in the immediate neighborhood have been discovered. Having only two neighbors, the likelihood of new items reaching the neighborhood is reduced, requiring more iterations of the protocol to update a cache with different items. Obviously, this topology is not realistic and should be avoided.

Another option for improving the recall rate without increasing the amount of entries exchanged per round is to

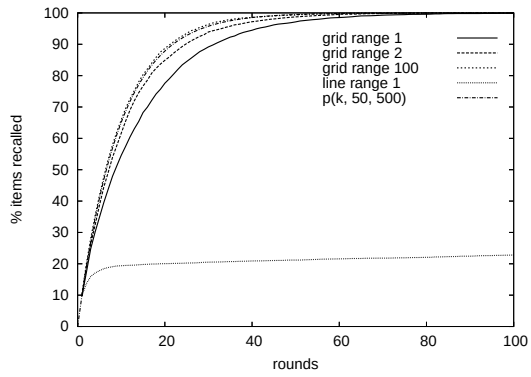

Fig. 11. Recall rate of news items.

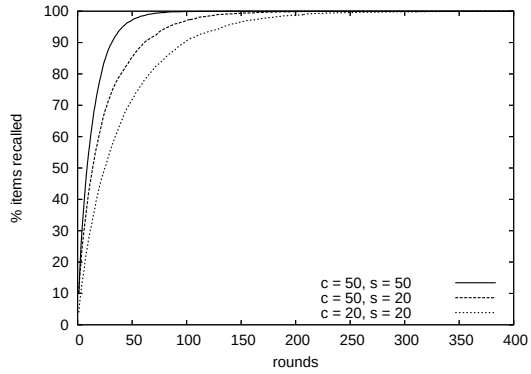

Fig. 12. Recall rate of news items.

increase the cache size. The results presented previously assumed that all entries in the cache were exchanged in each round. Figure 12 shows how increasing the cache size while exchanging the same amount of entries provides an initial boost in the recall rate. However, since the speed at which news items move through the network depends on the number of entries exchanged per round, having a bigger cache does not provide any benefits for finding the last few items that were not originally in the vicinity of the router.

V. Related Work

From a functional point of view, the news service we propose has some similarity with content-based distributed publish/subscribe systems [8], [9], [10], [11]. With the increase in popularity of wireless technology, some publish/subscribe systems have been extended to support for mobile, wireless clients [12]. For the most part, the systems proposed use a single tree-shaped overlay to interconnect a set of brokers which cooperate to deliver the events published to the appropriate subscribers. This approach, while efficient under static conditions, might face robustness and scalability issues in highly dynamic environments, such as wireless networks with mobile users. Efforts in maintaining a tree overlay under frequent topology changes aim at dealing with these issues [13]. However, depending on how frequently the changes occur, maintaining a tree may introduce additional overhead and complexity. Our approach offers robustness and scalability at the cost of periodic communication for gossiping.

Content-based publish/subscribe projects aimed explicitly at wireless, mobile environments [14], [15], [16] are more closely related to our work. In particular, systems that rely on probabilistic techniques for the delivery of events. In particular, [17] combines deterministic routing with probabilistic techniques to increase the resilience when faced with topology changes.

Our work also relates, in a way, to efforts in distributed storage [18], [19]. Like our news service, these systems rely on data redundancy to ensure robustness when node failures occur. However, while most of these systems carefully place replicas based on the reliability of nodes, we replicate items and relocate them in a random fashion. We do, nevertheless, manage to use the storage capacity in a fair manner, dynamically adjusting the number of replicas of an item according to the number of items in the network.

VI. Discussion

As mentioned in the introduction, one of the main advantages of having a decentralized system like the one we propose is avoiding single ownership of the service. Single ownership implies that one entity is fully responsible for the availability and quality of the service. This may not be a bad thing from the point of view of managing the system, however it restricts others from contributing to the service even when resources are available. One of the strenghts of our news service is the flexibility it allows for routers to have some control over the quality of service they offer. As explained in Section IV, the quality of the service as perceived by the users can be improved by increasing the wireless range or the amount of memory allocated for storing entries. Decisions to do so can then be taken on an individual basis by the administrators of each router.

Taking a more active approach for the recall of items is also a possible way of improving the perceived performance from the users point of view. As explained, clients connect to a nearby router and retrieve news items matching the user's interests. While the matching news items from the router's cache will be made available to the client immediately, discovering the totality of relevant news items may take several rounds and, depending on the time period between rounds, this delay might inconvenience some users. Instead of passively waiting for the news items to arrive, a router may decide to forward the user's filter to other routers, thus increasing the chances of discovering relevant news items. For example, we can calculate that for $c/d = 0.2$ it would take at least 10 rounds to retrieve approximately 90% of all items. In this case, by forwarding the filter to 4 other routers, the client could receive almost all news items in 2 rounds.

The flexibility of being able to independently decide on the amount of resources to invest in the news service coupled with the minimal requirements to participate opens up the possibility of deploying the service on a large scale using heterogeneous nodes. Deploying the service over large geographical areas, such as a campus or a city, may require

some considerations in the dissemination of news items. As mentioned before, we impose a time limit for the validity of the entries in the network. However, when disseminating the entries over a large area, it may also be necessary to establish geographic constraints. By adding location-awareness to the shuffling of entries, news items could be dispersed over limited areas. For example, an entry may be forwarded only within a radius from the location where it originated. As a result of restricting the area over which an entry can travel, space which would otherwise be taken by these entries is freed increasing the number of different items that the network can hold.

Another consideration to keep in mind is security. A gossip-based system like the one we propose is specially susceptible to denial-of-service attacks. We can imagine a scenario where a malicious node generates bogus news items and inserts them into the network at a high rate. Having large numbers of items at the same time slows down the dissemination speed, as there are less entries per item in the network. Without any security mechanisms in place, a single node could virtually bring the service to a halt. It is clear that some kind of regulation regarding who can publish news items is necessary.

VII. Conclusions and Future Work

In this paper, we presented a highly robust distributed news service suitable for wireless mesh networks. We have shown that by using an epidemic protocol at the core of our service, we can provide an efficient and scalable solution for delivering news items while, at the same time, offering the participating routers the flexibility of managing their own resources to better suit their clients needs. Through the use of simulations, we analyzed the effectiveness and robustness of the dissemination of items through the mesh backbone. We corroborated the effectiveness of the service by taking the user's perspective and providing an analysis of the quality of the service in terms of the delivery of relevant news items to the client devices.

Regarding future work, in addition to the issues already addressed in the discussion, we expect to explore other types of services that could be deployed on top of these networks. Even though large deployment of mesh networks is not yet a common ocurrence, we intend to focus on developing distributed solutions for large-scale networks with the belief that a collaborative effort can yield efficient results under the sometimes unpredictable conditions of wireless networks.

References

[1] I. F. Akyildiz, X. Wang, and W. Wang, "Wireless mesh networks: A survey," *Computer Networks Journal (Elsevier)*, March 2005.

[2] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry, "Epidemic algorithms for replicated database maintenance," in *Proceedings of the 6th Annual ACM Symposium on Principles of Distributed Computing (PODC '87)*, ACM, Ed., Vancouver, Canada, August 1987, pp. 1–12.

[3] K. P. Birman, "The surprising power of epidemic communication," in *Future Directions in Distributed Computing: Research and Position Papers*, January 2003, vol. 2584/2003, pp. 97–102.

[4] M. Jelasity, R. Guerraoui, A.-M. Kermarrec, and M. van Steen., "The Peer Sampling Service: Experimental Evaluation of Unstructured Gossip-Based Implementations," in *Middleware 2004*, ser. Lecture Notes on Computer Science, vol. 3231, ACM/IFIP/USENIX. Berlin: Springer-Verlag, Oct. 2004, pp. 79–98.

[5] A. Stavrou, D. Rubenstein, and S. Sahu, "A Lightweight, Robust P2P System to Handle Flash Crowds," *IEEE Journal on Selected Areas in Communication*, vol. 22, no. 1, pp. 6–17, Jan. 2004.

[6] S. Voulgaris, D. Gavidia, and M. van Steen., "Inexpensive Membership Management for Unstructured P2P Overlays," *Journal of Network and Systems Management*, vol. 13, no. 2, pp. 197–217, June 2005.

[7] S. Voulgaris and M. van Steen, "An Epidemic Protocol for Managing Routing Tables in very large Peer-to-Peer Networks," in *Proc. 14th IFIP/IEEE Workshop on Distributed Systems: Operations and Management (DSOM 2003)*, Oct. 2003, pp. 41–54.

[8] A. Carzaniga, D. Rosenblum, and A. Wolf, "Design and evaluation of a wide-area event notification service," *ACM Trans. Comput. Syst.*, vol. 19, no. 3, pp. 332–383, 2001.

[9] G. Cugola, E. D. Nitto, and A. Fuggetta, "The JEDI event-based infrastructure and its application to the development of the opss wfms," *IEEE Trans. Softw. Eng.*, vol. 27, no. 9, pp. 827–850, 2001.

[10] P. R. Pietzuch and J. Bacon, "Hermes: A distributed event-based middleware architecture," in *ICDCSW '02: Proceedings of the 22nd International Conference on Distributed Computing Systems*. Washington, DC, USA: IEEE Computer Society, 2002, pp. 611–618.

[11] B. Segall and D. Arnold, "Elvin has left the building: A publish/subscribe notification service with quenching," in *Proceedings of the Australian UNIX and Open Systems User Group Conference (AUUG'97)*, September 1997, pp. 243–255.

[12] M. Caporuscio, A. Carzaniga, and A. L. Wolf, "Design and evaluation of a support service for mobile, wireless publish/subscribe applications," *IEEE Transactions on Software Engineering*, vol. 29, no. 12, pp. 1059–1071, Dec. 2003. [Online]. Available: http://serl.cs.colorado.edu/ carzanig/papers/

[13] G. Picco, G. Cugola, and A. Murphy, "Efficient content-based event dispatching in the presence of topological reconfigurations," in *Proc. of the 23 Int. Conf. on Distributed Computing Systems (ICDCS 2003)*, 2003. [Online]. Available: citeseer.ist.psu.edu/picco03efficient.html

[14] G. Cugola, A. L. Murphy, and G. P. Picco, "Content-based Publish-subscribe in a Mobile Environment," in *Mobile Middleware*, P. Bellavista and A. Corradi, Eds. CRC Press, 2005, invited contribution. To appear.

[15] Y. Huang and H. Garcia-Molina, "Publish/subscribe tree construction in wireless ad-hoc networks," in *MDM '03: Proceedings of the 4th International Conference on Mobile Data Management*. London, UK: Springer-Verlag, 2003, pp. 122–140.

[16] R. Meier and V. Cahill, "STEAM: Event-based middleware for wireless ad hoc networks," in *Proceedings of the 1st International Workshop on Distributed Event-Based Systems (DEBS '02)*, Vienna, Austria, July 2002.

[17] P. Costa and G. P. Picco, "Semi-probabilistic content-based publish-subscribe," in *ICDCS '05: Proceedings of the 25th IEEE International Conference on Distributed Computing Systems (ICDCS'05)*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 575–585.

[18] A. Haeberlen, A. Mislove, and P. Druschel, "Glacier: Highly durable, decentralized storage despite massive correlated failuresglacier: Highly durable, decentralized storage despite massive correlated failures," in *Proceedings of the 2ndt USENIX Symposium on Networked Systems Design and Implementation (NSDI '05)*, Boston, Massachusetts, May 2005.

[19] A. Adya, W. J. Bolosky, M. Castro, G. Cermak, R. Chaiken, J. R. Douceur, J. Howell, J. R. Lorch, M. Theimer, and R. P. Wattenhofer, "Farsite: federated, available, and reliable storage for an incompletely trusted environment," *SIGOPS Oper. Syst. Rev.*, vol. 36, no. SI, pp. 1–14, 2002.