Distributed Storage in Disruption Tolerant Network

Jingzhe Du, Evangelos Kranakis
School of Computer Science
Carleton University
Ottawa, Canada
{jdu3, kranakis}@scs.carleton.ca

Amiya Nayak
School of Information Technology and Engineering
University of Ottawa
Ottawa, Canada
anayak@site.uottawa.ca

***Abstract*—We describe a novel Distributed Storage protocol in Disruption (Delay) Tolerant Networks (DTN). Since DTNs can not guarantee the connectivity of the network all the time, distributed data storage and look up has to be performed in a store-and-forward way. In this work, we define local distributed location regions which are called cells to facilitate the data storage and look up process. Nodes in a cell have high probability of moving within their cells. Our protocol resorts to storing data items in cells which have hierarchical structure to reduce routing information storage at nodes. Multiple copies of a data item may be stored at nodes to counter the adverse impact of the nature of DTNs. The cells are relatively stable regions and as a result, data exchange overheads among nodes are reduced. Through experimentation, we show that the proposed distributed storage protocol achieves higher successful data storage ratios with lower delays and limited data item exchange requirements than other protocols in the literature.**

***Index Terms*—Distributed Data Storage, Disruption (Delay) Tolerant Network, Local Algorithm.**

I. Introduction

Mobile Ad Hoc Networks (MANET) consist of autonomous mobile nodes connected by wireless channels without any pre-existing network infrastructure. Typically, some of these mobile devices are part of the network only while they can communicate with the rest of the network. Existing ad hoc distributed data storage protocols usually assume that the network is dense and there is always a connected path from message (data item) source to destination. In situations where network partitions exist, these protocols drop the message if a path could not be found and thus perform insufficiently in terms of data item delivery. Disruption (Delay) Tolerant Networks (DTN) are proposed to address such issues in MANET where instantaneous source and destination node connections may not exist. There are increased DTN applications in recent years, including military communications [13], inter-planetary networks [5], wildlife tracking [8] and intermittent Internet connection in under-developed countries (areas) [14], to name a few.

Geographic data storage has been studied in MANET. Nodes could get their location information either by global positioning system (GPS) or localization algorithms [18]. In existing geographic data storage schemes [16], [10], a node makes data storage decisions according to the mapping of a data item to a specific location. During the storage process, the node forwards this data item to the location according to its neighboring nodes' location information and the data item is stored at the node closest to the mapped location. The look up action is similar to this storage process. Since contemporaneous source to destination node connections may not exist in DTN, network disruptions have to be properly dealt with if geographic data storage is applied on the network. And since different DTN networks have different network characteristics (e.g., different node densities and movement patterns), a simple mapping approach can introduce too much overheads and thus is not the best choice in dealing with different situations.

A. Related Work

Peer-to-Peer (P2P) is a decentralized way of networking in which network participants have equal responsibilities and capabilities. Distributed Hash Table (DHT) based P2P systems [19], [20] are well known for their efficiency in the storing and searching of data items. In DHT, an object is mapped to an ID through a one way hash function. DHT has been widely used in distributed data storage and look up on the Internet, which is considered to be able to provide reliable and fast connections between any two nodes. With only neighboring connections among nodes, the distributed data storage and look up in MANET is different from the DHT substrate on the Internet. There are several protocols [10], [11], [15] dealing with distributed data storage in MANET. In [10], the authors proposed the idea of a mobile structured peer to peer network, called Mobile Hash Table (MHT) to facilitate data storage and look up. In MHT, every node is assumed to know its moving trajectory and a data item is mapped to a tuple (position, direction, speed). This data item is further stored at a node with the "closest matching" pattern. When this node moves out of the circle with radius half its communication range and with center a data item's matching position, the data item has to be moved to another node. As shown in Figure 1, a data item with mapping location d has to be stored in, e.g., node a within $\frac{r}{2}$ (r is communication range) of d such that another node b can find the data item when it comes within $\frac{r}{2}$ of the mapped location. In [11], the authors proposed MPI, another location based information sharing mechanism in MANET. In MPI, a data item is always kept by the owner and relevant look up information on the data item needs to be updated accordingly when the owner moves around. MPI uses equal sized lower level squares to hierarchically divide the upper

level squares. The lowest level squares are called grid cells, with side length $\frac{r}{\sqrt{2}}$ (r is node communication range). In this scheme, a node shares its data item with others by publishing its data item and location information, starting from its grid cell, level by level up until the top level. A data item look up process works first in a bottom up fashion, by searching lowest level up to a level where information of the data item can be found, then from this level sends data item request down to the lowest level location. Both MHT and MPI use GPSR [9] as the underlying routing protocol. In [15], the authors proposed a DHT substrate in MANET, which is a combination of DSR [7] and Pastry [17]. However, it can not scale well when the number of nodes increases. According to the mapping schemes in [10] and [11], existing location based mechanisms would not work properly when the area of a region is very large, the number of nodes is small or when the nodes are not uniformly distributed. MPI can not scale well and is fragile when lots of source nodes move around.

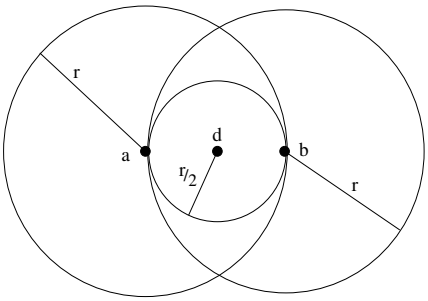

Fig. 1: MHT data item storage

B. Contributions and Organization of the Paper

In this paper, we propose Cell-based Hash (mapping) Table (CHT), a novel Distributed Storage scheme in DTN. CHT maps a data item to a region, called cell. Cells are defined in such a way that node movements inside cells are far more frequent than the node movements crossing cell borders. CHT mapping works in a hierarchical manner. When a node wants to store a data item, it first needs to map this data item to the highest layer (level) cell where the data item should be placed. This data item is mapped either layer by layer down to the lowest level cell using only local information, or to a certain layer cell which is different from the cell of source node where the mapping stops. Then the data item is routed towards the available different cell by using DTN routing protocols, e.g., GLR [6] algorithm. Once it reaches a node in that cell, this node can then further map it down to a lower layer. A data item is mapped and stored at a node in the lowest level cell. Data item look up request mapping works in the same way. Location diffusion is only performed in the lowest level cell or at most in several upper layers, no global location diffusion is needed to reduce the storage overhead.

We present the formal algorithm and compare it with MHT [10] (using DTN routing protocol instead of GPSR [9]) and show that it is advantageous in communication overheads, delays and data item storage success ratios.

The rest of the paper is organized as follows. Section II elaborates on our proposed solutions. Section III describes the details of experiments and analysis. Section IV concludes with possible future work.

II. Distributed Storage Algorithm

We propose the idea of using distributed peer-to-peer (P2P) solution to counter network disruption (delay), without relying on centralized network servers or super nodes.

Our proposed solutions use a mapping function $f(name) \rightarrow cell$ (we use the same symbol f in the following paper which may take different parameters) to facilitate distributed data item storage and look up. A data item first maps to a cell, then from a cell maps to a node in the cell. If a one to many mapping is necessary, the mapping cells (or nodes) are better distributed evenly to counter network delay and disruptions. The mapping function $f(name) \rightarrow cell$ is also required to map a node name to a cell.

A. Delay-Tolerant Distributed Storage and Look up

1) Network Partitioning: In the proposed solution, network regions are divided into cells (regions). Cells may be flat or with multiple layers and are not necessary to cover whole area (gaps are allowed if it is impossible that there will be nodes in these gaps). Nodes can define their own cells with variable size. The random sized area description can be stored in an approximate way to save the storage space. Region definition is a slow changing mechanism. Once the border of a region changes, this information is broadcasted to all other regions in the same layer, within the same upper region. A node does not need detailed global information to communicate with nodes in faraway places. In this way, local information (with very limited global information) is used to achieve global communication without relying on centralized nodes in the proposed solution. A node outside a region can use the mapping function $f(name) \rightarrow region$ to find the region of interest and a node inside this region can use mapping function $f(name) \rightarrow ID$ to locate the node which is responsible for storing the data item with corresponding name. As a special case, network may be divided in a way that cells can follow some specified shapes (e.g., hexagon, rectangle etc.), where the required storage space for area description is greatly reduced.

There are two ways for network partitioning. The first one makes use of existing or predicted network characteristics. If there are high probabilities that nodes will move within certain areas, then the network partitions can be made based on this information, using either a flat (one layer) or a hierarchical structure, depending on the network size. In DTN, there are works on model based [4] and history based [12] routing protocols. Model and history statistical information can also be used in cell formation process. The second way works autonomously. Nodes will try to formulate their cells depending on the network evolving process, by gradually accumulating the network statistical information, from local neighbors to

faraway nodes. When nodes tend to move within their specific areas (regions) with high probability, a cell based data storage scheme is more robust with less message exchange overhead. Only when a node moves out of a cell, will message exchanges be necessary.

2) Layering: Layered regions are necessary when there are too many nodes distributed in a vast area. In the layered cell approach, the mapping function is $f(t,\ name,\ parameter_1,\ parameter_2,\ ...,\ parameter_{n-1}) \rightarrow\ lowest\ \ layer\ \ cell$, where the network has n layers with $layer_1$ the top layer, t is time, $name$ is the data item (or node) name and $parameter_i$ is a layer specific parameter. The mapping function $f(t, name, parameter_i) \rightarrow layer_{i+1}$ means that by using layer specific parameter, a name could be mapped to a lower layer at time t. This mapping parameter may only need to be kept in $layer_i$, within the framework of the upper layer. The boundary of a region in $layer_i$ is specified in a way that less nodes movement across region boundary will happen, compared with nodes movement inside the corresponding region. In the proposed solution, the higher the layer, the less possible boundary would need change. It is apparent that most of the time lower layer local change would not affect faraway nodes in this way. Assume each layer cell has m subcells and the storage space for mapping related information (cell description) is the same, then a node only needs to store $n + m - 2$ items of mapping information, rather than m^{n-1} items of mapping related information.

3) CHT with DHT: The proposed solution of distributed storage which uses P2P mechanism is called Cell-based Hash Table (CHT). In CHT, a cell is divided in a single layer if the area and nodes number is small and multiple layers are adopted if nodes number and its corresponding areas are large. In CHT, cells are used for data storage. Inside the lowest level cell, a data item is mapped to a node (or nodes if a one to many mapping is used) with closest matching ID(s) in the cell. Some existing DHT protocols [17], [19] can be used if nodes move within their cells with high probability, with the consideration that accurate data item to node mapping may not be complete because of the nature of DTN. A node with a specific data item will handover the item to another node in the cell when it moves out (crossing lowest cell border). If there is high probability that nodes will move across cell borders, then a simple balanced storage mechanism (e.g., only balance the number of data items stored at nodes without using DHT) may be a good choice.

The combination of CHT with DHT limits the use of DHT overlay to a small scale within the lowest layer cell. If the probability of nodes moving within their cells is very high, we would suggest the use of exact DHT substrate (e.g., Chord protocol in [19]). To further improve routing efficiency, we propose the use of Mchord (modified chord) if the underlying DTN routing protocol is GLR. In our Mchord scheme, a data item is mapped to an ID and it is supposed to be stored at a node with closest ID to its own. If a node which stores the data item meets a neighboring node with a closer ID to the data item, it will give this data item to its neighbor. In Figure 2, when node 1 which has a data item with ID 26 meets another node 15, it will give this data item to node 15 because $|26 - 15| < |26 - 1|$. There are two modes in our approach, one is reactive and the other is proactive. In reactive mode, data item and look up request are only sent to a neighboring node when the neighbor is closer to the stored data item ID (or request ID). While in proactive mode, a data item or request is always routed to a node with closest ID according to the local knowledge of a node. Depending on the priority of the data item or request, the reactive or proactive mode can be selected.

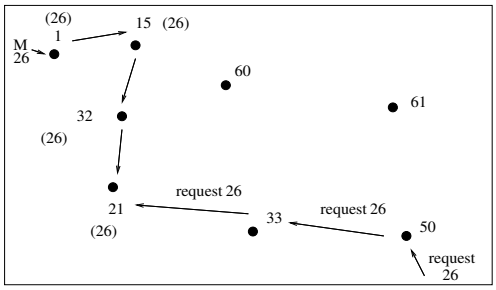

Fig. 2: Mchord storage and look up

4) Mapping: Two different rules exist for data items and nodes. A data item should stay at its mapping cell(s), while a node is allowed to move around. If nodes move faster, both CHT and DHT mapping should only be loosely coupled to reduce frequent message exchange overhead (due to cell changes of nodes). If nodes are relatively stable, mapping could be accurately coupled. For a node crossing multiple regions (large or small), it either can treat the mapping location as its home location and check regularly (when it requests a data item at that original location), or its ID can be mapped to more regions (large or small) and thus the reply for a request should be sent to multiple mapping regions. The extreme situation is one cell, then it is the same as traditional P2P.

5) Storage with Multiple Copies Option: We define the traffic hub (the point where most cell communication will pass through) in a cell as the cell center. When a data item (or request) message forwarding is necessary, this message will first be sent to a node in the cell with the shortest distance to the destination cell center. In Figure 3, when node $a1$ in a $layer_1$ cell with center $c1$ has a data item for another $layer_1$ cell with center $c2$, it needs to route the data item towards $c2$. However, once the data item enters into the $layer_1$ cell with center $c2$ and stores at node $a2$, $a2$ will decide the $layer_2$ cell with center c' where this data item should be placed. Similarly, a node $a3$ in $layer_2$ cell with center c' will decide the $layer_3$ cell, where the data item will finally be stored at a node d.

In CHT with Mchord, a data item can always be kept by a matching node so long as this node moves within the mapping cell. When a node stores a data item whose mapping ID is the same as its ID, there is no need to switch this data item with other nodes unless it leaves a cell. Even if a node stores a data item whose mapping ID does not equal to its ID, it

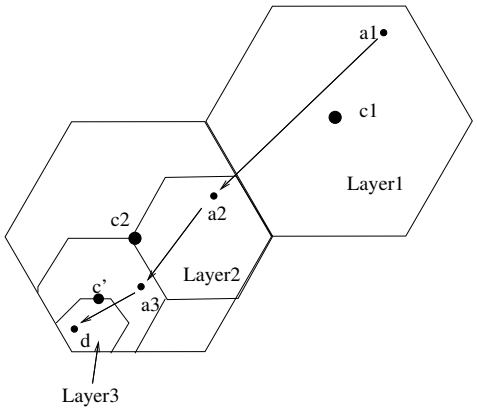

Fig. 3: Multiple layer storage

would not give this data item to other nodes unless another node with closer ID to the data item is found. When two nodes have the same distance (|node ID - data item mapping ID|), the node with smaller ID is chosen as the store node. This process converges to the node with the closest matching ID to the data item. In MHT however, a data item needs to be frequently exchanged among nodes because of the nature of mobility and its mapping mechanism. Due to the above reason, the maintenance overhead of CHT is less than that of MHT.

Multiple copies approach can be used in the proposed DTN P2P storage. In traditional P2P network, some protocols have proposed multiple copies approach. DKS(N, K, f) (f is replication factor) [3] and Tapestry [20] (Prefix routing) are two of them. The need of multiple copy approach is further necessary because of the characteristics of the DTN. The number of duplicate copies depends on the moving patterns of nodes.

6) Look Up: When a node performs data look up, it first maps the data item to a cell identifier (certain layer), and then use any possible connections to route its request towards this cell according to the look up mapping. The nodes in that certain layer further maps the data item look up into its sub-layer and forwards the look up. If mapping is loosely coupled, data item look up should be performed in the following order: first the mapping node in the cell, then any other node in the cell which has the data item and finally, nodes in surrounding cells should be checked. For nodes in a cell, if they perform P2P storage, their view of network may be incomplete. So counter measures in look up are necessary even if accurate mapping is adopted. A step by step option similar to the loosely coupled situation should also be used.

In DTN, data item look up uses the closest node to the cell to store request upon partition. It will be kept in a node that does not have a closer neighbor to the cell temporarily. Alternative ways (e.g., face routing) can also be used to send the look up request to the cell. In a certain region, it is possible that lots of nodes will move out. A few other nodes will have to store all the data items temporarily. When other nodes join again, these nodes will distribute data to the new joining nodes. Another alternative for nodes is to keep data items while away if they will for sure come back, so as to save some data exchange cost. Generally speaking, it is allowed to store data item at surrounding regions of a mapped region. Data item request is also allowed to be saved at surrounding regions, provided that they are finally directed to a certain region, and matching is found, it is fine. The delay in DTN is unavoidable.

To accelerate data look up, cache may be used. Nodes use cache to temporarily store their newly forwarded data items. If a new request matches one of the data items in the cache, no further look up is needed. However, the cache size can not be large. There are tradeoffs between cache size and the number of stored messages.

If nodes move out of their initial cells (nodes in cells could change cells as they wish), the initial cells work as relays (routers) for these (look up source) nodes. If a cell is empty temporarily, any information (look up request or returned data item) destined for it is kept at nodes that are surrounding the empty cell. Cell size does not necessarily need to be the same, so long as they cover the necessary communication area.

7) Location Diffusion: We assume nodes know their location and time. For every node, its description could be a tuple (location, time, movement pattern). When nodes meet each other, they will exchange (location, time) pairs. If there are multiple layer cells, all nodes in a lowest level cell report and store each other's location together with time stamp through location diffusion. In case there are two location reports concerning one node, the newer report prevails. This location information provides a node with an overall picture of who is in the lowest layer cell and can be used in the data storage and look up process.

B. Analysis of Multiple Copies Approach

Multiple copies approach is useful in accelerating data item storage and look up. In order to optimize the algorithms and protocols, we calculate the complexity of multiple copies approach. Assume every step uses an equal sized time interval T, a node is in the same state (alive or die in a certain cell) at every step and the probability $p = Pr[a\ node\ is\ alive\ over\ time\ interval\ T]$. We store a data item in s nodes and calculate the probability $Pr[item\ still\ exists\ in\ a\ node\ after\ k\ steps]$. If the probability that a node is alive is p, then the probability that a node either dies or moves out (of a cell) is $1-p$. If s copies are stored at nodes, the probability that an item exists in at least a node after k steps is $1-[(1+p+p^2+...+p^{k-1})(1-p)]^s = 1-(1-p^k)^s$. We use OCTAVE [1] to plot this probability with varying k, s.

We plot a figure with 0 to 20 steps and 0 to 20 duplicate copies. The node alive probability is 0.9 (Figure 4). It is clear that with multiple copies approach, the probability that at least a data item will stay in a cell is still very high even after 20 steps.

C. Operational Procedures

The proposed solution can be used in DTN applications, which include data item publication and storage, specific data

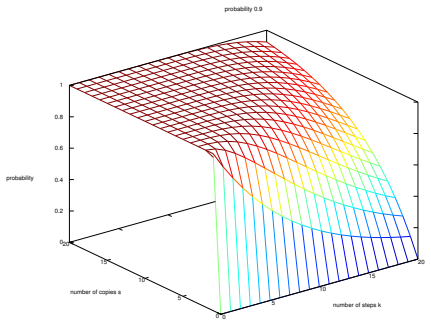

Fig. 4: Multiple copies with 0.9 alive probability

item look up and data item browsing. A node can publish its own data item and store it in DTN using CHT. If a node knows the name of the data item, it can map it to the specific cell (region) according to the working procedure of CHT. It is still possible that a node wants to browse available names of data items and find its own item of interest. In this case, we introduce a special type of name, called item inventory, which stores available data items. This name is reserved and can be mapped to a cell, just as other data item mappings. However, we emphasize that this inventory may not be complete in DTN. A record in the data item inventory includes the name and the data item description. Due to the distributed nature of the protocol, any inventory mainly stores data items in its locality (or a few faraway data items according to its own decision).

III. Experimental Evaluation

In order to evaluate our CHT distributed storage strategy, we perform simulations to compare CHT with MHT. During the experiments, we pay great attention to the key attributes, including maintenance overheads, data item storage latency, and storage success ratios.

A. Simulation Environment

The CHT is implemented using the NS-2 [2] simulator. This simulation environment includes full simulation of the IEEE 802.11 physical and MAC layers, which makes the simulation better reflect the real world. A random waypoint model is chosen as the motion pattern. For the propagation model, we have chosen *Two Ray Ground* which considers both the direct path and a ground reflection path. The simulation parameters are shown in Table I.

Through simulation, we show that data item maintenance overheads are significantly reduced if a data item is mapped to a cell instead of mapping to a location point. The delay performance and storage success ratios for storing a data item in a cell in CHT are also better than those in MHT.

For the simulation results, all points in the figures, as well as numbers in the tables are obtained as an average of 10 different runs with 10 different network topologies and movement patterns. The confidence intervals for the numbers are calculated at 95% confidence level.

TABLE I: Parameters of the simulations.

Parameter	Value
Number of mobile nodes	50
Mobility	0-20m/s(uniform distribution)
Transmission range	100m
Data rate	1 Mbps
Propagation model	*Two Ray Ground*
Simulation time	1000 seconds
Link layer queue length	150
Topology size	1500m×300m
Pause time	0 seconds
Packet payload size	1000 bytes
Antenna model	OmniAntenna

B. Single Cell Data Storage and Maintenance

We implement CHT with Mchord proactive mode on top of GLR routing protocol (GLR single copy approach is used for accurate maintenance overhead calculation). Although initially we want to compare CHT with original MHT which works on top of GPSR routing protocol, simulation results show that the original MHT can only achieve 6%±2% successful data item storage ratio. As a result, MHT on top of GLR is also implemented for comparison with our proposed solution. We store 20 data items at each scenario in both CHT and MHT. Our simulation results show that CHT experiences significant less data item handovers compared with that of MHT (one handover means from source node to the mapping node or from one qualified store node to another qualified store node), as shown in Figure 5. Table II shows the average data item storage delay (the latency when a data item first reaches a qualified storage node). It is clear that storage delay in CHT is also less than that of MHT. In the simulation, all data items have been properly stored at their storage sites in CHT while the success ratio in MHT is only 91.5%±4.85%.

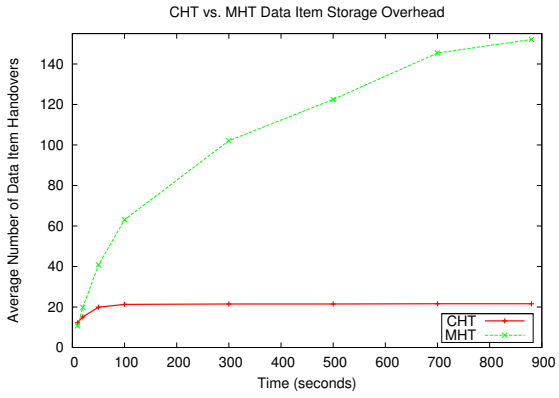

Fig. 5: Single cell storage maintenance overhead

TABLE II: Data Item Storage Delay

Protocol	Delay (seconds)
CHT	15.07 ± 2.14
MHT	81.92 ± 14.65

C. CHT Multiple Cell

Furthermore, we evaluate the multiple cell storage in CHT. Since MHT maps data items to locations, it does not consider multiple cells and works in the same way throughout the simulation. We divide the topology area into 9 cells with 100m×500m cell size and 5 nodes in each cell. So 90% nodes move within cells and 10% move globally. We assume rectangle centers as cell centers.

The simulation results clearly show that our multiple cell mapping scheme also significantly reduces the data item exchange overhead. Similar to the single cell situation, the longer the time, the more savings can be observed when compared with MHT. Since there are nodes which move globally and data items should be kept at their mapping cells, data item handovers exist over time in CHT, as shown in Figure 6.

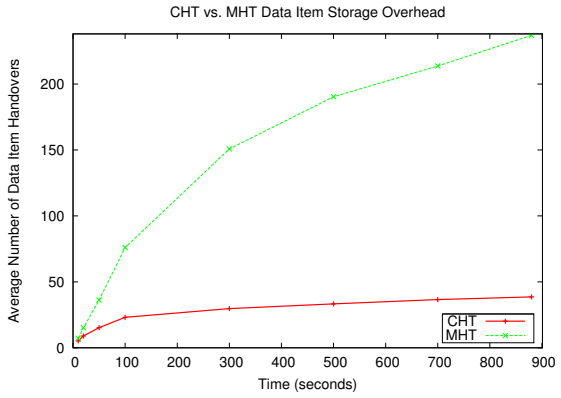

Fig. 6: Multiple cell storage maintenance overhead

IV. Conclusions

We have proposed a novel distributed data storage mechanism in DTN, called Cell-based Hash Table (CHT). CHT use cells (flat or hierarchical) to divide regions and cells are divided in such a way that nodes inside a cell have high probability of moving within. A data item is mapped to a lowest level cell by using local information and it is further stored at a node according to a modified chord mechanism (or DHT). Due to the use of CHT, data storage maintenance overheads are greatly reduced in DTN. Through simulation, we have shown that our scheme has less storage maintenance overheads with higher success ratios and less delays when it is compared with MHT storage mechanism. As future work, we plan to further study our CHT protocol, exploring the efficient distributed cell formation process in DTN.

References

[1] GNU OCTAVE, http://www.gnu.org/software/octave/, Accessed December 16, 2009.

[2] The Network Simulator, NS-2, http://www.isi.edu/nsnam/ns/, Accessed July 8, 2008.

[3] L. Alima, S. El-Ansary, P. Brand, and S. Haridi, "DKS (N, k, f): A Family of Low Communication, Scalable and Fault-Tolerant Infrastructures for P2P Applications," in *Proceedings of the 3st International Symposium on Cluster Computing and the Grid*, pp. 344–350, IEEE Computer Society. 2003.

[4] C. Becker and G. Schiele, "New mechanisms for routing in ad hoc networks through world models," *Proceedings of the 4th CaberNet Plenary Workshop, Pisa, Italy*, 2001.

[5] S. Burleigh, A. Hooke, L. Torgerson, K. Fall, V. Cerf, B. Durst, K. Scott, and H. Weiss, "Delay-tolerant networking: an approach to interplanetary Internet," *Communications Magazine, IEEE*, vol. 41, no. 6, pp. 128–136, 2003.

[6] J. Du, E. Kranakis, and A. Nayak, "A Geometric Routing Protocol in Disruption Tolerant Network," in *Proceedings of 6th Workshop on Wireless Ad hoc and Sensor Networks (WWASN2009), June 22, 2009, (ICDCS Workshops 2009, June 22-26)*, pp. 109–116. Montreal, Canada.

[7] D. Johnson, D. Maltz, J. Broch, *et al.*, "DSR: The dynamic source routing protocol for multi-hop wireless ad hoc networks," *Ad hoc networking*, vol. 5, pp. 139–172, 2001, Citeseer.

[8] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. Peh, and D. Rubenstein, "Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with ZebraNet," *ACM SIGPLAN Notices*, vol. 37, no. 10, pp. 96–107, 2002, ACM Press New York, NY, USA.

[9] B. Karp and H. Kung, "GPSR: greedy perimeter stateless routing for wireless networks," *Proceedings of the 6th annual international conference on Mobile computing and networking*, pp. 243–254, 2000, ACM Press New York, NY, USA.

[10] O. Landsiedel, S. Gotz, and K. Wehrle, "Towards scalable mobility in distributed hash tables," in *Proceedings of the Sixth IEEE International Conference on Peer-to-Peer Computing*, pp. 203–209, IEEE Computer Society. 2006.

[11] M. Li, W. Lee, and A. Sivasubramaniam, "Efficient peer-to-peer information sharing over mobile ad hoc networks," in *Second Workshop on Emerging Applications for Wireless and Mobile Access (MobEA II), in conjunction with the World Wide Web Conference (WWW)*, Citeseer. 2004.

[12] A. Lindgren, A. Doria, and O. Schelen, "Probabilistic routing in intermittently connected networks," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 7, no. 3, pp. 19–20, 2003, Springer.

[13] P. Marshall, "The disruption tolerant networking program," 2005. http://www.darpa.mil/sto/solicitations/DTN/briefs.htm, Accessed March 30, 2009.

[14] A. Pentland, R. Fletcher, and A. Hasson, "DakNet: Rethinking Connectivity in Developing Nations," *IEEE Computer*, vol. 37, no. 1, pp. 78–83, 2004.

[15] H. Pucha, S. Das, and Y. Hu, "Ekta: An efficient DHT substrate for distributed applications in mobile ad hoc networks," *Proceedings of the 6th IEEE IEEEWorkshop on Mobile Computing Systems and Applications (WMCSA)*, pp. 163–173, 2004.

[16] S. Ratnasamy, B. Karp, S. Shenker, D. Estrin, R. Govindan, L. Yin, and F. Yu, "Data-centric storage in sensornets with GHT, a geographic hash table," *Mobile networks and applications*, vol. 8, no. 4, pp. 427–442, 2003, Springer.

[17] A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," *Lecture Notes in Computer Science*, pp. 329–350, 2001, Springer.

[18] A. Savvides, C. Han, and M. Strivastava, "Dynamic fine-grained localization in Ad-Hoc networks of sensors," *Proceedings of the 7th annual international conference on Mobile computing and networking*, pp. 166–179, 2001.

[19] I. Stoica, R. Morris, D. Liben-Nowell, D. Karger, M. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: a scalable peer-to-peer lookup protocol for internet applications," *IEEE/ACM Transactions on Networking (TON)*, vol. 11, no. 1, pp. 17–32, 2003, IEEE Press.

[20] B. Zhao, L. Huang, J. Stribling, S. Rhea, A. Joseph, and J. Kubiatowicz, "Tapestry: A resilient global-scale overlay for service deployment," *IEEE Journal on selected areas in communications*, vol. 22, no. 1, pp. 41–53, 2004, Citeseer.