Performance Evaluation of Content-Based Information Retrieval Schemes for DTNs

M. Chuah, P. Yang
Department of Computer Science & Engineering
Lehigh University
chuah@cse.lehigh.edu, pey204@lehigh.edu

Abstract*—Mobile nodes in some challenging network scenarios suffer from intermittent connectivity and frequent partitions e.g. battlefield and disaster recovery scenarios. Disruption Tolerant Network (DTN) technologies are designed to enable nodes in such environments to communicate with one another. Several DTN routing schemes have been proposed. However, not much work has been done on providing information access in such challenging network scenarios. Existing client/server paradigm for information access will not be feasible in such scenarios since end-to-end path does not exist. Thus, 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 should data be replicated and stored at multiple nodes, (b) how should a query be disseminated in sparsely connected networks, (c) how should a query response be routed back to the querying node. We first describe two data caching schemes: (a) K-copy random caching, (b) K-copy intelligent caching. Then, we describe an L-hop Neighborhood Spraying (LNS) scheme for query dissemination. For message routing, we either use Prophet routing scheme or Highest Encounter First Routing (HEFR) scheme. We conduct extensive simulation studies to evaluate different combinations of these algorithms. Our results reveal that the scheme that performs the best is the one that uses the K-copy intelligent caching combined with the LNS query dissemination and HEFR scheme.

***Index Terms*—disruption tolerant networking, content-based routing, network performance.**

I. Introduction

With the advancement in technology, we have many wireless computing devices e.g. PDAs, sensors etc. Such devices can form infrastructureless adhoc networks and communicate with one another via the help of intermediate nodes. Such adhoc networks are very useful in several scenarios e.g. battlefield operations, vehicular adhoc networks and disaster response scenarios. Many adhoc routing schemes have been designed for adhoc networks but such routing schemes are not useful in some challenging network scenarios where the nodes have intermittent connectivity and suffer frequent partitioning. Recently, disruption tolerant network technologies [1],[2] have been proposed to allow nodes in such extreme networking environment to communicate with one another. 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 interests to multiple soldiers so it makes sense to replicate the data item, and store it 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 adhoc 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 approaches are not directly applicable 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 adhoc networks, namely CacheData, CachePath and HybridCache. However, their techniques are only useful for well-connected adhoc networks.

Thus, in this paper, we design a content-based information retrieval system for disruption tolerant networks. Our design consists of three main components, (a) data caching, (b) query dissemination, and (c) message routing. Security design for such a system is also very important but we do not address this issue in this paper. For data caching, we explore two data caching schemes, namely (i) K-copy random caching, and (b) K-copy intelligent caching. For query dissemination, we explore a L-hop neighborhood query spraying (LNS) scheme. For message routing, we either use Prophet [3] or Highest Encounter First Routing (HEFR) scheme. We conduct extensive simulation studies to evaluate different combinations of these algorithms. Our simulation results reveal that the scheme that performs the best is the one that uses K-copy intelligent caching combined with the LNS query spraying, and the HEFR scheme.

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 dissemination system for DTNs. In Section IV, we present the performance evaluation of the proposed schemes. In Section V, we provide our concluding remarks and discuss future research.

II. RELATED WORK

In [15], the author considered the data replica allocation problem in adhoc networks. More specifically, the author proposed three replica allocation methods, namely (a) Static Access Frequency (SAF), (b) dynamic access frequency and neighborhood (DAFN) method, and (c) dynamic connectivity based group (DCG). In SAF, each mobile host caches the 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 group access frequency. The disadvantage of this approach is they assume full knowledge of access frequencies and ability to share such information in well-connected ad hoc networks.

In [7], the authors considers 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 fixed tree model, they give an exact solution on how to place storage nodes to minimize total energy cost. For the dynamic tree model, they allow each sensor node to select the storage node for storage with respect to the minimal communication cost for data forwarding and query diffusion and reply once the storage nodes have been positioned.

In [8], the authors considered optimal replication strategy for large 2-D wireless mesh networks. They found that to minimize the object access cost, the number of replicas an object 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 lese the path to the data will be cached. LRU policy is used for cache replacement. These three approaches however do not take into considerations how nodes within a neighborhood can collaborate such that the data caching will always generate better benefits in terms of access cost. In addition, their approach also assumes well-connected adhoc networks.

III. CONTENT-BASED INFORMATION DISSEMINATION

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 supports both push and pull mechanisms that work in disruption tolerant network environment. Thus, contents can be searched and retrieved in our system even when connectivity is disrupted. Using the late-binding [2], and content-based routing features in our system, queries can be issued without prior knowledge of where the content lies.

A. Data Caching Schemes

For the push mechanism, we investigate two data caching schemes for DTNs, namely (a) K-copy random caching, and (b) K-copy intelligent caching.

1) K-copy Random Caching

For K-copy random caching, each node, j, will pass a copy of the data item to a node that j encounters next. Such an action is repeated K times so K nodes will store this data item.

2) K-copy Intelligent Caching

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 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). Since the nodes may move with different maximum speed, the nodes that move faster can encounter more nodes and hence are good candidates for relaying or the storage nodes for messages. Thus, we propose having each node measures 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 encountered. Let T be the observation window interval, t_n =[(n-1)T, nT] be the nth observation window, S_i (n) be the number of unique nodes observed by node i, and $\widetilde{S}_i$ be the FM of node i. Then,

$$\widetilde{S}_i = \partial\ S_i(n) + (1-\partial)\widetilde{S}_i \text{ ----------- Eq(1)}$$

Periodically, each node issues beacons and include the FM value in its beacon messages. Thus, for the K-copy intelligent caching, each source node will select a node with a FM value that exceeds FM_threshold to be a storage node. The source node repeats this action K times for each data item that it generates.

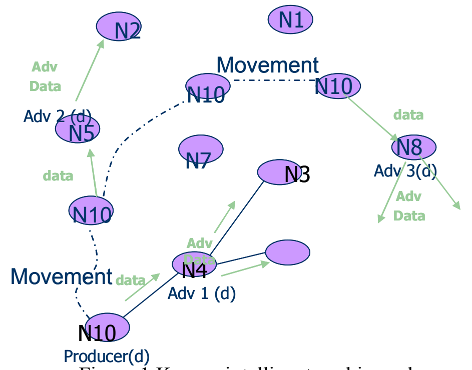

Figure 1 K-copy intelligent caching scheme

Figure 1 illustrates the K-copy intelligent caching scheme. K is set to 3 in the picture. Let us assume that Node N10 is the generator of a data item, *d*. Thus, N10 needs to select three storage nodes for the data item, *d*. N10 first selects the first node it encounters, N4, to store the data since the FM value of N4 exceeds the FM_threshold. Then, 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 but since their FM values are below 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, N10 selects this node to be the last storage node for the data item, *d*. N4, N5, and N8 will then include description of the data item *d* in their beacons that are broadcast periodically.

B. *Query Dissemination Schemes*

As indicated earlier, periodically each node broadcasts a beacon message. Several pieces of information are included in the beacon message, namely (a) a list of data items that are currently stored in this particular node, (b) a list of data items that are within its L-hop neighborhood of this node

In this paper, we explore a L-hop local neighborhood query spraying (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 has the requested data item, it will immediately generate a query response.

In Figure 2, we illustrate the LNS approach. N8 generates a query and broadcasts it with a TTL of L=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.

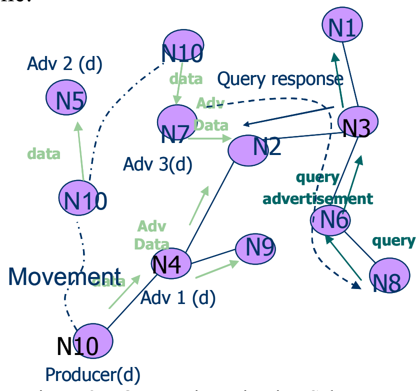

Figure 2: Query Dissemination Schemes

Note that since a query is stored at multiple nodes, 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.

C. *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 two DTN routing scheme, namely (a) Prophet [3], and (b) Highest Encountered First Routing (HEFR) scheme.

1) *Prophet*

Prophet uses the history of encounters and transitivity to route messages for intermittently connected networks. In this scheme, each node broadcasts a beacon periodically. This probabilistic routing scheme establishes a probabilistic metric called delivery predictability at every node A 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. The delivery predictability ages with time and also has a transitive property, i.e., a node A that encounters node B which encounters node C allows node A to update its delivery predictability to node C based on its (A's) delivery predictability to node B and node B's delivery predictability to node C. 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:

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

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

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

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 pass it to another node which has the highest delivery predictability to the destination.

2) Highest Encountered First Routing (HEFR) scheme

In this scheme, each node also broadcasts beacons periodically. Each node maintains a friendliness metric (FM) as described in Section III.A. When a node receives a message that needs to be forwarded, the node will select a node that has the highest FM value that exceeds a threshold (referred to as the FM threshold) to be the next hop node. If there is no next hop node with FM value that exceeds the FM threshold, the message will be retained until such a node can be found.

D. Implementation Details

1) Information contained in the beacon

Periodically, a node broadcasts a beacon. The interval between two beacons broadcasted is referred to as the beacon interval. The beacon message includes the following information: (a) its friendliness metric (FM), (b) a list of the descriptions of data items (refer to as meta-data) cached locally, and (c) a list of data items that are cached within its M-hop neighborhood,

2) Information contained in the data items, queries

Each stored data item contains information about the data item identifier, the source node identity, the timestamp when the data item is generated, and the data item expiry time. If the data item contains sensitive information, the data item can be encrypted either using a data category key or a key that only the source node knows. Each stored query contains information about the identity of the node which issues the query, and the query expiration time (QET).

3) Maintenance of Cached Data Item List

A node maintains two lists of cached data items: a list that contains all data items that are stored locally, the other contains all data items that a node is aware of but stored within its M-hop neighborhood (M is set to 2 in this paper).

When a node receives a query or when the stored data list runs out of space, then a node will search through its locally cached data items and remove those that have expired. Those data items that have not expired are referred to as active data items. When a node receives new data items and cannot find expired data items to purge, then a node will remove those active data items with the earliest expiry time (even though such data items have not expired) when the node runs out of storage space.

4) Maintenance of Cached Queries List (CQL)

A node that receives a query message from another node and cannot find the requested data item will insert the received query message into a Cached Query List (CQL). The Cached Query List can be maintained as an ordered list of queries according to the query expiration time. While inserting new queries, the node will remove those cached queries that have expired.

5) Processing of Received Beacons

Whenever a node receives a beacon from another node, it will increment an encounter counter. This counter is used to measure the number of unique nodes that a node heard within a beacon interval. A node updates its FM value based on this counter value every beacon interval.

When a node receives a beacon, it will check whether the advertised data items from the other node matches with the items requested by the stored queries. If there is a match, the node will request for the data items and then generate a query response to the querying node. Both nodes will cache the retrieved data item if there is available storage space.

IV. PERFORMANCE EVALUTION

A. Simulation Setup

To investigate the usefulness of different combinations of the design, we implement the different data caching, query spraying and message routing schemes in NS-2 version 2.29 [12]. We also implement the CacheData scheme [9] as follows: each node will monitor the queries that it has forwarded, and set a flag if it receives more than K=2 queries from different nodes for the same data item. If that flag is set and the same node receives a query response that contains that data item, the node will cache the data item.

We assume that the wireless bandwidth is 2 Mb/s and the transmission range is 250 m. We use a network that consists of 20 nodes randomly distributed over (a) 700x700m^2 , (b) 1400x1400m^2 (default), (c) 2100x2100m^2 , and (d) 2800x2800 m^2. Note that scenario (a) is a well-connected scenario where each node on the average has 8 neighbors. The default scenario is one where each node has two neighbors on the average.

Node Movement Model: The nodes move either according to the random waypoint (RWP) mobility model [13] or the Zebranet [14] model. For 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 0, and vmax is set to 5 m/s for all nodes in the homogenous model (referred to as RWP1). For the heterogeneous model, we let 50% of the nodes select speed from (0, 5) m/s and 50% of the nodes select speed from (5,10) m/s. For the Zebranet model [14], we create a semi-synthentic model as follows: we synthesize node speed and turn angle distributions from the observed data and create other node-movements using the same distribution. We use both distance and time scaling to fit the original data found in the trace into the network environment that we are interested in.

Data Item Generation Model: 10 nodes generate data items, each of them generates two data items periodically. The data item interval is set to 500 seconds. Each data item has a certain expiry time. The data item expiry time is set to 500 seconds to ensure that there are only 20 data items throughout the whole simulation time. We assume that the data size is fixed.

Query Model: The query model is similar to what have been studied in the past [8][9]. 50% of the nodes generate read-only queries periodically. The query interval, T_{query} is varied from 2 seconds to 1000 seconds. The query expiration time is set to 500s. The access pattern is based on Zipf-like distribution which has been frequently used [15] to model non-uniform access pattern. In the Zipf-like distribution, the access probability of the ith $(1 \le i \le N)$ data item is represented as follows:

$$P_i = \frac{1}{i^\theta \sum_{k=1}^{n} \frac{1}{k^\theta}} \text{ ------- Eq (3)}$$

where $1 \le \theta \le 1)$. When $\theta = 1$ (the default value we use), it follows the strict Zipf distribution. θ=0.8 has been shown to generate access pattern similar to those real web traces [15]. In this paper, we only report results we get when $\theta = 1$. Results related to the impact of θ on the performance will be reported in a technical report [17]. Each node has a buffer size of 20 for stored queries.

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 that issues a query.
- Data efficiency – this is measured as the number of useful data bytes over the number of total transmitted data bytes (does not include control overhead).

Each data point reported in the simulation results section is the average of 5 runs. Each simulation runs until at least 400 successful queries are obtained.

B. Simulation Results

1) Impact of Data Caching Schemes

In our first set of experiment, we would like to investigate whether it makes a big difference in terms of which nodes to select for caching data items. We use the network scenario where 20 nodes are distributed over 1400x1400m^2 . We use the random waypoint mobility model where all nodes have a maximum speed of 5 m/s We set K=10 for random and intelligent caching. For this experiment, we use four combinations of different schemes, namely (a) NQ-HEFR-RC which stands for no query dissemination, using HEFR scheme for message routing and random caching scheme, (b) NQ-HEFR-IC which is the same as (a) except the intelligent caching scheme is used, (c) LNS-HEFR-RC which is the same as (a) except that the local neighborhood query spraying scheme is used to disseminate queries, (d) LNS-HEFR-IC which is the same as (c) except that we use intelligent caching.. When there is no query dissemination, each querying node can determine if its requested data item can be found within its 2-hop neighborhood. If not, it will cache the query and periodically check its 2-hop neighborhood for the data item. We vary the query interval from 100 second to 1000 seconds. Tables 1(a),1(b), and 1(c) tabulate the average query success ratio, the average query response time (for successful queries), and the data efficiency obtained in this set of experiments.

In this default network scenario, the average node encounter time is about 770s but the query expiry time is 500s. Thus, without query dissemination, if a querying node cannot find its requested data item within two hops, then its query will likely expire. When the query load is very low (one query every 1000s), a total of 5 queries are generated every 500s, and the cost of replicating 20 data items with K=10 is 20x10=200 transmissions. Each query, on the average, requires about 4 transmissions (2 hops in the forward/reverse path) so the data efficiency can only be 5/(5*4+200)=0.022. Better performance can be achieved via intelligent caching and query dissemination. Even though intelligent caching improves on the query success ratio, the improvement is small compared to what can be achieved when the LNS query dissemination scheme is used. The use of intelligent caching also reduces the average query response time slightly. Such improvement comes without incurring extra cost in terms of data efficiency. So, for the rest of the paper, we will use random caching when Prophet is used and use intelligent caching when HEFR combination is used since when the Prophet scheme is used, FM metric will not be collected.

Table 1(a): Query Success Ratio

Query interval	NQ HEFR-random caching	NQ HEFR-intelligent caching	LNS HEFR random caching	LNS HEFR intelligent caching
100	0.610	0.616	0.624	0.637
250	0.637	0.651	0.664	0.678
500	0.629	0.646	0.709	0.721
1000	0.701	0.710	0.777	0.791

Table 1(b): Average Query Response Time

Query interval	NQ HEFR-random caching	NQ HEFR-intelligent caching	LNS HEFR random caching	LNS HEFR intelligent caching
100	42.5	40.5	43.5	41.6
250	45.1	42.6	55.2	53.4
500	82.3	44.0	79.2	75.9
1000	125.3	54.6	120.3	112.8

Table 1(c): Average Data Efficiency

Query interval	NQ HEFR-random caching	NQ HEFR-intelligent caching	LNS HEFR random caching	LNS HEFR intelligent caching
100	0.189	0.187	0.198	0.195
250	0.075	0.073	0.078	0.076
500	0.037	0.037	0.041	0.041
1000	0.019	0.019	0.022	0.022

2) Impact of Query Interval

In our second experiment, we investigate on the performance impact with/without query dissemination, with random or intelligent caching and with Prophet or HEFR schemes as we vary the query interval. We use the same network scenario as in the previous section and also set K=10. We use four different combinations, namely (a) NQ-Prophet-RC, (b) NQ-HERF-IC, (c) LNS-Prophet-RC, (d) LNS-HEFR-IC and the CacheData scheme. We vary the query interval from 2 to 100 seconds. The results for average query success ratio, average query response time and average data efficiency are plotted in Figures 3(a), 3(b) & 3(c). We see that all four combinations outperform CacheData scheme in terms of query success rate, query response time and the data efficiency. As the query interval decreases, the query success rate for the four combinations first increases slightly and then it starts dropping. The initial increase with increasing query load is due to the fact that more queries can benefit from the cached data items. However, as the query load increases such that there is a buffer overflow of stored queries, then the query success rate will drop. The average query response time decreases as the query interval reduces since more queries can make use of the cached data items as query interval reduces. LNS-HEFR-IC and LNS-Prophet-RC schemes give slightly better query response time when the query load is very light. The data efficiency increases with decreasing query interval initially because more queries can benefit from the cached data items. When the query load increases beyond a certain threshold, the data efficiency drops since some transmissions are wasted in propagating queries to nodes that run out of buffer space for stored queries

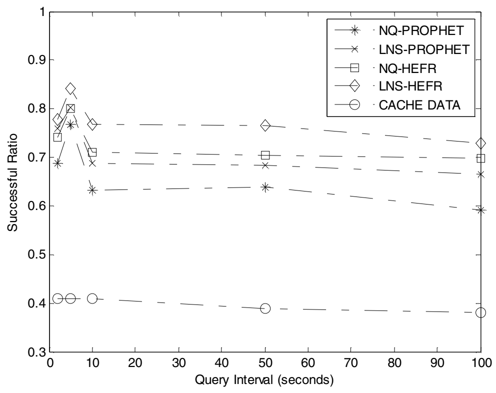

Figure 3(a) Average Query Success Ratio vs Query Interval.

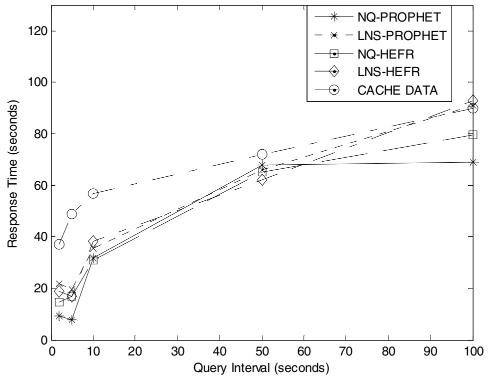

Figure3(b): Avg Query Response Time vs Query Interval.

3) *Impact of Node Density*

In our third set of experiment, we investigate the impact of node density when NQ-HEFR-IC, LNS-HEFR-IC, and CacheData schemes are used. The four node density we investigate are 20 nodes distributed over (a) 700x700m^2 , (b) 1400x1400 m^2 , (c) 2100x2100 m^2 , and (d) 2800x2800 m^2 . Query interval is set to 100s. Our results for the average query success rate, the average query response time and the average data efficiency are plotted in Figures 4(a), 4(b) and 4(c) respectively. The results indicate that LNS-HEFR provides the highest query success ratio for node density below 3.75x10^{-5}. CacheData has the highest query response time while LNS-HEFR and NQ-HEFR have comparable query response time. As node density decreases, the query success ratio drops while the average query response time increases because it takes longer time to disseminate query and retrieve data items.

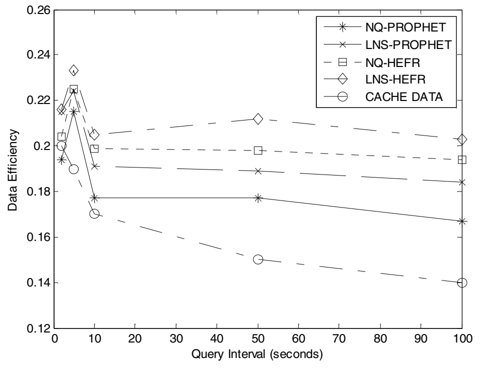

Figure 3(c): Average Data Efficiency vs Query Interval

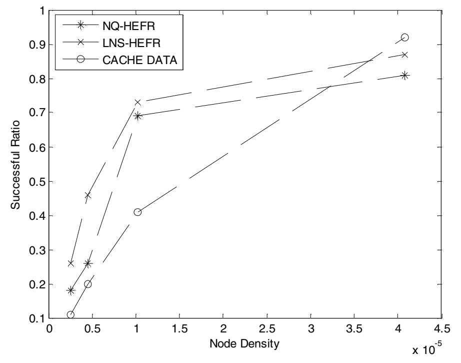

Figure 4(a): Avg Query Success Ratio vs Node Density

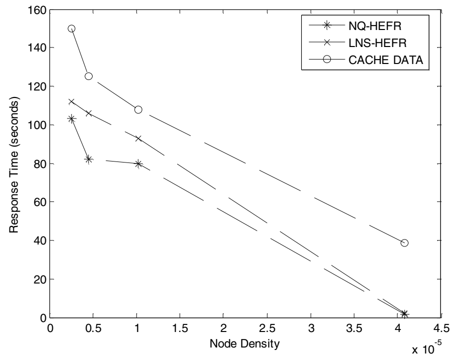

Figure 4(b): Avg Query Response Time vs Node Density

The data efficiency for LNS-HEFR and NQ-HEFR increases with decreasing node density. This is because we only consider the data efficiency of successful queries. With sparser network and very tight query expiry time of 500s, only those queries that can find data items close by are successful. Thus, the data efficiency increases with decreasing node density. We also see that CacheData provides higher data efficiency for high node density. This result suggests that perhaps we should adjust K according to the node density. For dense network, we need not replicate too many data items. We leave this for future work.

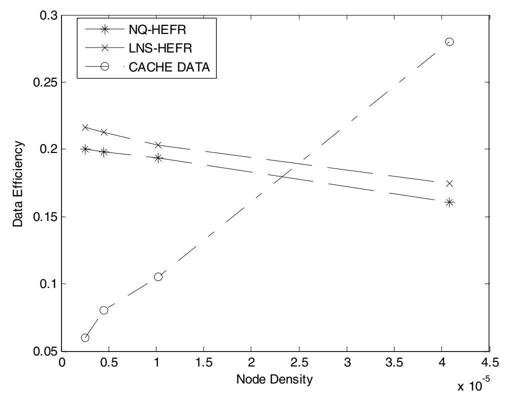

Figure 4 (c): Average Data Efficiency vs Node Density.

4) *Impact of Mobility Model*

In our fourth experiment, we investigate the impact of using different mobility models. We use a network scenario with 20 nodes distributed over 1400x1400 m^2 . We use three different mobility models: (a) RWP1 where all nodes move according to RWP and have the same maximum speed of 5 m/s, (b) RWP2 where all nodes move according to RWP with 50% having a maximum speed within the (0,5)m/s range and 50% having a maximum speed within the (5,10) m/s range, (c) ZebraNet model. The average speed for Zebranet is 8.5 m/s. 10 nodes generate queries with each node generating 1 query/100 sec. We use the NQ-HEFR-IC and LNS-HEFR-IC schemes. Our results are tabulated in Tables 2(a), 2(b) and 2(c).

	RWP1	RWP2	Zebra
NQ-HEFR-IC	0.698	0.788	0.803
LNS-HEFR-IC	0.729	0.793	0.811

Table 2(a): Average Query Success Rate

	RWP1	RWP2	Zebra
NQ-HEFR-IC	79.7	70.6	57.3
LNS-HEFR-IC	93	80.5	63.7

Table 2(b): Average Query Response Time

	RWP1	RWP2	Zebra
NQ-HEFR-IC	0.194	0.195	0.203
LNS-HEFR-IC	0.203	0.202	0.201

Table 2(c): Average Data Efficiency.

From the results, one can see that Zebranet achieves the highest query success rate followed by RWP2 and RWP1. The faster the nodes move, the higher the query success rate. The average response times for Zebranet and RWP2 is also smaller than RWP1. The average data efficiencies achieved for these three mobility models are similar. LNS-HEFR-IC scheme has slightly higher data efficiency.

V. CONCLUDING REMARKS

In this paper, we have presented a design of an information retrieval system for disruption tolerant networks. In our information retrieval system, we use K-copy random or intelligent caching and L-hop neighborhood query spraying to increase the average query success rate. In addition, we also propose using the HEFR scheme to achieve smaller query response time and hence achieve higher query success rate. Via simulations, we demonstrate that using LNS query spraying, HEFR routing scheme and intelligent caching, we can achieve the best performance in terms of higher query success rate and lower query response time with reasonable data efficiency.

In the near future, we are interested in exploring another query dissemination scheme called the W-copy query spraying scheme (WSS). Comparison results between WSS and LNS will be reported in [17]. We are also interested in exploring the impact of using the community mobility model on the performance of our information retrieval system. Last but not least, we hope to implement our scheme in a reasonable size testbed and evaluate its performance.

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. We thanked Jianbin Han for doing the CacheData simulation studies.

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”, draft-irtf-dtnrg-arch-08.txt, 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 Adhoc Networks”, Proceedings of IEEE Infocom, 2004. T. Spyropoulos et al, “Efficient routing in intermittently connected mobile networks: single copy case” to appear in IEEE/ACM Transactions on Networking, 2007.

[10] A. Vahdat, D. Becker, “Epidemic Routing for partially connected adhoc networks”, Technical Report CS-200006, Duke University, April, 2000

[11] T. Spyropoulos et al, “Efficient routing in intermittently connected mobile networks: multiple copy case” to appear in IEEE/ACM Transactions on Networking, 2007.

[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 Adhoc Network Routing Protocols”, ACM Mobicom, pp 85-97, Oct, 1998.

[14] Y. Wang et al, “Erasure-Coding Based Routing for Opportunistic Networks”, Proceedings of ACM workshop on WDTN, 2005.

[15] L. Breslau et al, “Web Caching and Zipf-like Distributions: Evidence and Implications”, IEEE Infocom, 1999.

[16] T. Hara, “Effective Replica Allocation in AdHoc Networks for Improving Data Accessibility”, Proceedings of IEEE Infocom, 2001.

[17] M. Chuah, P. Yang, ”Performance Evaluations of different content-based information retrieval schemes for Disruption Tolerant Networks”, Lehigh CSE Department Technical Report, April, 2007.