Performance Evaluations of Data-Centric Information Retrieval Schemes for DTNs

P. Yang, Member, IEEE M. Chuah[1], Senior Member, IEEE

Abstract—**Mobile nodes in some challenging network scenarios, e.g. battlefield and disaster recovery scenarios, suffer from intermittent connectivity and frequent partitions. Disruption Tolerant Network (DTN) technologies are designed to enable communications in such environments. Several DTN routing schemes have been proposed. However, not much work has been done on designing schemes that provide efficient information access in such challenging network scenarios. In this paper, we explore how a content-based information retrieval system can be designed for DTNs. There are three important design issues, namely (a) how data should be replicated and stored at multiple nodes, (b) how a query is disseminated in sparsely connected networks, (c) how a query response is routed back to the issuing node. We first describe how to select nodes for storing the replicated copies of data items. We consider the random and the intelligent caching schemes. In the random caching scheme, nodes that are encountered first by a data-generating node are selected to cache the extra copies while in the intelligent caching scheme, nodes that can potentially meet more nodes, e.g. faster nodes, are selected to cache the extra data copies. The number of replicated data copies K can be the same for all data items or varied depending on the access frequencies of the data items. In this work, we consider fixed, proportional and square-root replication schemes. Then, we describe two query dissemination schemes: (a) W-copy Selective Query Spraying (WSS) scheme, (b) L-hop Neighborhood Spraying (LNS) scheme. In the WSS scheme, nodes that can move faster are selected to cache the queries while in the LNS scheme, nodes that are within L-hop of a querying node will cache the queries. For message routing, we use an enhanced Prophet scheme where a next-hop node is selected only if its predicted delivery to the destination is higher than a certain threshold. We conduct extensive simulation studies to evaluate different combinations of the replication and query dissemination algorithms. Our results reveal that the scheme that performs the best is the one that uses the WSS scheme combined with binary spread of replicated data copies. The WSS scheme can achieve higher query success ratio when compared to a scheme that does not use any data and query replication. Furthermore, the square-root and proportional replication schemes provide higher query success ratio than the fixed copy approach with varying node density. In addition, the intelligent caching approach can further improve the query success ratio by 5.3% to 15.8% with varying node density. Our results using different mobility models reveal that the query success ratio degrades at most 7.3% when the community-based model is used compared to the Random Waypoint (RWP) model [13]. Compared to the RWP and the community-based mobility models, the UmassBusNet model from the DieselNet project [14] achieves much lower query success ratio because of the longer inter-node encounter time.**

Index Terms—**disruption tolerant networking, information retrieval, performance evaluation, data centric.**

[1] Corresponding author, chuah@cse.lehigh.edu, 19 Memorial Drive West, Bethlehem, PA 18015, USA, Phone 610-758-4061, Fax 610-758-4096.

I. INTRODUCTION

With the advances in technology, we have many wireless computing devices e.g. PDAs, sensors etc. Such devices can form infrastructureless ad hoc networks and communicate with one another via the help of intermediate nodes. Such ad hoc networks are very useful in several scenarios e.g. battlefield operations, vehicular ad hoc networks and disaster response scenarios. Many ad hoc routing schemes have been designed for ad hoc networks but such routing schemes are not useful in some challenging network scenarios where the nodes have intermittent connectivity and suffer from frequent partitioning. Recently, disruption tolerant network technologies [1],[2] have been proposed to allow nodes to communicate with one another in such extreme networking environment. Several DTN routing schemes [3],[4],[5],[6] have been proposed.

Although routing is an important design issue for such sparsely connected networks, the ability to access information rapidly is also an important feature that a DTN should have since the ultimate goal of having such a network is to allow mobile nodes to access information quickly and efficiently. For example, in a battlefield, soldiers need to access information related to detailed geographical maps, intelligent information about enemy locations, new commands from the general, weather information etc. In addition, a particular data item may be of interest to multiple soldiers, so it makes sense to replicate the data item, and store them at multiple nodes so that it can be accessed by other nodes. This allows us to save battery power, bandwidth consumption and the data item retrieval time. Such data caching also means that the source of the data items need not know the identities of the nodes that need to access the data items.

Researches on data access and dissemination techniques in ad hoc and sensor networks are not new. For example in [7], the authors consider the storage node placement problem aiming at minimizing the total energy cost for gathering data to the storage nodes and replying queries. In [8], the authors study the optimal number of replicas for a set of objects in large two-dimensional wireless mesh networks such that the access cost can be minimized. Their approach is not directly applicable to mobile networks since they assume stationary nodes in their environments. They also do not consider how a node can discover the replicas of the data items. In [9], the authors propose three distributed caching techniques for well-connected ad hoc networks, namely CacheData, CachePath and HybridCache. However, their techniques are only useful for well-connected ad hoc networks.

Two key technologies enhance our ability to access information efficiently on the Internet [19]. The first is the indexing and search infrastructure e.g. Google's search engine that enables one to access information by collecting and maintaining mappings of content to location. The second is a caching infrastructure that maintains a mapping from the content location to a cache location. These technologies, however, assume that the network devices are strongly connected to the Internet and hence are not tolerant to disruption. In addition, current technologies do not take into consideration that nodes move in some scenarios (e.g. mobile devices carried by human beings riding in vehicles). For a content-based information retrieval system to work in challenging network scenarios, data items need to be replicated and cached at different nodes. Questions like how many copies of each data item need to be replicated and which nodes should be selected to cache the replicated data items need to be answered. In addition, queries may need to be disseminated and cached by some intermediate nodes to increase the chances of them being answered in a timely manner. Again, the issue as to how a querying node selects other nodes to store its queries needs to be explored.

In this paper, we design a content-based information retrieval system for disruption tolerant networks. Our design focuses on answering questions related to *data caching*, *query disseminations* and *message routing*. For data caching, we explore two data caching schemes, namely (i) *random caching*, and (b) *intelligent*

caching. With *random caching*, each node generating a data item creates K tokens (representing K copies) and binary spread them to the nodes that they first encounter. With *intelligent caching*, each node generating a data item only spread the K tokens to carefully selected nodes that they encounter. The node selection is based on a friendliness metric: a node will be selected if it can meet many other nodes. The value for K can be fixed for all data items or varied depending on the access frequencies of each data item. For *query dissemination*, we explore two possible schemes, namely (a) *W-copy selective query spraying (WSS),* and (b) *L-hop neighborhood query spraying (LNS)*. In the *WSS* scheme, a querying node carefully selects W nodes based on a friendliness metric to cache replicated copies of the same query. In the *LNS* scheme, the querying node disseminates each query to its L-hop neighborhood. This is done by setting the TTL of its query message to L and any intermediate node that receives a query with a TTL more than one will rebroadcast the query after decrementing the TTL value. For *message routing*, we use an *enhanced Prophet* [3] scheme that only forwards messages to a next-hop node with an estimated delivery probability to the destination that exceeds a certain threshold. Other routing schemes can also be used.

We conduct extensive simulation studies to evaluate different combinations of these data and query dissemination schemes. Our simulation results reveal that the scheme that performs the best is the one that uses *intelligent caching* combined with the *WSS* scheme. With a fixed replication factor of 5 for all data items, the *WSS* scheme achieves higher query success rate over the *CacheData* scheme (a scheme proposed for well-connected ad hoc network) in network scenarios with varying node density. Our results also indicate that the *WSS* scheme achieves higher query success rate over the *LNS* scheme in scenarios with varying node density. In addition, our results with different mobility models indicate that the query performance degrades by at most 7.3% when the community based (CB) mobility model [11] is used compared to that achieved by the *Random Waypoint (RWP)* model [13] with varying query/data expiration time. With the UmassBusNet model (a mobility model built from traces collected in a vehicular ad hoc network) [14], the degradation can be as large as 51% (with data/query expiration time of 750s) or 31% (with data/query expiration time of 2000s).

In this paper, we assume that a new copy of a data item is generated only after the old copy expires so we do not have to address the data consistency issue. Such an assumption is reasonable for some application scenarios e.g. surveillance images are only refreshed every 15 minutes. In [24], the authors describe three types of consistency, namely strong, delta , and weak consistency. Our cache design works for the weak consistency model but can also work for delta consistency model with some minor changes. For example, each node that replicates data to other nodes stores the identifiers of these nodes so that when it receives an invalidation message from the source node, it can relay such messages so that old copies can be removed. We also do not address security issues in this work. More discussions on security design can be found in [22]. Another important design issue is related to the naming of the data items. A declarative language that allows us to specify facts, rules and location-based queries can extend the capabilities of current information retrieval system. For example, one would be interested in gathering traffic condition information within a kilometer radius around the Washington DC area on the fly. Some discussions on knowledge-based declarative approaches can be found in [19], [20].

The rest of the paper is organized as follows. In Section II, we discuss related work. In Section III, we describe our content-based information retrieval system for DTNs. In Section IV, we present the performance evaluation of the proposed schemes. In Section V, we discuss additional design issues that need to be addressed before an effective content-based information retrieval system can be deployed in DTNs. We conclude by discussing some future research that we intend to perform in Section VI.

II. RELATED WORK

A. DTN Routing Schemes

Several routing schemes have been designed for DTNs [3],[4],[5],[6], [28]. A good summary of various DTN routing schemes can be found in [27]. These different schemes can be grouped into three categories. The first category [5] uses special nodes called ferries to deliver messages between partitioned networks. Ferry routes have significant effect on the data delivery performance, hence they need to be designed efficiently. The second category [3],[4] uses multihop routing approach where contact history information is used to determine the next hop node to pass a message. For example, in [3], a probabilistic metric called delivery predictability is used to determine if a node needs to pass any stored messages to a new contact that it comes across. The third category [25],[26] uses a two-hop routing approach where the intermediate nodes that receive messages from any source have to store the messages until they can deliver the messages when they come into contact with the destinations of the messages. Sometimes, erasure-coding is used to encode and divide the message into multiple blocks and these different blocks are sent to different relays to increase the chances of a destination receiving a particular message since the destination only needs to receive a certain fraction of the encoded blocks to reconstruct the original message.

B. Data Caching Schemes for Ad Hoc Networks

In [16], the author considered the data replica allocation problem in ad hoc networks. More specifically, the author proposed three replica allocation methods, namely (a) Static Access Frequency (SAF) which makes use of the access frequency to each data item, (b) dynamic access frequency and neighborhood (DAFN) method which makes use of the access frequency of data items and the neighborhood of mobile hosts, and (c) dynamic connectivity based group (DCG) which considers both the access frequency and the whole network topology. In SAF, each mobile host caches the data items that it accesses most frequently. In DAFN, the approach is similar to SAF except that replications among neighboring nodes are eliminated. In DCG, nodes that are connected form groups and the replica allocation is determined based on the group access frequency. The disadvantage of these approaches is that they assume full knowledge of access frequencies and the ability to share such information in well-connected ad hoc networks.

In [7], the authors consider the storage node placement problem in sensor networks with the goal of minimizing the total energy cost for gathering data to the storage nodes and replying queries. The authors consider both fixed and dynamic tree models. For the fixed tree model, the authors give an exact solution on how to place storage nodes to minimize total energy cost. For the dynamic tree model, they design a scheme for each sensor node to select the storage node that results in the minimum communication cost for data forwarding and query diffusion and reply once the storage nodes have been positioned. This approach is not applicable to our problem because they assume that a single sink issues a query and that all the nodes do not move.

In [8], the authors consider an optimal replication strategy for large 2-D wireless mesh networks that can minimize object access cost. They found that to minimize the object access cost, the number of replicas for a data item should have is proportional to $p^{0.667}$ where p is the access probability of the object. They also present an online optimal replacement scheme and a localized replacement algorithm that can approximate the optimal replication scheme. Via simulations, they demonstrate that a significant performance gain can be achieved by the optimal strategy and their online replacement algorithm is very effective. Again, this approach is not applicable to the problem we study in this paper because it assumes that the nodes are fixed and that the access frequencies for all the objects are known apriori.

In [9], the authors present three distributed caching schemes, namely (a) CacheData which caches data items that pass by a node, (b) CachePath which caches the path to the nearest cache, and (c) HybridCache which caches the data item if its size is small enough or else the path to the data will be cached. An LRU policy is used for cache replacement. These three approaches however do not take into consideration how nodes within a neighborhood can collaborate such that the data caching will result in better access cost. In addition, their approach also assumes well-connected ad hoc networks.

In [18], Stillerman has proposed flooding queries using limited radius and also replicating object to improve the information retrieval latency. However, no detailed algorithms have been provided on how to replicate the objects. In addition, in our approach, the nodes carefully select which nodes to replicate their queries rather than blindly flooding them.

III. CONTENT-BASED INFORMATION RETRIEVAL DESIGN

There are three main components in our content-based information retrieval system, namely (a) data caching, (b) query dissemination, and (c) message routing e.g. routing query responses. Our system assumes that some nodes generate data items and some nodes generate queries. Note that data generating nodes can also be querying nodes. We further assume that all nodes are cooperative i.e. they are willing to store data items or queries when they are asked by other nodes. Each data item and each query has an identifier associated with it and has an expiration time e.g. a data item may look like <DataID, Data, Expired_Time>, a query may be of the form <QueryID, DataID, Expired_Time>. We assume all data items (or queries) have the same expiration times. Data items and queries are removed from storage nodes when they expire. Our system supports both push and pull mechanisms for information retrievals. For the push mechanism, each node replicates K copies of a data item it generates and pushes them to K selected nodes. The mechanism for selecting the K cached nodes is called the data caching scheme. Thus, newly generated information is pushed to several nodes to improve the latency of retrieving such information by other nodes. For the pull mechanism, each node that generates a query can disseminate copies of this query to W selected nodes or to its L-hop neighborhood. More details on how queries are disseminated will be discussed in the query dissemination section. Such query disseminations allow a node to pull data faster. Thus, contents can be searched and retrieved in our system even when connectivity is disrupted. Last but not least, we need an efficient message routing scheme to forward the query response message back to a querying node. We discuss in details the schemes we consider for these three components below:

A. *Periodic Beacons*

Periodically, each node broadcasts a beacon message. Such beacons are used for node discovery purposes. Several pieces of information are included in the beacon message, namely (a) the node's friendliness metric (FM), (b) a list of the descriptions of the data items that are currently stored in this particular node, (c) a list of the descriptions of the data items that are within the *M*-hop (*M* is set to 2) neighborhood of this node. The friendliness metric is a measured metric of the average number of unique nodes that a node encountered during an observation period. More details on how this metric is obtained will be given in Section II.B. For scalability, one can use the bloom filter approach to store the meta-descriptions of data items that are currently stored in a node. An example of such an approach for resource-constrained devices can be found in [29].

B. *Data Replication*

For the push mechanism, we assume that each data item is replicated to *K* copies where *K* can be set (a) the same for all data items (referred to as the Fixed Replication Scheme), (b) proportional to the access frequency of a data item (referred to as the Proportional Replication Scheme, and (c) proportional to the square-root of the normalized access frequency of a data item (referred to as the Square-Root Replication

Scheme). To distribute *K* copies of a data item, we assume that each newly generated data item has K tokens associated with it. The source node then selects a method to spread these copies to other nodes. In [11], the authors propose a spray and wait routing scheme where a source node generates *L* tokens for any message it generates. During the spraying phase, a node, n1, that carries a message copy with c tokens is allowed to spawn a message copy, allocate some of the tokens (say *m* where $m< c$) to that message copy and send it to another node n2. The node n1 retains *(c-m)* tokens for its own copy. Any node that receives a message copy with only one token can only forward this message to the destination itself (this is referred to as the wait phase). This scheme was originally designed for message routing but we use this idea to disseminate replicated data items. Thus, in our work, we assume that each node that receives a data item that carries *w* tokens (where $w > 1$) will retain $\left\lfloor \frac{w}{2} \right\rfloor$ tokens but send a replicated copy with $(w-\left\lfloor \frac{w}{2} \right\rfloor)$ tokens to a selected next-hop node. Such binary spread of replicated data items ensure that *K* copies can be distributed quickly. In challenging network environments where disruptions are frequent and intercontact times may be large, the data item may expire before *K* copies are replicated. Different nodes in the network have different moving characteristics. Hence, it is important to select the appropriate nodes for spreading copies of a data item. In this work, we investigate two methods that a node can use to select which nodes to spread the extra copies to. We refer to the two schemes as (a) the random caching scheme, and (b) the intelligent caching scheme. We describe these two schemes below:

1) Random Caching

For the random caching, each node, *j*, which has $(w>1)$ tokens for a data item will repeatedly pass a copy of the data item with $\left\lfloor \frac{w}{2} \right\rfloor$ tokens to whichever node that it encounters until $w=1$.

2) Intelligent Caching

Since the nodes may move with different maximum speeds, the nodes that move faster can encounter more nodes and hence are good candidates for storing replicated data items. Thus, we propose having each node measure the number of unique nodes that it observes within each observation window (set to be the same as the beacon interval in this paper) and maintains a metric called friendliness metric (FM) which is merely a smoothed estimate of the average number of unique nodes it encounters during an observation window. Let *T* be the observation window interval, $t_n =[(n-1)T, nT]$ be the n^{th} observation window, $V_i(n)$ be the number of unique nodes observed by node *i* during the n^{th} observation window, and $\tilde{V}_i$ be the smoothed version of $Vi(n)$. $\tilde{V}_i$ is used as the FM metric of node *i*. We show how $\tilde{V}_i$ is updated after each observation window in Eqn(1).

$$\tilde{V}_i(new) = \alpha V_i(n) + (1-\alpha)\tilde{V}_i(old) \qquad \text{Eq(1)}$$

where α is the usual smoothing coefficient. A higher value of α means the instantaneous value is given more weight while a lower value indicates that past observed values are given more weight. This smoothed *FM* value is included in the beacon message that a node periodically broadcasts. In the intelligent caching scheme, each node which has $w>1$ tokens for a data item will select a neighbor with a *FM* value that exceeds a threshold, *FM_threshold*, to be a storage node. This *FM_threshold* is chosen based on the node density. We set it to be twice the average number of nodes that a node can encounter in an observation window i.e. $FM\_threshold=4vRNT/A$ where *v* is the average node speed, *R* is the transmission range, *N* is the total number of nodes in the network, *T* is the observation window, and *A* is the network area. The observation window is set to be the same as the beacon interval in this work.

Figure 1 illustrates the K-copy intelligent caching scheme without the binary spreading of the tokens. *K* is set

to 3 in the picture. Let us assume that the node N10 is the generator of a data item, *d*. Then, N10 needs to select three storage nodes for the data item, *d*. At time t_1, N10 selects the first node it encounters, N4, to store the data since the *FM* value of N4 exceeds the *FM_threshold*. Then, N4 starts advertising the data item to N5 and N6. Later, at time t_2 , N10 moves and encounters N5 which also has an *FM* value that exceeds *FM_threshold*. Thus, N5 is also selected as a storage node for the data item, *d*. As N10 moves along, it encounters nodes N7, N1 at time t_3 but since their *FM* values are below the *FM_threshold*, N10 does not select them as storage nodes for the data item *d*. When N10 encounters N8 with a *FM* value that exceeds the *FM_threshold* at time t_4 , N10 selects this node to be the last storage node for the data item, *d*. N4, N5, and N8 will then include a description of the data item *d* in their beacons.

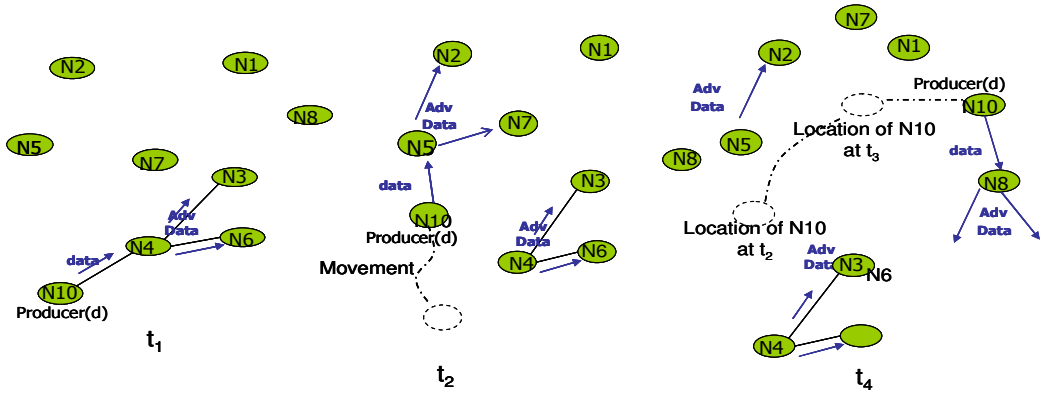


Figure 1 K-copy intelligent caching scheme

C. *Query Replication Schemes*

For the pull mechanism, we explore two query replication schemes, namely (a) the *W* copy selective query spraying (WSS) scheme, and (b) the L-hop local neighborhood spraying (LNS) scheme. Query replication can help to reduce the query response time but it can decrease the data efficiency (which is defined as the number of transmissions per query needed to get a response) because redundant replies may be generated when multiple copies of the same query exist in the network.

(a) WSS scheme

In the WSS scheme, again the binary replication method described in Section III.B is used for replicating queries. Assume that node i receives a copy of a query with w tokens. When node *i* encounters node *j,* node *i* first checks if node *j* has any data items within its local cache or *M*-hop neighborhood for the queries that node *i* stores. If there is, then node *i* will retrieve the corresponding data items. Otherwise, node *i* checks the Friendliness Metric (*FM*) value of node *j*. If the *FM* of node *j* exceeds the *FM_threshold*, node *i* disseminates a copy of the query with $\left\lfloor \frac{w}{2} \right\rfloor$ tokens to node *j* and retains (w-$\left\lfloor \frac{w}{2} \right\rfloor$) tokens for its own copy of the query. Each node follows the same replication procedure until the query has only one token.

In Figure 2(a), we illustrate how the WSS approach works. Assume that a query with four tokens is generated at node N8. N8 first encounters N6 which does not have the data items that match N8's new query. So, N8 disseminates the query with two tokens to its neighboring node N6 whose FM value exceeds the FM_threshold. Then, N8 moves to an area within the transmission range of N4 and N9. Since N4 does have the data item N8 is looking for, N8 requests for the data item. Thus, N8 gets back a query response while N6 is still trying to distribute the query.

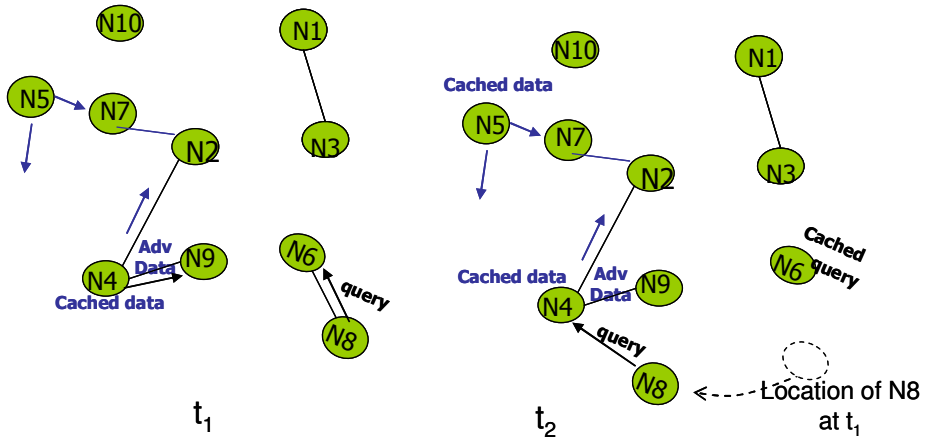


Figure 2(a): Query dissemination via the WSS approach

(b) LNS scheme

In the LNS scheme, each node that has a query broadcasts a query message after setting the TTL of the query message to be L. If a node does not have the data item requested in the query, that node immediately relays such a query after decrementing the TTL of the query message. In addition, it will also store the query message. Any node, *j*, that receives a query message with a TTL of one, will not relay the query further even if node *j* does not have that requested data item. If a node which receives the query has the requested data item, it will immediately generate a query response.

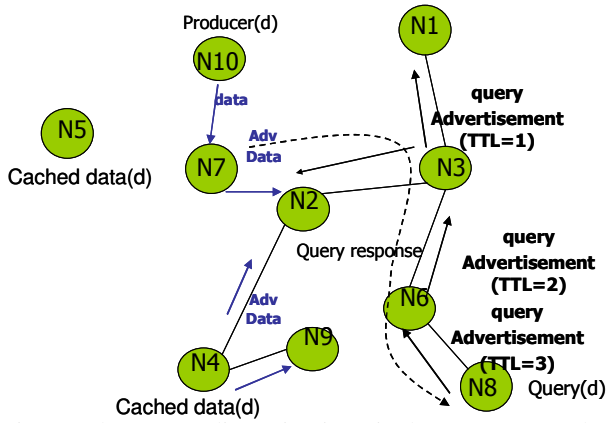


Figure 2(b): Query dissemination via the LNS approach

In Figure 2(b), we illustrate the LNS approach. N8 generates a query and broadcasts it with a TTL of 3. N6 that receives this query first stores the query since N6 does not have the requested data item. Then, N6 relays the query to N3 because the TTL still exceeds one after decrementing. N3 does not have the data item so it also stores and relays the query to N1 and N2. Since N2 knows N7 has the data item (from the periodic beacon N7 broadcasts), N2 requests for the data item and sends a query response to N8. If N2 does not have the requested data item, N2 merely stores the query but does not relay the query further since the TTL of the query that N2 receives is one.

Note that since a query is stored at multiple nodes when either WSS or LNS approach is used, multiple query responses may be generated. To increase the efficiency of the information retrieval system, each node can cache the identifiers of the query responses it has generated so that it does not relay any redundant query responses. Appendix 1 contains the pseudo codes for our data caching schemes and WSS/LNS query dissemination schemes.[2]

[2] We intend to release our NS-2 simulator in the near future.

D. DTN Message Routing Scheme

Once a query response is generated by a node, the query response will be delivered to the querying node using the underlying DTN message routing scheme. In this paper, we consider a DTN routing scheme which is an enhanced version of the Prophet [3] scheme. Prophet uses the history of encounters and transitivity to route messages for intermittently connected networks. In this scheme, each node broadcasts a beacon periodically. Prophet requires each node A to compute a probabilistic metric called the delivery predictability for each known destination B. This metric indicates how likely it is that node A will be able to deliver a message to that destination. When two nodes meet, they exchange summary vectors which contain the delivery predictability information stored at the nodes. This information is used to update the internal delivery predictability vector according to the following three equations, and then the information in the summary vector is used to decide which messages to request from the other node based on the forwarding strategy used. In Prophet, a node will forward a message to another node it encounters if that node has higher delivery predictability to the destination than itself. Such a scheme was shown to produce superior performance than epidemic routing [10]. The three equations used for updating the delivery predictability are as follow:

$$P(a,b) = P(a,b)_{old} + (1 - P(a,b)_{old}) * \alpha \qquad \text{Eq(2a)}$$

$$P(a,b) = P(a,b)_{old} * \gamma^{k} \qquad \text{Eq(2b)}$$

$$P(a,c) = P(a,c)_{old} + (1 - P(a,c)_{old}) * P(a,b) * P(b,c) * \beta \qquad \text{Eq(2c)}$$

Eq 2(a) is used to ensure that the nodes that are often encountered have a high delivery predictability. α is the initialization constant whose value lies between 0 and 1. Eq 2(b) is the aging equation which is used to reduce the delivery predictability when a pair of nodes does not encounter each other. γ is the aging constant whose value lies between 0 and 1. The delivery predictability also has a transitive property [3] such that if node A frequently encounters node B, and node B frequently encounters node C, then node C probably is a good node to forward messages destined to node A. Eq 2(c) shows how this transivity affects the delivery predictability where β is a scaling constant which decides how large an impact the transivity should have on the delivery predictability. In [3], α is set to 0.75, β is set to 0.25 and γ is set to 0.98. When a node receives a message that needs to be relayed, it will hold the message until it can pass it to a neighboring node which has the highest delivery predictability to the destination. Our simulation studies indicate that Prophet is not a loop-free routing protocol and hence may result in unnecessarily long delivery path. Thus, we enhance Prophet to let a node hold a packet until it can find a next-hop node with a delivery predictability that exceeds a threshold, DVR_Th, to the destination. This enhancement significantly reduces the number of hops it takes to deliver a packet to its destination and hence increases the data efficiency. The price to pay is a probable increase in the delivery latency.

IV. PERFORMANCE EVALUATION

In this section, we first present the analysis of a scheme where the data items are duplicated using the binary spread approach but there is no query dissemination. We refer this scheme as the No Query Duplication (NQD) scheme. This is one of the schemes which we study that can be analyzed and can be used as a baseline to compare against other schemes that use query duplication. Next, we compare the analytical results we obtain with the simulation results. Then, we use simulation studies to compare the different querying and replication schemes and study the impact of different parameter values on the query performance.

A. Analytical approach for the NQD scheme

Let us assume that there are N nodes within a network area of A. We further assume that the nodes are uniformly distributed over the network area. The average node speed is v and the transmission range of the radio at each node is R. The period for a node to encounter another node is $t=(A/N)/(2vR)$. The data expiration

time is T_e. Let us consider a node (S) which generates a data item i. Using the binary spread method, the maximum number of copies that node S can disseminate before the data expires is $2^{\lfloor T_e/t \rfloor}$. If node S intends to disseminate K number of copies of the data item i to different K nodes, the number of copies that node S can indeed disseminate is $C_i = \min(K, 2^{\lfloor T_e/t \rfloor})$. When a query for data item i is generated at node B, node B will search its own cache and its 2-hop neighbor's cache for data item *i*. The number of nodes within 2-hop transmission range for node B (including itself since node B caches data items as well) is $N_B = \pi(2R)^2 N/A$. For data item *i,* there are C_i copies in the network. The probability that the querying node *B* finds the data item *i* is

$$P_i = 1-(1-\frac{C_i}{N})^{N_B} \text{ where } N_B = \frac{\pi 4R^2 N}{A}, C_i = \min(K, 2^{\left\lfloor \frac{T_e}{t} \right\rfloor}) \qquad \text{Eq (3)}$$

We conduct some simulation experiments to understand how close the observed success ratio from the simulation is compared to what is predicted from Eq (3). In our experiments, we set *K, N,* T_e , and *R* to 5, 40, 1000seconds, and 250 meters respectively. We vary *A* as 1000x1000, 2000x2000, 3000x3000 and 4000x4000m^2. We plot the simulation and the analytical results (obtained using Eq (3)) in Figure 3. The results indicate that our analytical and the simulation results match closely.

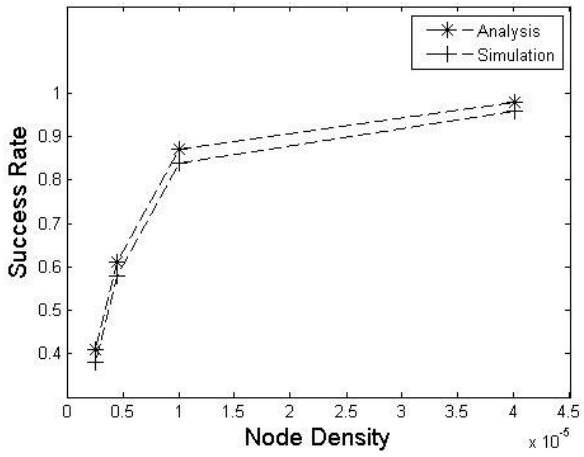


Fig 3: Analytical and simulation results for the NQD scheme with varying node density.

B. Simulation Setup

To investigate the usefulness of different combinations of our designed schemes, we implement the different data caching, query spraying and message routing schemes in NS-2 version 2.29 [12]. We assume that the wireless bandwidth is 2 Mb/s and the transmission range is 250 m. For some experiments, we use a network that consists of 40 nodes that are randomly placed in a network area. We use four sizes for the network area, namely (a) 1000x1000m^2, (b) 2000x2000m^2, (c) 3000x3000m^2 (default), and (d) 4000x4000 m^2. To show that our designed schemes still work for larger scale networks, we increase the network size to 100 nodes and the corresponding network areas to (e) 1600x1600 m^2,(f) 3200x3200 m^2, (g) 4750x4750 m^2, and (h) 6300x6300 m^2 in order to maintain the same network density as the 40-node scenarios. The enhanced prophet scheme is used to route query responses back to the querying nodes and DVR_Th is set to 0.1. This value is chosen after some experiments and seems to provide better data efficiency values for the scenarios we study. The query success ratio is not too sensitive to the value chosen for DVR_Th but the data efficiency is. For the WSS scheme, *M* is set to 2 because no additional control messages are needed to maintain such information and using larger number does not help since topology changes frequently in sparse ad hoc networks. To avoid having too many redundant query responses, we set W in the WSS scheme and the L in the LNS scheme to 5.

Node Movement Model: The nodes move either according to (a) the random waypoint (RWP) mobility model [13], (b) the UmassBusNet [14] model and (c) the Community-based (CB) mobility model [11]. These mobility models are chosen because (i) RWP is a well-known mobility model, (ii) the UMassBusNet model is based on traces collected from a real vehicular ad hoc network, (iii) the CB model is a non-homogeneous model. For the RWP model, each node selects a random destination and moves towards the destination with a speed chosen randomly between (*vmin*, *vmax*) m/s. After the node reaches its destination, it pauses for a period of time and repeats this movement pattern. Unless otherwise stated, *vmin* is set to 1, and *vmax* is set to 5 m/s for all nodes in the homogenous model (referred to as RWP). For the UMassBusNet model, we extract the locations of twenty buses at different times from multiple traces, and scale their relative locations to fit into the geographical area of interest. We mimic the CB mobility model proposed in [11] as follows: 50% of nodes randomly choose their initial locations and move according to RWP within a restricted area (1% of the total area) centered around their initial locations; while, the other 50% of nodes moves freely according to RWP within the entire simulation area.

Data Item Generation Model: For the small scale 40-node network scenarios described above, 10 out of the 40 nodes generate data items and each node generates 2 data items periodically. For the large scale 100-node network scenarios described above, 5 out of 100 nodes generate data items and each node generates 20 data items periodically. For the CB model, the data nodes are selected from those nodes that move within small area. The data item interval is set to be the same as the data expiration time so that we can keep the total number of unique data items fixed. In this paper, we assume that each data item has a fixed expiration time. The default data item expiry time is set to 1000 seconds. Intermediate nodes check the expiration time of the data item contained within a query response they receive. Any query response messages that contain expired data items will be dropped. In the future, we intend to explore the scenario where data item expiration time is not fixed. For such scenarios, the data consistency issue needs to be dealt with.

Query Model: The query model is similar to what have been studied in the past [8][9]. Twenty nodes generate read-only queries periodically. For the CB model, the querying nodes are selected from those nodes that move locally. The access pattern is based on the Zipf-like distribution which has been frequently used in [15] to model a non-uniform access pattern. In the Zipf-like distribution, the access probability of the i^{th} $(1 \le i \le N)$ data item is represented as follows:

$$P_i = \frac{1}{i^{\theta} \sum_{k=1}^{n} \frac{1}{k^{\theta}}} \qquad \text{Eq (4)}$$

where $1 \le \theta \le 1$. θ is used to represent the skewness of the queries. When $\theta = 1$ (the default value we use), it follows the strict Zipf distribution. When $\theta = 0$, it follows the uniform distribution. Larger θ results in more "skewed" access distribution. θ=0.8 has been shown to generate access pattern similar to those real web traces [15]. Each node allocates enough buffer space to store 500 queries. Unless otherwise stated, the query expiration time is set to 1000 seconds.

The performance metrics that we use to compare different combinations of schemes are

- Query success ratio – this is measured as the average number of successful queries (those which the querying node receives responses before the query expires) over the total number of queries generated.
- Query Response time – this is measured as the average time it takes for the successful query response to arrive back at a node which issues a query.
- Data Efficiency – this is measured as the number of useful data bytes over the total transmitted data bytes (does not include control overhead).

- Overall Efficiency – this is measured as the number of received data bytes over the total number of transmitted bytes (which include control overhead).

Each data point that we report in the simulation results is the average of ten runs. Each simulation has a warm up period of 1000s. Queries are generated only after the warm up period and a total of 1000 queries are generated. The default query rate is one query every 5s. After that, the simulation continues for another 5000s to allow more queries to be completed.

C. Simulation Results

In this subsection, we present our simulation results for the various experiments we conduct. In our first set of experiment, we are interested in understanding how sensitive the query performance of the WSS scheme is to the FM_Threshold value. Next, we conduct an experiment to understand which combinations of the data and query replication schemes will work better in different node density. We are also interested in comparing our schemes with the CACHE-DATA scheme described in [9]. Our results indicates that WSS performs the best when the RWP model is used. However, we are interested in knowing whether WSS still performs well when different mobility models are used. Thus, we conduct another set of experiments where we study the performance of the WSS scheme using three mobility models, namely the RWP model, the CB-model, and the UMassBusNet model. These mobility models have different characteristics e.g. the CB model is a non-homogeneous model where the global nodes can reach more nodes while the local nodes cannot. We anticipate that the query performance will degrade when the nodes move such that they incur longer node encounter time e.g. in the UMassBusNet model. We also anticipate that the query performance achieved in the CB model may be better than the UMassBusNet model since our scheme utilizes the friendliness metric to select nodes to cache data items.

In our fourth set of experiments, we are interested in understanding whether we can see better query performance when we use either the proportional or square-root replication techniques rather than the fixed copies approach for the case where the queries follow a Zipf-based model. In sparsely connected environments and finite data expiration times, not all copies of the frequently assessed data items can be replicated before the data expires. Next, we study the performance gain that can be achieved with intelligent caching over what can be achieved using random caching when a non-homogeneous mobility model is used. Lastly, we study the impact of query expiration time on the query performance.

1) Impact of FM_Threshold on Query Performance

We first conduct an experiment to evaluate the impact of the FM_Threshold on the query performance of the WSS scheme. We use the network scenario with 40 nodes distributed over 3000x3000 m^2. The nodes are assumed to move according to the CB model. The data and query expiration time is set to be 1000s. 10 nodes generate data items. The data generation rate of each data source is set such that we maintain a total of 400 data items in the network. 20 nodes generate queries with each querying node generates 1 query/5s. The Zipf-based querying model is used. The intelligent data caching scheme is used and WSS is used as the query dissemination scheme. The data replication factor, K, is set to be 5. M is chosen to be 2 and W is chosen to be 5 for the WSS scheme. The proposed FM_threshold (=4vRNT/A) value with R=250m and v=3m/s and t=5s is 0.067. Table 1 shows the query success rate as the FM_Threshold is varied from 0.03 to 0.09. The results indicate that the query performance is the best when FM_Threshold is set at 0.07 (closed to the proposed threshold). Thus, we use this setting for the rest of our simulations using the WSS scheme.

FM Threshold	0.03	0.05	0.07	0.09
Success rate	0.60	0.61	0.64	0.59
Response time	92	93	91	91
Data efficiency	0.28	0.28	0.29	0.29

Table 1: Query Performance vs FM_Threshold for the WSS scheme

2) Impact of Data and Query Replication Schemes

Next, we investigate whether it makes a big difference when different data and query replication schemes are used. The enhanced Prophet scheme is used to route query responses back to the querying node. In this experiment, we use the network scenario where 40 nodes are distributed over 1000x1000 m^2, 2000x2000 m^2, 3000x3000 m^2, and 4000x4000 m^2. We use the random waypoint mobility model where all nodes have a maximum speed of 5 m/s. As in the previous section, 10 nodes generate data items, and 20 nodes generate Zipf-based queries. Each querying node generates one query every 5 seconds. Five different data/query replication schemes are simulated, namely (a) the No Data Duplication (NDD) scheme where no data and query replication is used; (b) the No Query Duplication (NQD) scheme where only binary spreading scheme is used for data replication; (c) LNS, which uses the binary spreading scheme for data replication and LNS query dissemination; (d) WSS, which is the same as (c) except that WSS is used for query dissemination; and (e)CACHE-DATA, which is a data caching scheme proposed for dense sensor network in [22] . For cases (c) and (d), we set the data replication factor to be 5. For the WSS scheme, we set W to 5 but for the LNS scheme, we set L to 2. Table 2 summarizes all these five schemes we consider in this set of experiments.

Schemes	Descriptions
NDD	No Data/Query Replication
NQD	Data Replication Only
LNS	Data Replication with LNS for Query Dissemination
WSS	Data Replication with WSS for Query Dissemination
CACHE-DATA	Data Caching Scheme [9]

Table 2: The Five Schemes Considered

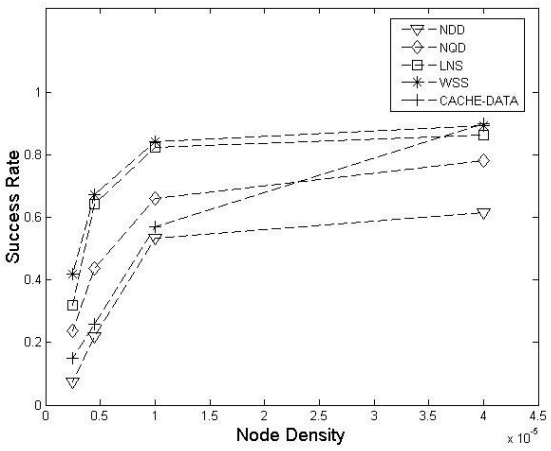
Figures 4(a), 4(b), and 4(c) plots the average query success ratio, the average query response time (for successful queries), and the data efficiency obtained for this set of experiments. The results in figure 4(a) show that WSS is the best scheme. When compared to the NDD scheme, the NQD scheme achieves 54% to 270% higher query success ratio with varying node density. The much improved query success ratio shows the great benefits of replicating data items in DTN environments. When queries are replicated (using either LNS or WSS scheme), we can achieve an additional 3-80% query success rate improvement. Overall, the WSS scheme achieves 45% to 460% higher query success rate when compared to the NDD scheme. WSS achieves 7% higher query success rate when compared to that achieved using the CACHE-DATA scheme in a relatively dense network (40 nodes over 1000x1000 m^2) and 200% higher query success rate in a sparse environment (40 nodes over 4000x4000m^2). The query success rate for the NQD, LNS and WSS schemes are much better than the CACHE-DATA scheme when the network is sparse due to the data replication. Replicated data copies allow more queries to be answered from the caches. The WSS and LNS schemes are better than the NQD scheme due to the query replication. WSS achieves 3.7%~6.7% higher query success ratio than LNS. Table 3 tabulates the average number of query copies for the WSS and LNS schemes at different node densities. The numbers show that both schemes have comparable number of replicated query copies. Thus, WSS performs better because the query messages are more likely to be disseminated to nodes with higher *FM* values when the WSS scheme is used, and such nodes have better chances of meeting the nodes that cache the queried data items.

The results in Figure 4(b) show that NQD, LNS and WSS have shorter response times than the CACHE-DATA scheme in sparse networks due to the data replication. Replicated data copies allow more queries to be answered from the caches and hence shorter response time. The CACHE-DATA scheme has shorter response time than others in the dense network because flooding is used to discover the queried data item, but such a flooding method is still not sufficient to retrieve data items before queries expire in a sparse network. *WSS has longer query response time than the NQD, NDD and LNS schemes because more queries which fail in other schemes are successful when WSS is used but such successful query responses take longer hops. However, if we consider only those packets that are successfully delivered in all schemes, the response*

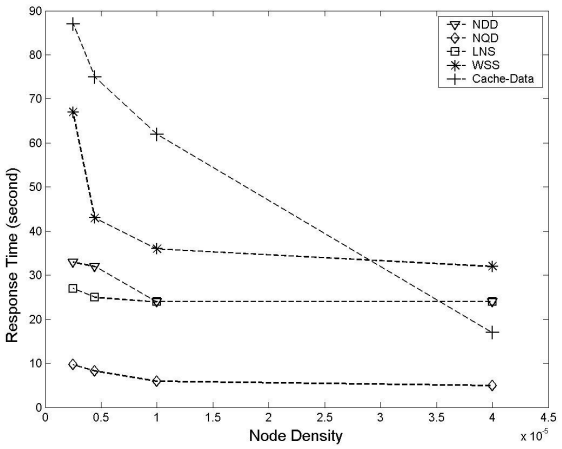
time achieved by NQD, LNS and WSS are similar. The results in Figure 4(c) show that WSS and LNS have lower data efficiency than the NQD scheme since more query responses are delivered via longer hops and there are redundant query responses due to the query replication.

Area (m^2)	1000x1000	2000x2000	3000x3000	4000x4000
WSS	5	4.5	3.9	3.2
LNS	4.8	4.3	3.8	3.3

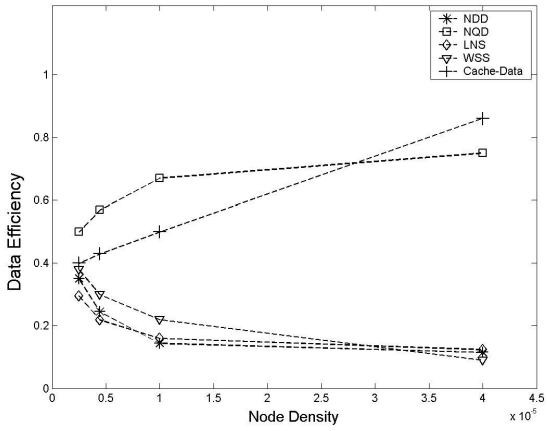
Table 3: Average number of copies/query for WSS and LNS schemes



(a) Query Success Ratio vs Node Density



(b) Average Response Time vs Node Density



(c) Data Efficiency vs Node Density

Figure 4: Comparison of Different Schemes using 40-node Network

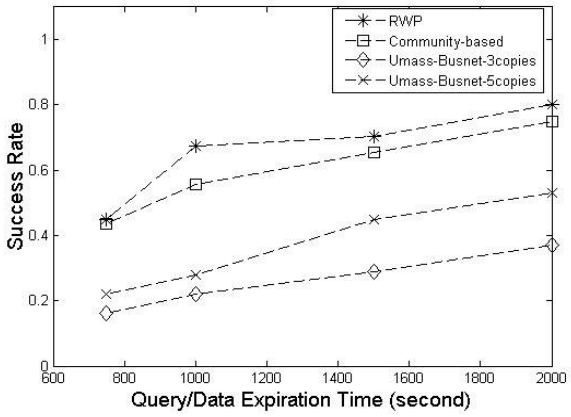
3) Impact of Mobility Model

In this subsection, we investigate how query success ratio changes when different mobility models are used. This experiment was conducted to ensure that our schemes can perform better than existing schemes that do not use data and query replications in some well-known or useful mobility models. We consider three mobility models, namely (a) the random waypoint (RWP) mobility model [13], (b) the Community-Based (CB) mobility model [11], and (c) the UmassBusNet [14] model. We use the 40-node distributed over 3000x3000m^2 network scenario. In our experiments, we set the data and query expiration times to be the same, and vary their values from 750, 1000, 1500 to 2000 seconds. We set K=5 for the data replication factor for the first two mobility models and set K=3 for the UmassBusNet model in order to keep the percentage of nodes having data items the same for all models (since only 20 nodes with long enough moving activities are found

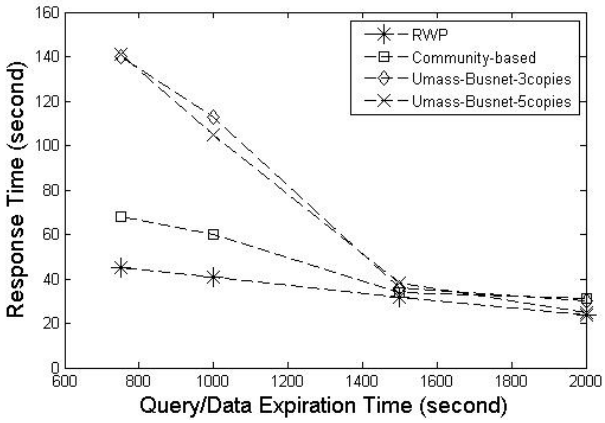
in the UMassBusNet traces). Since the success ratio for the UmassBusNet model is not high when we set *K=3*, we further try *K*=5. Our results show that performance can be improved by choosing a higher *K* value for the UMassBusNet model. Here, we only report the simulation results using the WSS scheme since this is the best scheme. The query replication factor, *W*, for the WSS scheme is set to 5. The comparison results between the WSS and the LNS schemes for the CB model are also included in Appendix 2.

The results for the average query success ratio, the average query response time (for successful queries), and the average data efficiency are plotted in Figures 5(a), 5(b), 5(c). The results from Figure 5(a) show that as the data/query expiration time increases, the query success ratio increases since more queries can enjoy the benefits of data/query caching. The results in Figure 5(a) also show that the highest query success ratio is achieved using the RWP mobility model, followed by the Community-based mobility model, and the UMassBusNet model. This is because the RWP mobility model has the shortest inter-contact time, followed by the community-based (CB) mobility model, and the UMassBusNet model. For example, with a 750s query expiration time, a node in the RWP model can have about 4 encounters before the query expires. Corresponding numbers for the CB model and UMassBusNet model are 3 and 1.3 respectively. For a 2000s query expiration time, the corresponding number of node encounters for the three models (RWP, CB and UMassBusNet) are 11, 8 and 3 respectively.

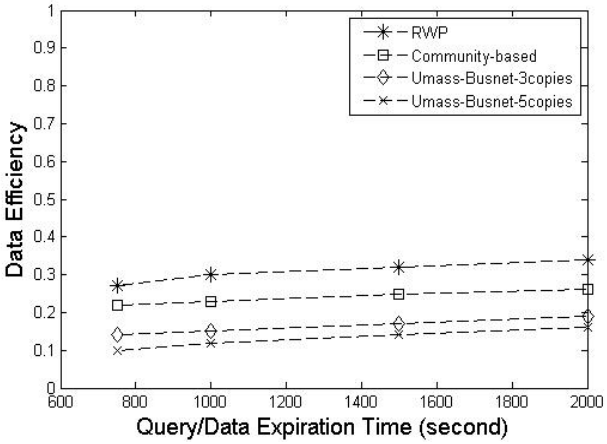
The results in Figure 5(b) show that the average response time is the smallest with the RWP model, followed by the Community-based model, and the UMassBusNet model. Again the large delay for the UMassBusNet model is caused by its high intercontact time.



(a): Query Success Ratio vs Query/Data Expiration Time



(b) Avg Response Time vs Query/Data Expiration Time

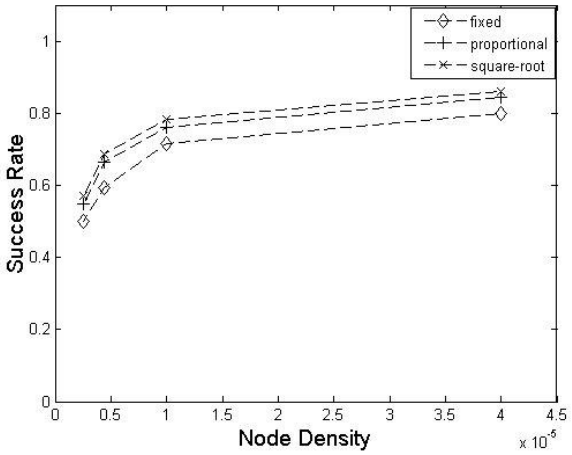


(c) Data Efficiency vs Query/Data Expiration Time

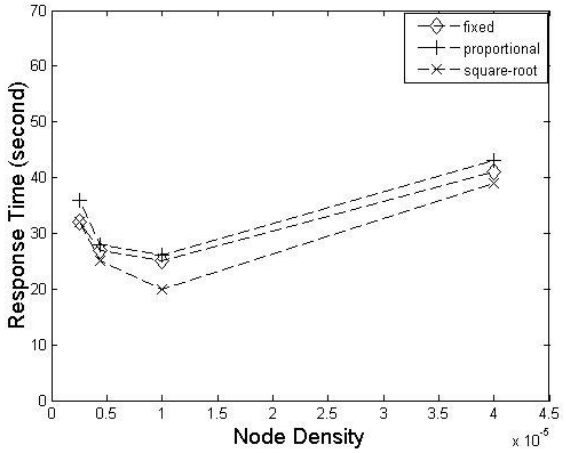
Figure 5: Performance Difference Using Different Mobility Models

4) Impact of Different Data Replication Schemes

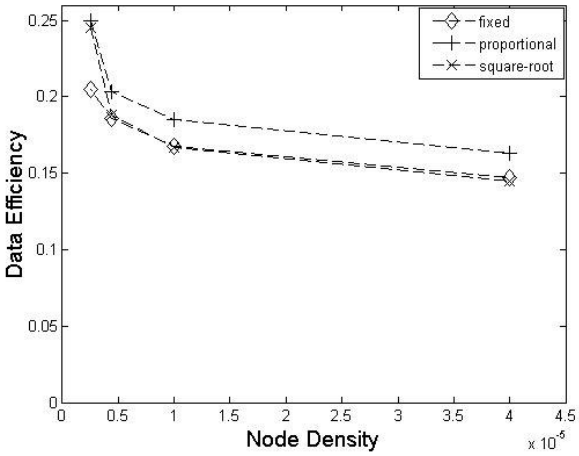
In traditional peer-to-peer networks, it has been shown that both the proportional and square-root replication methods [21] can achieve better data retrieval performance. Since in DTN environments, the nodes may not meet one another for a long time, therefore, not all copies of the frequently accessed data items can be replicated before the data expired. Thus, we are interested in understanding how much difference in performance one can see when the proportional and square-root replication schemes are used compared to the fixed copy scheme in DTNs. We use the network scenario with 100 nodes distributed over an area of size (a) 1600x1600, (b) 3200x3200, (c) 4750x4750, and (d) 6300x6300 m^2. For the proportional and square-root schemes, we assume the total number of buffers in the network allocated for storing all the replicated data items is 400. Five nodes generate data items and 20 nodes generate queries. Each querying node generates one query every 5s. Using the proportional replication scheme, the number of replicated copies for the top 5 most frequently accessed data items are (i) 72 , (ii) 35, (iii) 25, (iv) 19, and (v) 16 respectively. Using the square-root method, the number of replicated copies for the top 5 most frequently accessed data items are (i) 21 ,(ii) 15, (iii) 12 , (iv) 10, and (v) 10. For the fixed method, we set the replication factor K=4 (which uses the same total number of buffers (400) to store replicated data copies). The query and data expiration times are both set to 3000s.



(a) Query Success Ratio vs Node Density



(b) Average Response Time vs Node Density



(c) Data Efficiency vs Node Density

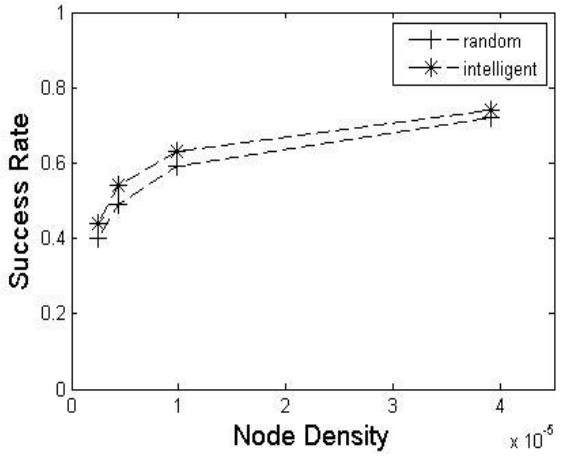
Figure 6: Impact of Different Replication Copies

Figure 6(a) to 6(c) plot the average query success ratio, the average query response time and the average data efficiency at different node density when different data replication copies are used. The results indicate that the proportional and square-root methods achieve 4.2% to 12.2% improvement in query success rate when

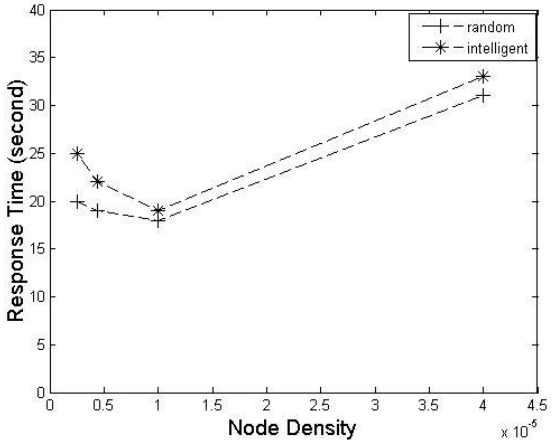
compared to the fixed copy method. Our results also indicate that the square root method has the highest query success ratio, the lowest average response time among the three replication schemes. In Figure 6(b), we observe that the average successful query response time seems to be the lowest for the scenario with 100 nodes over 3200x3200m^2. As the node density decreases, the longer inter-node encounter time is expected to cause the average query response time to increase. So, we only need to explain why the average query response time for the denser scenario (100 nodes over 1600x1600m^2) is higher. When the network becomes denser, more queries that fail to succeed in the less dense scenario can be successful but these transactions take longer time to complete. For example, when the network density increases from 100 nodes over 3200x3200m^2 to 100 nodes over 1600x1600m^2, the query success ratio using the square-root replication scheme increases from 78% to 87% and the mean query response time increases from 25 seconds to 37 seconds. If we consider only the same query responses which contribute to the 78% success ratio for both network densities, then their mean query response times are similar, which are 25 seconds and 24 seconds respectively. The longer query response time with higher node density is caused by the longer mean query response time (201 seconds) of the remaining 9% (from 78% to 87%) queries that succeed in the dense scenario but fail in the less dense scenario. As for data efficiency, the proportional method achieves the highest data efficiency because it replicates many copies of those frequently accessed data items and hence such items can usually be found in the caches. However, the data efficiency achieved by the square root method when the network is very sparse is similar to that with the proportional method.

5) Impact of Intelligent and Random Caching Schemes

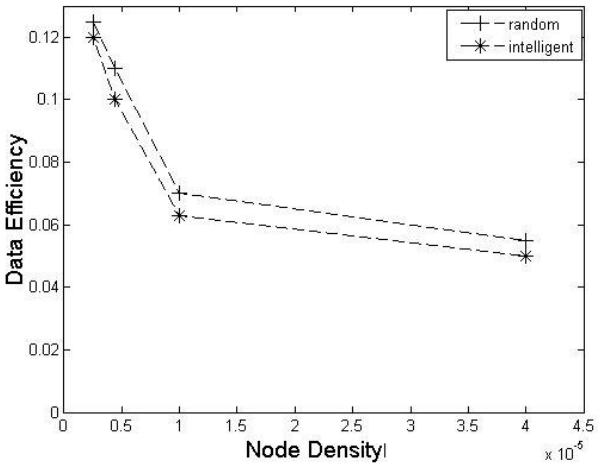
In this section, we compare the performance difference when intelligent or random caching schemes are used. We use the network scenario with 100 nodes spread over (i) 1600x1600, (ii) 3200x3200, (iii) 4750x4750, and (iv) 6300x6300 m^2. The nodes move according to the CB model where 50 nodes move locally (within 1% of the total area) and 50 nodes move globally over the entire area. The data and query expiration times are both set to 3000s. 5 nodes generate data items, and 20 nodes generate queries, each generating 1 query/5s. The items queried follow the Zipf model. We use the square-root replication scheme with a total buffer capacity of 400. Figure 7(a)-7(c) plot the query success ratio, the average response time and the data efficiency for the intelligent and random schemes with different node densities. The results indicate that the intelligent caching scheme improves the query success ratio by 5.3 to 15.8% with varying node density when compared to the random caching scheme. Some of the queries that fail using the random scheme can be answered using the intelligent scheme but these are accomplished with more hops and hence we see slightly larger query response time and smaller data efficiency for the intelligent scheme.



(a): Delivery Ratio vs Node Density



(b): Avg Response Time vs Node Density

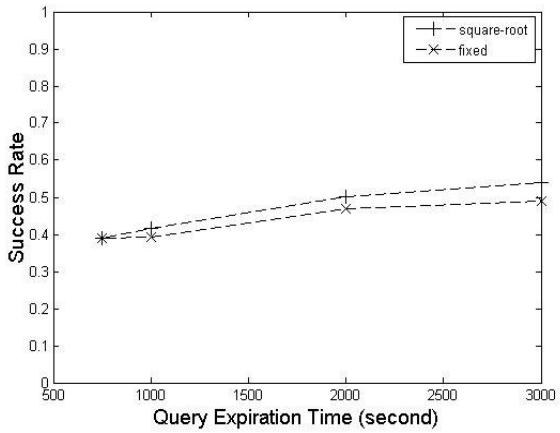


(c) Data Efficiency vs Node Density

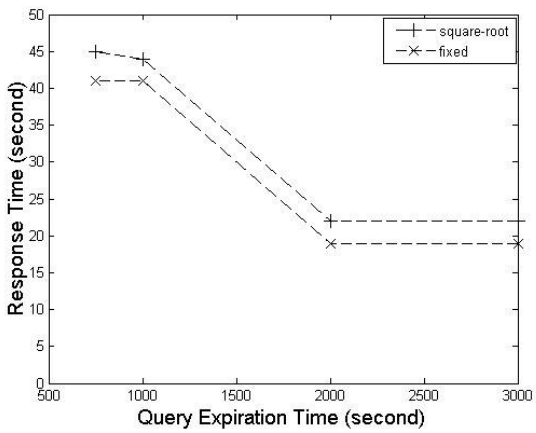
Figure 7: Intelligent vs Random Caching Schemes

6) Impact of query expiration time

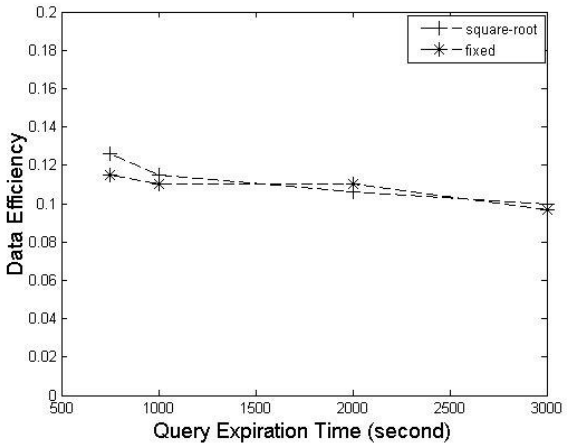
In our fifth set of experiments, we investigate the impact of query expiration time (QET) on the query success ratio, the average query response time and the data efficiency. We use the network scenario where 100 nodes are randomly distributed over an area of size 4750x4750 m^2. The nodes move according to the CB model with 50% of the nodes move locally and 50% of the nodes move over the whole area. Fifty nodes generate queries; each node generates one query every 5 seconds. We use the intelligent caching and the WSS scheme. We use either the square-root or fixed copy replication methods for data replication. We fix the data expiration time to 3000s but varies the query expiration time from 750s to 3000s. Figures 8(a), 8(b), 8(c) plot the average query success ratio, the average query response time, the average data efficiency obtained from the simulation results.



(a) Query Success Rate vs Query Expiration Time



(b) Query Response Time vs Query Expiration Time



(c) Data Efficiency vs Query Expiration Time

Figure 8: Impact of Query Expiration Time on Query Performance

When the query expiration time increases, the query success ratio increases since more queries can be answered if the queries can stay in the network longer. In addition, as the query expiration time increases, the average query response time drops because the querying nodes have higher chances of meeting the nodes carrying the requested data items directly. The data efficiency decreases slightly with increasing query expiration time because the longer query expiration time allows some queries that take more hops to be successful. The data efficiencies for both square-root and fixed copy approach are similar because the cost of replication is small compared to the cost of querying dissemination. The interesting thing to note is that with short query expiration time, the fixed copy approach can achieve as good a query success ratio as the square-root replication approach. When the query expiration time is 1000s, the square-root approach can achieve 5.9% better query success ratio but it can achieve 10.2% better query success ratio when the query expiration time is 3000s.

V. DISCUSSION

Using simulation studies, we have shown that data and query replications help to improve the query success ratio in sparsely connected mobile ad hoc networks. The WSS query dissemination scheme combined with the binary spread data replication method can improve the query success ratio by 45% to 460% in networks with varying node densities. In addition, our results also indicate that square-root replication scheme can still improve the query performance over the fixed copy approach, even though in some cases, not all copies of the same data item can be replicated before the data expires in very sparse mobile ad hoc networks.

To successfully deploy an effective information retrieval system in DTN, apart from the data/query dissemination and replication issues discussed in earlier sections of this paper, there are three other important issues that need to be addressed. One is related to security. We need to ensure that data items can only be accessed by legitimate nodes. We refer the readers to another paper [22] for discussions on the security related issues. The second topic is related to an efficient design of indices for the cached data items so that a node can quickly determine if the newly encountered nodes carry any data items that match the queries stored locally. In [19], a simple meta-data structure is described. More research needs to be done in this area. The third topic is related to the late binding issue discussed in [2], [19]. In traditional well-connected networks such as the wired Internet or cellular networks, destination names are typically resolved at the source to routable identifiers e.g. IP addresses. This is typically facilitated by a quasi-static DNS hierarchy. In networks subject to disruption, such information to map destination names to routable identifiers may not be readily available at the source and the nodes that carry such information may not be reachable by the source. Thus, in DTNs, late binding is proposed to defer the name resolution as late as possible. In [19], the authors propose

using a declarative logic language in addition to attaching a meta-data extension block to carry information for the name resolution. Again, more work needs to be done to ensure that researchers agree upon a common language for describing physical objects e.g. sensor nodes in a certain location where information is stored so that such information can be retrieved easily without requiring the querying node to know the identifiers of these objects.

VI. CONCLUDING REMARKS

In this paper, we have presented the design of an information retrieval system for disruption tolerant networks. We focus mainly on using data and query replications to improve the query performance. We also discuss how nodes should be selected to cache the replicated data copies. In our information retrieval system, we show that the intelligent caching scheme provides better query performance when compared to the random caching scheme. In addition, we show that query replication is effective in improving the query performance. Both query replication schemes, namely the W-copy selection spraying (WSS) scheme and the L-hop neighborhood query spraying (LNS) scheme provides higher query success ratio. The WSS scheme performs better than the LNS scheme because it uses binary spread method to duplicate queries to nodes that have higher chances of meeting other nodes. The WSS scheme can provide 45% to 460% improvement in the query success rate when compared to that achieved using the NDD scheme. The query performance is greatly affected by the mobility models. Our results using different mobility models reveal that the query success ratio degrades by at most 7.3% when the community model is used compared to the RWP model. The UmassBusNet model achieves a much lower query success ratio when compared to the RWP and community models. The poorer performance achieved by the UmassBusNet model is due to its long inter-node encounter time. Our investigations also reveal that the proportional and square-root replication schemes achieve higher query success ratios than the fixed copy approach.

In this paper, we only consider three mobility models. We are interested in studying other mobility models e.g. those built using traces obtained from community networks [23] or other vehicular ad hoc networks. In this work, we also assume that data items have a fixed expiration time. In real life scenarios e.g. battlefields, the expiration times of the data items may not be known ahead of time so mechanisms to invalidate old data need to be designed such that queries will not be answered using stale information. One possibility is to spread multiple copies of data invalidation packets such that nodes that carry stale information will delete expired data items upon receiving the data invalidation packets. In addition, we intend to explore how indices can be designed to allow for efficient data advertisement and retrievals since data retrieval schemes have to be implemented on small mobile devices like PDAs. Data centric security design needs to be done to ensure that only legitimate nodes are allowed to access and retrieve data in an information retrieval system operating in challenging network environments. Last but not least, we intend to build a medium size testbed that demonstrates our information retrieval schemes. Preliminary testbed results are included in Appendix 3. Detailed description of the implementation effort will be reported in a technical report.

ACKNOWLEDGMENT

This work has been supported by DARPA under Contract W15P7T-06-C-P430. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the sponsor of this work.

REFERENCES

[1] K. Fall, " A delay tolerant network architecture for challenged networks", Proceedings of ACM Sigcomm, 2003..
[2] V. Cerf et al, "Delay Tolerant Networking Architecture", IETF RFC 4838, Informational, June, 2007
[3] A. Lingren et al, "Probabilistic Routing in Intermittently Connected Networks", Proceedings of Workshop on Service Assurance with Partial and Intermittent Resources, Aug, 2004.
[4] J. Burgess et al, "MaxProp: Routing for vehicle-based disruption tolerant networks", Proceedings of IEEE Infocom, 2006.

[5] M. M. B Tariq, M. Ammar, E. Zegura, "Message Ferry Route Design for Sparse Ad Hoc Networks with Mobile Nodes", ACM Mobihoc, May, 2006.
[6] M. Chuah, P. Yang, "Node Density-Based Adaptive Routing Scheme for Disruption Tolerant Networks", Proceedings of IEEE Milcom, 2006.
[7] B. Sheng et al, "Data Storage Placement in Sensor Networks", Proceedings of ACM Mobihoc, May, 2006.
[8] S. Jin, L. Wang, "Content and Service Replication Strategies in Multihop Wireless Mesh Networks", Proceedings of MSWiM, Oct, 2005.
[9] L. Yin, G. Cao, "Supporting Cooperative Caching in Ad hoc Networks", Proceedings of IEEE Infocom, 2004.
[10] A. Vahdat, D. Becker, "Epidemic Routing for partially connected ad hoc networks", Technical Report CS-200006, Duke University, April, 2000
[11] T. Spyropoulos et al, "Efficient routing in intermittently connected mobile networks: multiple copy case", IEEE/ACM Transactions on Networking, Feb, 2008.
[12] "The network simulator ns-2", [Online] at http://www.isi.edu/nsnam/ns/.
[13] J. Broch et al, "A Performance Comparison of Multhop wireless Ad hoc Network Routing Protocols", ACM Mobicom, pp 85-97, Oct, 1998.
[14] .X. Zhang et al, "Modeling of a bus-based disruption tolerant network trace", Proceedings of ACM Mobihoc, Sept, 2007.
[15] L. Breslau et al, "Web Caching and Zipf-like Distributions: Evidence and Implications", IEEE Infocom, 1999.
[16] T. Hara, "Effective Replica Allocation in Ad hoc Networks for Improving Data Accessibility", Proceedings of IEEE Infocom, 2001.
[17] M. Chuah, P. Yang, "Performance Evaluations of content-based information retrieval schemes for Disruption Tolerant Networks", Invited Paper, IEEE Milcom, 2007.
[18] M. Stillerman, "Hotdiffusion: peer to peer tactical information management platform", DARPA DTN Phase II Kickoff meeting, Aug, 2006.
[19] R. Krishnan et al, "The SPINDLE Disruption-Tolerant Networking System", Proceedings of IEEE Milcom, pg 1-7, Oct, 2007
[20] B. T. Loo et al, "Declarative routing: extensible routing with declarative queries", Proceedings of ACM Sigcomm, Philadephia, PA, pp 289-300, Aug, 2005.
[21] E. Cohen, S. Shenker, "Replication strategies in unstructured peer-to-peer networks", Proceedings of ACM Sigcomm, Pittsburg, PA, pp 177-190, Aug, 2002.
[22] M. Chuah, "Secure Data Retrieval in DTNs", CSE Technical Report, Dec, 2007
[23] MIT Media Lab: Reality Mining. http://reality.media.mit.edu
[24] J. Cao, Y. Zhang, G. Cao, L. Xie, "Data Consistency for Cooperative Caching in Mobile Environments", IEEE Computer,Vol 40, Issue 4, pp 60-66, April, 2007.
[25] Y. Wang, S. Jan, M. Martonosi, and K. Fall, "Erasure-Coding Based Routing for Opportunistic Networks", Proceedings of ACM Sigcomm WDTN Workshop, August 2005.
[26] S. Jain, M. Demmer, R. Patra, K. Fall, "Using Redundancy to cope with Failures in a Delay Tolerant Network", Proceedings of ACM Sigcomm, August, 2005.
[27] Z. Zhang, Q. Zhang, "Delay/disruption tolerant mobile ad hoc networks: latest developments", Wireless Communications & Mobile Computing Journal, Vol 7, Issue 10, pp 1219-1232, Dec, 2007.
[28] A. Balasubramanian, B. Levine, A. Venkataramani, "DTN Routing as a Resource Allocation Problem", Proceedings of ACM Sigcomm 2007, Japan, Aug, 2007.
[29] H. Wang, C. C. Tan, Q. Li, "Snoogle: A Search Engine for the Physical World", Proceedings of IEEE Infocom 2008, April 2008.

Appendix 1

Data Dissemination

```
initialize(BROADCAST_TIMER);
activate(BROADCAST_TIMER);
initialize(query_list);
initialize(loal_cache);
Upon generation of data_item do
   item.originator=local_identifier;
   item.num_dissemination=0;
   item.payload=data_item.payload;
   local_cache.add(item);

Upon expiration of BROACAST_TIMER do
   BroadcastDataItemInfo();

Upon reception of BCp from node N do
   for each item i in BCp.item_list
      if i is in the local_cache or neighbors_items_list then
         break;
      else
```

```
          insert <i, N> tuple into neighbors_items_list;
          for each query in query_list
             if(query.item_id==i)
                fetch item i from node N;
                replyItem(query.originator, item i);
        endif;
   end // for loop
    dataDissemination();
    ifdef WSS
      if BCp.FM > FM_Threshold then
              DisseminateQuery(N);
      endif
    endifdef
Upon reception of data dissemination packet DDp do
   local_cache.add(DDp.item);

PROCEDURE BroadcastDataItemInfo()
   BCp ←composeBcastPkt();
   broadcast(BCp);
   activate(BROADCAST_TIMER);
PROCEDURE composeBcastPkt()
   create a broadcast packet BCp;
   append own identifier in BCp;
   append identifiers of data items in local cache in BCp.item_list;
   append identifiers of data items in 1-hop neighbors in BCp.item_list;

   ifdef WSS
   BCp.FM=my own FM value;
   endif
   return BCp;

PROCEDURE dataDissemination(N)
   for each data_item in local_cache
      if data_item.originator=local_identifier &&
                  data_item.num_copies <MAXCOPY &&
                  N not in data_item.dissemination_list then

          ifdef Intelligent_Caching
            if FM(N)> FM_Thresh then
              data_item.dissemination_list.add(N)
                        unicastDataItem(data_item, N);
           endif
           endif
             ifdef Random_Caching
              data_item.dissemination_list.add(N)
                        unicastDataItem(data_item, N);
          endif
PROCEDURE unicastDataItem(data_item, destination)
   create a packet DDp;
   DDp.destination =destination;
   DDp.item=data_item;
   send out DDp to destination;
```

Pseudo Code for the maintenance of 2-hop neighbors information

```
Upon reception of BCp from node N do
    update 1HopNeighborList using N as input
    for each id in BCp.neighbor_list
        if id in 1HopNeighborList then
            continue;
        else
            update 2HopNeighborList using <id, N> as input
PROCEDURE composeBcastPkt()
        create a broadcast packet BCp;
        append own identifier in BCp;
        append 1HopNeighborList to BCp.neighbor_list;
        append identifiers of data items in local cache in BCp.item_list;
        append identifiers of data items in 1-hop neighbors in BCp.item_list;
```

Query dissemination and Query response (LNS)

```
initialize QUERY_SEQ ← 0;
Upon reception of a self-generated query for item i do
   query= composeQuery(my_id, creation_time, i, QUERY_SEQ++);
   query_list.add(query);
   if item i is in local_cache then
      query_list.remove(query);
      reply the item i;
   else
      next_hop= neighbors_items_list.find(i);
      if next_hop is not empty then
         fetch item i;
         replyItem(query.originator,item i);
            else
               broadcastQuery (query,ttl);
Upon the reception of a query packet Qp do
   if Qp.query.item_id in (local_cache || neighbors_items_list) then
      route item Qp.query.item_id to the query originator Qp.query.originator
            through underlying routing;
      else
            query_list.add(Qp.query);
   if Qp.ttl >0 then
      rebroadcastQuery (Qp);

PROCEDURE broadcastQuery(query,ttl)
   create a query packet Qp;
   Qp.query=query;
   Qp.ttl=ttl;
   broadcast Qp;

PROCEDURE rebroadcastQuery(Qp)
   Qp.ttl=Qp.ttl-1;
   broadcast Qp;

PROCEDURE composeQuery(originator, creation_time, item_id, seq)
            query← <originator, creation_time, item_id, seq>;
```

```
        return query;

PROCEDURE replyItem(query_originator, item)
   create a packet Rp;
   Rp.destination=query_originator;
   append item into Rp;
   route Rp through underlying routing;
```

Query dissemination using WSS

```
Initialize QUERY_SEQ ← 0;
Upon reception of a self-generated query for item i do
   query= composeQuery(my_id, creation_time, i, QUERY_SEQ++);
   if item i is in local_cache then
      reply the item i;
   else
      next_hop= neighbors_items_list.find(i);
      if next_hop is not empty then
         fetch item i through next_hop;
         replyItem(query.originator,item i);
      else
      query_list.add(query);
      endif
    endif

Upon the reception of a query packet Qp do
   if Qp.query.item_id in (local_cache || neighbors_items_list) then
      route the retrieved data item to the query originator Qp.query.originator
            through underlying routing;
      else
            query_list.add(Qp.query);

PROCEDURE DisseminateQuery(N)
   for each query in query_list
      if query.originator==local_identifier and query.num_copies < MAX_QCOPY
         unicastQuery(query, N);

PROCEDURE unicastQuery(query, destination)
   Qp.query=query;
   Qp.destination=destination;
   send out Qp;

PROCEDURE composeQuery(originator, creation_time, item_id, seq)
            query← <originator, creation_time, item_id, seq>;
            return query;

PROCEDURE replyItem(query_originator, item)
   create a query response packet Rp;
   Rp.destination=query_originator;
   append item into Rp;
   route Rp through underlying routing;
```

Appendix 2

Here, we present the results that compare the query performance of the WSS and LNS schemes when the CB model is used. The network scenario used is the one with 40 nodes distributed over the 3000x3000 m^2 and the nodes move according to the CB model. The data/query expiration time is varied from 750s to 2000s. The query generation rate is set at 1 query/s. The data replication factor is set to 5 while W is set to 5 for the WSS

scheme, and L is set to 4 for the LNS scheme. The values for the query replication factor are chosen such that the number of query replication is the same for both schemes. Table 4 tabulates the results we obtain for both the WSS and LNS schemes. The results indicate that the WSS scheme gives better delivery ratio than the LNS scheme. *The lower average response time for the LNS scheme is misleading. When we consider only those queries that are successful in both schemes, the average query response times for both schemes are the same.* Those queries that are not successful in the LNS scheme but successful in the WSS scheme take longer to complete, and thus result in slightly higher average query response time seen for the WSS scheme.

Exp Time	750	1000	1500	2000
WSS	0.43	0.54	0.73	0.79
LNS	0.39	0.50	0.70	0.75

(a) Delivery Ratio

Exp Time	750	1000	1500	2000
WSS	71	59	38	26
LNS	55	50	32	23

(b) Avg Query Response Time

Exp Time	750	1000	1500	2000
WSS	0.23	0.23	0.24	0.25
LNS	0.25	0.26	0.27	0.27

(c) Data Efficiency

Table 4: Query Performance Comparison between the WSS and LNS schemes for the CB model.

Appendix 3

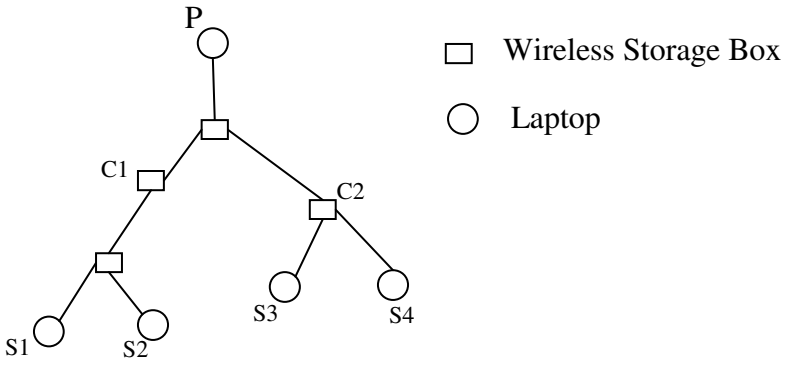
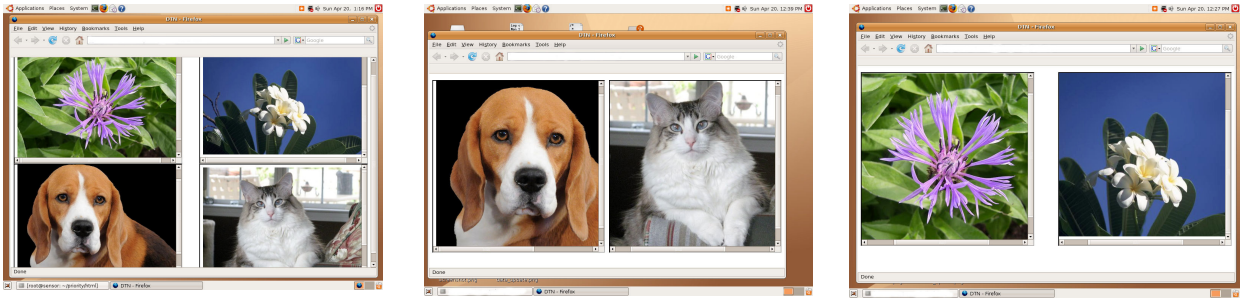


Fig. 9: Topology of the 9-node testbed.

A testbed that consists of 9 nodes as shown in Figure 9 has been set up. 4 nodes are wireless routers built by BBN for the DARPA DTN project. Two of the wireless routers are used as storage nodes while the other two are merely used for traffic forwarding. The rest of the 5 nodes are laptops. These nodes are connected via wireless links and run a modified version of the reference DTN2 software (http://www.dtnrg.org) that we develop to support multicast and information retrieval features. There is one publisher (node P), two storage nodes (node C1 and C2) and four subscribers(node S1, S2, S3 and S4) in the 9-node testbed. In this experiment, we let the publisher (node P) publish two categories of pictures, namely animal pictures and flower pictures. Subscriber S1 and S3 subscribe animal pictures with C1 and C2 respectively while subscriber S2 and S4 subscribe flower pictures with custody C1 and C2. The snapshot of data at C1, S1 and S2 are show in figure 10(a) 10(b) and 10(c) respectively.



(a): pictures at node C1 (b): pictures at node S1 (c): pictures at node S2

Figure 10: Pictures that are stored in storage node and retrieved by subscribers.