A Lightweight Content Replication Scheme for Mobile Ad Hoc Environments

Vineet Thanedar, Kevin C. Almeroth, and Elizabeth M. Belding-Royer

Department of Computer Science
University of California, Santa Barbara
Santa Barbara, CA, USA 93106
{vineet, almeroth, ebelding}@cs.ucsb.edu

Abstract. The mobile, wireless, and self-organizing features of ad hoc networks pose many challenges with respect to continuous availability and accessibility of data. In such a dynamic environment, there are many advantages in replicating a data item so there are multiple copies, including reduced response times and higher data availability. Also, if done efficiently, replication can help reduce energy usage. In this paper, we propose the Expanding Ring replication strategy for pull-based information dissemination environments. One of our primary objectives is the development of a lightweight scheme for mobile nodes. We evaluate the performance of our scheme with respect to a number of parameters and compare it to a system without replication. Our results show a reduction in the average response times and the message processing overhead on nodes. The scheme also does well when both, the overall willingness of nodes to cache data and their individual caching capabilities vary.

1 Introduction

There has been considerable recent interest in the area of information dissemination in a wireless mobile environment. These environments are characterized by the presence of an information server that, through a public wireless access point, delivers useful information to a population of interested mobile clients. This information can be delivered in two ways: 1) a "push-based" approach where the server periodically broadcasts data items and client nodes access this data by monitoring the broadcast channel and waiting until the next broadcast of the data item, and 2) "pull-based" access methods where mobile nodes query the server for data they need. Since both push-based and pull-based systems have their own pros and cons, an efficient, integrated push-pull environment may be a better solution. Such systems have been explored previously [2], [7].

Previous work has addressed pieces of the problem but never in the context of or for the applications we consider. Research efforts in the past have mainly focused on push-based information delivery in a wireless mobile environment. The "Broadcast Disks" approach by Acharya et. al. [1], on-line scheduling algorithms by Vaidya and Hameed [8], [12], and caching strategies proposed by

N. Mitrou et al. (Eds.): NETWORKING 2004, LNCS 3042, pp. 125–136, 2004.

Hara [9] explore push-based systems in detail. Little work has been done in pull-based mobile environments. Also, the focus has mainly been on the scheduling of data items to broadcast in pull-based broadcast systems. Some of the work in pull-based systems include Aksoy et. al.'s scheduling algorithm for large scale on-demand broadcasting [3], [4], and Karakaya and Ulusoy's [11] scheduling algorithm based on an approximate version of the *Longest Wait First* heuristic. Replication schemes for ad hoc networks have been proposed before by Hara [10], and Chen and Nahrstedt [6]. However, these schemes are mainly directed towards replication within a group of mobile hosts. Replication of the requested data items in a pull-based environment has not been significantly explored.

In this paper, we propose the Expanding Ring replication scheme to replicate content in a mobile ad hoc network in the presence of a pull-based information delivery environment. It is worthwhile to mention that although our method is pull-based, the replication strategy is orthogonal to a system where a server is periodically broadcasting content. Hence, our solution can be deployed alongside a push-based system. Given the energy constraints in a mobile ad hoc network, our focus has been on developing a scheme that reduces the burden on a mobile node. Our proposed strategy is elegant and lightweight. The decision to replicate data is handled entirely by the server, thus relieving the nodes of this overhead. In our scheme, the server replicates *in-demand* data in a unique fashion that increases the likelihood of a node finding the required content within its neighborhood. The analysis and simulation results show that our scheme considerably reduces the energy spent by nodes in terms of the messages processed.

The remainder of the paper is organized as follows. In Section 2 we discuss the motivation behind our scheme. The system design and the replication strategy are discussed in Section 3. We present the simulation results of our scheme in Section 4 and concluding remarks and directions for future work in Section 5.

2 Motivation

The motivation behind our work is developing a lightweight replication scheme for a class of application environments that, we believe, will play a major role in the future. These are environments that consist of a server delivering publicly accessible and useful information. A few examples of these include:

- **Public Transportation Areas**: We are already seeing access points being deployed at airports and subway stations that serve information such as up-to-date schedules, last-minute ticket deals, hotel reservations, and car rental information. In addition, because travellers spend a significant amount of time waiting at such places, headline news updates, entertainment news, or similar data may be in high demand.
- **Large Events**: In the future, large sporting events such as the Olympic games will see the deployment of access points that deliver information such as up-to-date games schedules, real-time scores, medals tally, interesting trivia about the games and so on. Visitors carrying mobile devices can then request this information from the server.

We now present a formal analysis of a scheme with no replication, i.e., where each request travels to the server. The motivation for our scheme is the poor performance of this system with respect to the overhead on nodes.

2.1 Analysis of Zero Replication

In this section, we analyze a purely pull-based system with no replication of content in the network. We intend to show that such a model is not very efficient and can be improved by replicating content in the network. We also show that even when content is replicated in the network so that a fraction of content requests are satisfied by other nodes multiple hops away, there is considerable energy overhead on nodes in the network.

Say n items of data, $d_1, d_2, ..., d_n$ are being served by the information server. Consider a m-node one-hop network that generates M requests, $k_1, k_2, ..., k_m$ ($M = \sum_{j=1}^{m} k_j$) for content d_i in time period T. Here k_j is the number of requests generated by node j for d_i in time T. Say the server is an average of H hops away from this one-hop network. The requests then have to travel H hops to be fulfilled. This implies that an average of $2 * H * M$ *extra* messages will be processed by the intermediate nodes for all the requests to be satisfied. Thus, the energy consumption overhead, O_{norep}, on intermediate nodes for a single data item in terms of the additional messages processed per unit time, per hop of requests generated is:

$$O_{norep} = MP_{add} = \frac{2 \times H \times M}{T} \tag{1}$$

As an example, let $H = 3$, $M = 25$ generated by $m = 6$ nodes in the one-hop network, and $T = 1$ min. Substituting these values in the above equation we can see that 150 extra messages are processed in the intermediate hops for 25 content requests, i.e., 6 times more messages are processed in the network for requests generated just 3 hops away from the server. When applied to all the requests generated in the entire network for all the data items, we can see the large amount of energy consumed (in terms of the messages processed) in processing *extra* messages.

In the above analysis, we looked at the case where each request travels to the central server. This is an extreme option. A better scheme would be where some of the requests are satisfied by intermediate nodes rather than the requests travelling all the way to the server. Say M' out of M messages are satisfied by intermediate nodes and say they take an average of H' hops to find the cached data. In such a scenario, $2 * H * (M - M')$ messages are processed by intermediate nodes for requests going all the way to the server and $2 * H' * M'$ messages are processed by intermediate nodes for all other requests. The energy consumption overhead, $O_{partialrep}$, now in terms of additional messages processed per unit time, per hop is:

$$O_{partialrep} = MP'_{add} = \frac{2 \times H' \times M' + 2 \times H \times (M - M')}{T} \tag{2}$$

Substituting the same set of values as before with $H' = 1$, we see that the number of additional messages processed are 90, or almost 4 times the number of content requests generated. We can see that even if all the requests did not travel to the server, we could still have a significant message processing overhead on the intermediate nodes. This can discourage node participation in the network.

Our goal in this paper is to present our replication strategy, which replicates data in a manner such that requests are satisfied within a node's one-hop network. This considerably reduces the energy spent by nodes in processing content requests and their responses for other nodes.

3 Expanding Ring Replication Scheme Design

In this section, we first describe the core for our replication scheme. The data accessibility mechanism used to access information in the network is then discussed. Finally, our proposed replication scheme is described in Section 3.3.

3.1 Data Agent

Our scheme involves the presence of an agent on each mobile device, called the *Data Agent*. The data agent performs tasks related to the exchange of data between mobile devices. These include support for the replication scheme, management of data stored on the device, and resource monitoring to determine whether the device can participate in the replication mechanism. We term the information delivered by the server as *remote* data items. The data agent marks these items acquired from the server as one of the following two types: Shared Remote and Non-shared Remote data items. Shared remote data items are remote data items that are made available for sharing with other nodes. The node may have acquired these for its own purposes or may hold them as a result of replication. A node services requests for any shared remote data items. If the node lacks the resources to fulfill requests, the agent marks them as non-shared.

3.2 Data Accessibility

In our scheme, each node advertises to its neighbors only the kind of information it needs or in which it is interested. These advertisements, called *interest advertisements*, essentially contain a description of the information required. Once an interest advertisement is created, it is broadcast to all nodes within a distance of one hop. Neighboring nodes do not further broadcast the interest advertisement. This is to prevent a flood of broadcast requests in the network as well as a potential flood of responses. If any of the neighboring nodes has a copy of the requested information, it unicasts the data item back to the advertising node. The data agent on the advertising node sets a timeout period referred to as the *ad response time* for the request to be satisfied by its neighbors. If the request is not satisfied within this time period, the node initiates an information request to the server with the original version of the requested information.

One obvious question that could arise is why the advertisement is not forwarded to nodes beyond the one hop neighborhood. The rationale behind this is to reduce the energy expended by nodes, as well as the network traffic load. Since the request is not forwarded beyond the node's neighborhood, considerable energy is saved by other nodes in the network since they do not have to expend their battery power processing interest advertisements for nodes that may be several hops away. For example, consider a network of 3 hops. If each of these hops has about 5 nodes, then an interest advertisement, when forwarded to nodes beyond one hop, is processed by 12 nodes. On the other hand, only 6 nodes (4 neighbors + 2 additional nodes) will process the advertisement if it is broadcast only within the neighborhood and if not found, requested from the server. Also, nodes may shy away from responding to advertisements not from their neighbors due to a number of reasons such as selfishness, unwillingness to cooperate, or disinterest towards requests from non-neighbors. Hence, we do not forward the request by broadcasting it multiple hops. This also reduces the amount of network traffic.

3.3 Expanding Ring Replication Mechanism

In the previous section we discussed the data accessibility mechanism where the node initiates a content request to the server if it does not find a data item within its one-hop neighborhood. In our replication scheme, the data server monitors demand for each data item and replicates it in the network when demand crosses a certain threshold. The data server keeps track of the demand for each data item by measuring the frequency of requests for various data items. If n is the number of data requests it receives for a data item d_i in time T, then the access frequency f_i for d_i is n/T. This time period, T, is called the *Window Period*. If f_i exceeds a threshold value, λ_i, set by the server for d_i, then the server decides to replicate the data on one or more *capable* nodes in the network. An obvious question here is the accuracy of this method in estimating content access frequencies. For example, if a number of mobile hosts request a data item such that the item is eventually replicated near them and then do not request it again from the server since it is cached nearby, how does the server know whether the items are not being requested or are serviced by nearby nodes? We argue that if such a situation arises, it would be a success of the replication scheme. Also, if requests are not being generated for an item, then the server need not replicate such data items as they are either already replicated or are not being requested.

In the above discussion, we used the term *capable* node. Here, the capability of a node is a measure of a node's ability to service requests for the replicated data items. The data agent on the node computes the capability by applying a capability function. The capability function considers parameters such as available memory space, remaining battery power, and processing power[6]. We assume that such a function exists and it gives a good measure of the node's ability to cache data and service requests.

Node capability is an issue that has been ignored for the most part by research efforts in the past. In this paper, we designate a percentage of the nodes initially as capable nodes. Only these nodes have the ability to cache replicated

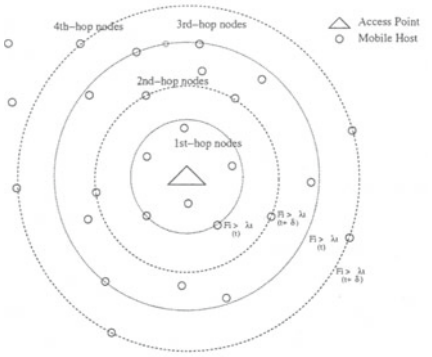

Fig. 1a. Expanding Ring Replication

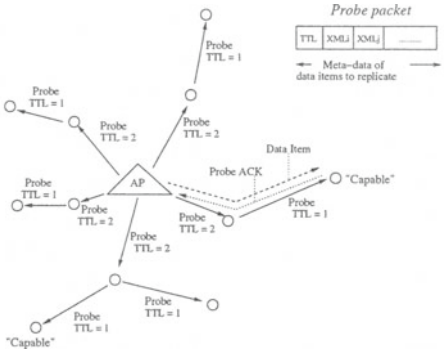

Fig. 1b. Expanding Ring Replication

data items and respond to advertisements for the same. Furthermore, in our simulations we also vary this set of capable nodes. After each window period, we toggle the capabilities of a fraction of the nodes in the network. Thus, some fraction of the capable nodes are *turned* into incapable nodes and an equal number of incapable nodes are made capable. This ensures a practical, dynamic simulation environment.

Our replication strategy to some extent resembles the expanding ring multicast query technique that is based on a Time-To-Live (TTL) search. In our scheme, the data server maintains a set of hop count values, S_i, for each data item, d_i. These hop count values represent the number of hops from the server where the data should be replicated. For example, the set of hop count values could be: $S_i = 1, 3, 5, etc.$ Each time the access frequency for a data item exceeds the threshold, the server selects the current set of hop count values for the data item. The server then attempts to replicate the data item on capable nodes in each of the hops in the set. Upon successfully replicating data, the server sets the values in the set, S_i, to the alternate hops which will now be: $S_i = 2, 4, 6, etc.$

The difference between successive hop count values is no less than two hops, i.e., data is not replicated on nodes nearer than alternate hops from each other. Typically, the transmission range of a wireless access card is around 150 feet in an indoors/crowded environment. Thus, if data is cached on a host, devices within a 300 feet diameter have access to it. This is the primary reason why the server picks alternate hops. The replication scheme is illustrated in Figure 1a.

To replicate the data in the network, the server probes the nodes $h \epsilon S$ hops into the network, soliciting their capabilities to replicate the data item. This is done by the server through broadcast of a small *probe* packet with the current set of hops, S, included in the packet. The TTL value of the probe packet is set to the maximum of the hop numbers in the current set, i.e., $TTL = max(h_l)$, where $h_l \epsilon S$. To reduce the broadcast traffic generated, the server does not broadcast a separate probe packet for every data item. Meta-data information about multiple data items is included in a single probe packet. Intermediate nodes decrease the TTL value by one and further broadcast the packet. This is

depicted in Figure 1b. Since the probe packet is broadcast by the server and the nodes, the possibility of duplicate probes reaching a node arises. We detect duplicate probe packets using sequence numbers and ignore them. When the probe packet reaches a host $h \in S$ hops away, the data agent on the host computes the capability of the mobile device to replicate a data item. If the device is capable, the data agent replies back to the server, sending a *probe acknowledgement* packet. The probe acknowledgement packet contains the list of items the node is interested in or has the memory space to cache. The server then directly unicasts the data items to the hosts from which it receives a probe acknowledgement as shown in Figure 1b. The data agent on the mobile host then marks the data item as *shared replicated* and services any requests for the data item.

3.4 Formal Analysis

Consider the same parameters used in Section 2.1. For content to be available to the nodes in the one-hop network, it has to be replicated by the server on at least one of the nodes in the one-hop network. To replicate the content, the server probes nodes h hops away soliciting their capabilities. The *capable* nodes then reply back to the server indicating their interests of data items to cache. The server then directly sends the data items to the *capable* nodes. Since the one-hop network under consideration is an average H hops away from the server, the number of *probe* messages processed by intermediate nodes will be $H * m$ in the worst case, i.e., if each of the m nodes in the one-hop network receives a probe packet on a different route. In the best case, all the nodes in the one-hop network may receive the probe packet on the same route; the number of messages processed by intermediate nodes in that case will be only H. In our analysis, we consider the worst case to demonstrate the effectiveness of the scheme. Say $1/p$ of the nodes are capable of caching the replicated data. The number of messages then processed by intermediate nodes to transmit replies to the server and content from the server will be $2 \times 1/p \times m \times H$ in the worst case. The number of additional messages then processed by intermediate nodes per unit time, per one-hop network, is

$$MP''_{add} = \frac{H \times m + 2 \times 1/p \times m \times H}{T} \tag{3}$$

As before, let $H = 3$ hops and $m = 6$ nodes. Let the percentage of capable nodes be 50%, i.e., $1/p = 1/2$. We then have $MP''_{add} = 36$ messages by intermediate nodes in the network. This is the overhead that the replication scheme has on intermediate nodes in the worst case. As we see, this is only one fifth of the overhead of the pure pull-based replication-less model and close to one third of the overhead of the improved replication model. We can imagine that this value would be considerably smaller in a better case where each node does not receive a probe packet on a different route, i.e., where some of the routes overlap.

Let r be the rate of replication, i.e., the number of times the server picks a set of hops $h_1, h_2, ..., h_l$ for replication in time T. The general equation for

the overhead imposed by our scheme per unit time in terms of the messages processed by network nodes is,

$$MP''_{add} = r \times \frac{\sum_{i=1}^{i=l} m \times (h_i + 2 \times h_i \times 1/p)}{T} \quad (4)$$

Note here that the index, i, increments in steps of 2. We can see from the above equation that the scheme is independent of the demand for the data item, i.e., the number of content requests generated in each hop of the network. Although the scheme depends on the number of nodes per hop of the network, we argue that the scheme will work better even in case of high-density networks. This is because in a high density network, the number of content requests generated will also be high and this will impact the performance of schemes that depend on requests going to the server or being forwarded on multiple hops in the network.

Our simulation results, which we now discuss, show that such a strategy efficiently replicates data in the network by gradually replicating the data in the network as the demand for data increases.

4 Evaluation

4.1 Simulation Model

We use the GloMoSim discrete event simulator to evaluate the performance of our proposed scheme. Our simulation environment is shown in Table 1, which shows a listing of the simulation parameters, the range of values tested, and the nominal value for each parameter when not varied. To model data requests by mobile hosts, each node throughout the simulation period generates random requests for data items with a random delay between any two subsequent requests. The random delay is the *Inter-request time* shown in Table 1. The server evaluates the access frequency for each data item once in every *Window Period.* In our simulations we model data requests as an approximate Zipf distribution [5]. We have three categories of data items: highly popular data items, moderately popular, and less popular data items. The probability of a request for a data item depends on the category to which it belongs and therefore to its popularity. All categories have an equal number of data items and the probability of occurrence of a data item within a category is uniformly random. We also vary the cache size of each node in the simulations to model the capability of a node. The cache size i.e., the number of replicated data items a node can hold, is modelled in terms of a fraction of the total number of items that the server delivers. In our simulations, we also vary the total number of nodes in the network that are capable of participating in the replication mechanism from a low 25% to a high of 75% of the total number of nodes.

4.2 *Response Time Simulation Results*

We first examine the effect of the number of nodes in the network on the data request response time. The Response Time, R_t, is measured as the time delay

Table 1. Simulation parameters and values

Parameter	Range of values simulated	Nominal value
Network area	$500m$x$500m$, $1000m$x$1000m$, $1500m$x$1500m$	$1000m$x$1000m$
Simulation time	$1200s$ - $1800s$	$1800s$
Mobility model	-	Random waypoint
Mobility	0-$10m/s$, $30s$ pause time	0-$5m/s$, $30s$ pause time
Number of Nodes	10-100	60
Percentage of capable nodes	25-75%	50%
Number of data items	10-50	15
Cache Size	10%-100% of total items	30% of total items
Inter-request time	-	20-$40s$
Ad-response time	-	$100ms$
Window period	-	$60s$
Number of Seed values	-	5

experienced by a node to receive a data item from the time it issues an interest advertisement. Thus, $R_t = T_{data\_received} - T_{interestad\_broadcast}$.

The average of the response times of all requests made within the simulation period is plotted against the number of nodes in the network. The graph of the simulation results is shown in Figure 2. From Figure 2 we can see that when there is no data replication, the average response time increases as the number of nodes in the network increases. This is because the increased number of data requests results in longer response times at the server. In our scheme, for a small number of nodes in the network, the average response time is higher than without replication. This is because of two reasons: (1) there are not sufficient nodes in the network to replicate data, and (2) the node density is very low and hence the probability of a node finding content within its neighborhood is greatly reduced. Since the node initially does a local one hop broadcast to search for content in its neighborhood but does not find it, this additional time results in higher average response times for fewer nodes. However, as the number of nodes in the network increase, we see that the average response times decrease appropriately. Due to a higher number of nodes in the network, the number of content requests rises. This results in the access frequency crossing the threshold at the server more often. The server then frequently attempts to replicate data in the network. As the number of data item copies increases in the network, more requests are satisfied within one hop of a node's range resulting in shorter response times.

4.3 *Message Processing Overhead Simulation Results*

In this section we evaluate the performance of our scheme with respect to the overhead it imposes on nodes in the network. We measure this overhead in terms of the number of *extra* messages processed by the nodes, as explained earlier in

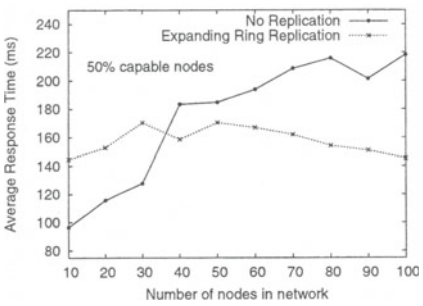

Fig. 2. Average response time versus number of nodes in the network

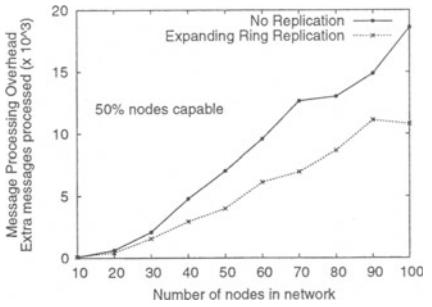

Fig. 3. Control message overhead versus number of nodes in the network

the formal analysis of the scheme. We vary the number of nodes in the network and measure the additional messages processed in each case. This also gives us an indication of the scalability of our scheme with respect to the population of hosts in the network. The graph depicting this is shown in Figure 3.

We can see from the graph that without replication, there is a steep linear rise in the number of messages processed as the network gets more dense. In our scheme, although there is a rise in the overhead, there is a significant difference between the extra messages processed in our scheme and one without replication. From the graph we can see that even in a dense network of 100 nodes, the number of additional messages processed in our scheme is on the order of 11,000 whereas without replication, the number is on the order of 19,000 messages. Thus there is a considerable decrease in the overall energy usage in terms of the additional messages processed in the network with our scheme.

4.4 *Capable Node Percentage Simulation Results*

We now evaluate the performance of our scheme when the actual number of capable devices can vary. To test this aspect, in each simulation run, only a percentage of the total number of nodes are capable. Therefore, when the server probes nodes h hops away, only some of the nodes respond to the probe packet. We evaluate this aspect with respect to the one-hop hit percentage, i.e., the fraction of the total number of times that a requested data item is found in a node's neighborhood or is present in the node's cache as a result of replication. The results of the simulation are shown in Figure 4.

It is clear from the graph that as the percentage of nodes capable and willing to cache data items increases, the hit ratio increases significantly. Also, as the number of nodes in the network increases, the one-hop hit ratio increases. From the graph we can also see that beyond a point (50% nodes capable), the increase in the one-hop hit ratio is not as rapid. This is because even if a higher number of nodes are capable of caching the data, only one of a nodes' neighbors needs to cache the data item for a node requesting a data item to find it within its neighborhood. Thus, we argue that the scheme will work well even if less than

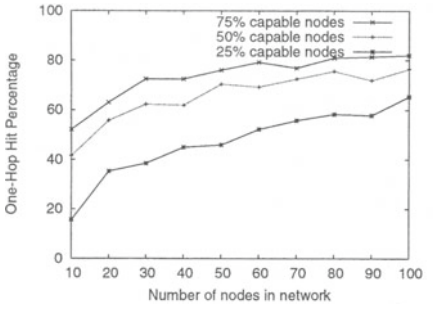

Fig. 4. Hit ratio versus number of nodes

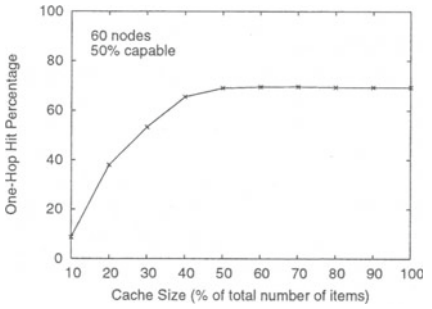

Fig. 5. Hit ratio versus cache size

100% of nodes in the network are capable of participating in the replication scheme.

4.5 *Cache Size Simulation Results*

Since a mobile device in general has less memory, we evaluate the effect of the cache size on the one-hop hit ratio. We vary the number of data items that a node is capable of caching in its *shared replicated* segment in terms of the percentage of the total number of data items offered by the server. The graph depicting the simulation result for one-hop hit percentage is shown in Figure 5. We can see that initially as the cache size increases, the one-hop hit ratio increases as more data items can be cached on a node and hence a greater number of requests are satisfied within a node's neighborhood. However, increasing the cache size beyond a point does not make a significant difference in the hit percentage because it then depends on the data items requested, i.e., the data request distribution pattern. Since the popular data items may have been already cached on one or more neighboring nodes, even if the cache size is increased, no major improvement is noticed. Hence, our scheme does not require an excessively large cache size. It depends on the cache size only to an extent where some nodes in each neighborhood can, in a combined manner, cache a few data items. The server replicates data items on these nodes based on their demand such that requests are satisfied within the one-hop neighborhood of a node. Also, since the server replicates a data item on multiple capable nodes in a hop, the caching capacity of a single node does not significantly affect the scheme.

5 Conclusions and Future Work

In this paper, we have proposed a lightweight replication mechanism for ad hoc networks by utilizing the communication channel in a pull based information delivery system. Since connectivity can be intermittent in such an environment, a node's best source for the data it needs may indeed be the nodes in its neighborhood. Thus, a mechanism that replicates data in a manner such that the

likelihood of finding data within the neighborhood is high, can effectively reduce the response time as well as the network traffic generated due to data requests.

The simulation results show that our scheme performs well in a variety of scenarios. To summarize, the features of our replication scheme are, 1) it is scalable with respect to the number of nodes in the network, 2) it is lightweight in terms of message processing overhead for the mobile device, 3) it does not depend excessively on the percentage of *capable* mobile devices in the network, and 4) it does not depend on a single mobile device's cache size to a large extent.

In our scheme we assume that if a node has the resources to cache a data item and service requests for it, then the node behaves as a cooperative member in the network. This unfortunately may not be true in a real ad hoc network setting. Hence, if an incentive scheme can be added to the system so that a node is encouraged to cache data items and share them with others, the effectiveness of the replication scheme can be greatly increased.

References

1. Acharya, S., Franklin, M., Zdonik, S.: Dissemination-based data delivery using broadcast disks. IEEE Personal Communications, 2(6) (1995)
2. Acharya, S., Franklin, M., Zdonik, S.: Balancing push and pull for data broadcast. In Proc. of the ACM SIGMOD (1997) 183-194
3. Aksoy, D., Franklin, M.: Scheduling for large-scale on-demand data broadcasting. In Proc. of the IEEE INFOCOM (1998) 651-659
4. Aksoy, D., Franklin, M.: Rxw: A scheduling approach for large-scale on-demand data broadcast. IEEE/ACM Transactions on Networking, Vol. 7 (1999) 846-860
5. Breslau, L., Cao, P., Fan, L., Phillips, G., Shenker, S.: Web caching and zipf-like distributions: Evidence and implications. In Proc. of the IEEE INFOCOM (1999) 126-134
6. Chen, K., Nahrstedt, K.: Cross-layer design for data accessibilty in mobile ad hoc networks. In Proc. of the fifth World Multiconference on Systemics, Cybernetics and Informatics (2001) 315-320
7. Deolasee, P., Katkar, A., Panchbudhe, A., Ramamritham, K., Shenoy, P.: Adaptive push-pull: disseminating dynamic web data. In the tenth International World Wide Web Conference on World Wide Web (2001) 265-274
8. Hameed, S., Vaidya, N.H.: Efficient algorithms for scheduling data broadcast. Wireless Networks, Vol. 5 (1999) 183-193
9. Hara, T.: Cooperative caching by mobile clients in push-based information systems. In Proc. of the eleventh International Conference on Information and Knowledge Management (2002) 186-193
10. Hara, T.: Effective replica allocation in ad hoc networks for improving data accessibility. In Proc. of the IEEE INFOCOM (2001) 1568-1576
11. Karakaya, M., Ulusoy, O.: Evaluation of a broadcast scheduling algorithm. Lecture Notes in Computer Science (2001)
12. Vaidya, N.H., Jiang, S.: Data broadcast in asymmetric wireless environments. In Proc. of the first International Workshop on Satellite-based Information Services (WOSBIS) (1996)