Implementation and Evaluation of Delay Tolerant Network Routing Protocols

Ravi Jadhav
rjadhav@cs.stonybrook.edu

Samhitha Kuduvalli Vasudeva Murthy
skuduvalliva@cs.stonybrook.edu

***Abstract*—In this project, we have considered implementing several Delay Tolerant Network routing protocols and evaluating each implemented protocol against each other to analyze their efficiency. Delay tolerant networks are those, which lack continuous network connectivity. We implement a subset of these routing protocols and analyze their performances in different environments. The project considers the dataset, which contains the real time data collected on the Cambridge Campus of number of interactions between 100 nodes, obtained from CRAWDAD for the analysis. The results are published in the form of graphs.**

I. INTRODUCTION

A Delay tolerant network is a network designed to operate efficiently over extreme distances such as those encountered in space communications or on an interplanetary scale[1]. In such an environment, there are huge latencies observed and that is inevitable. Delay tolerant networks require nodes that can store large amounts of data and those that can survive power loss. The data stored in a network should be accessible at any instant so as to achieve the best success rate; if otherwise then the efficiency of the network would decrease and might result in loss of data. Even in smaller scale networks, the characteristics of a delay tolerant network remain the same and the network best performs when the storage capacity at each of the participating nodes is the maximum and the every node interacts with every other node eventually either directly or indirectly. Such a situation cannot always occur and hence there is a need to explore on different ways in which the performance of the network can be maximized and the corresponding resource utilization minimized.

In this project, we have considered 4 delay tolerant network protocols, the epidemic routing, PROPHET, Spray and Wait- Vanilla and the Spray and Wait- Binary routing.

We implemented each of the mentioned protocols in Java, considering the functionalities of the respective protocols in different scenarios. Every protocol was then executed on a common dataset with the intent of comparing the performance of the protocols with each other.
The dataset obtained from CRAWDAD[2] consists of information such as the number of nodes and the details about the node movements. The dataset provides us with the information about the node interactions and the time at which the interaction took place with respect to a local time scale.
The dataset that we have referred to was collected in the Cambridge College Campus where several people were given small carrying devices (iMotes) which record the time that a particular device comes within a communicable range with another device. The dataset contains numerous such details among the 100 nodes that were involved in the data collection.

Among all the interactions in the dataset, we, for our experiment, have considered the number of interactions ranging from 5000 to 30000 in steps of 5000. The reduced dataset provides the opportunity to study the behavior of the protocols efficiently and the come up with a pattern that the protocols follow with respect to different attributes.
Apart from the number of interactions between nodes, we have also generated a list of messages that each node has to start off with. Each message in the implementation has a message ID, the source and the destination the message should finally be delivered to. All messages are considered to be of constant time and we have also considered that a message can be exchanged between two nodes in constant time. So, when two nodes in the network meet, there is no probability of partial message transfer between the nodes.

As the storage capacity at each node plays a very important role in the functionality of a network and its efficiency, we have evaluated the different protocols for varying buffer sizes. Buffer is the place where each node stores the information that it has to transmit to the other nodes. When the buffer size meets its capacity, then the node is modeled not to receive any further messages, until some message that is already in the buffer makes way for the incoming message.

II. RELATED WORK AND CONTRIBUTIONS

A significant amount of work has been done on the general Delay tolerant networks architecture and the routing protocols that can be considered in such networks. Delay tolerant routing algorithms are opportunistic-based routing protocols and a consensus on the right protocol to be used has not been arrived at.
The epidemic routing protocol[3] is a protocol for the partially connected ad-hoc network, which exploits each and every connection that the nodes in the network make with every other node hoping that the messages would eventually reach their

destinations.

On the other hand, the Probabilistic routing protocol[4] expresses certain restraint in the transferring of messages when two nodes in the network meet and would transfer the message if and only if the node has a higher probability of meeting the destination node than itself.

The spray and wait protocols[5][6], which derive ideas from epidemic and Probabilistic routing protocols, were proposed by three people in the electrical Engineering department of the University of Southern California.

We study the performance of all the mentioned protocols as a part of our project and finally compare the results from all the protocols to conclude on the efficient algorithm that can be used in our network. We explain the different algorithms in more detail in the respective sections.

III. THE EPIDEMIC ROUTING PROTOCOL

Epidemic routing[3] is a flood based routing protocol where the nodes continuously transmit the messages to every other node it meets while in motion, by replicating the message, if the newly encountered node does not have a copy of the message. This is a flooding based protocol. However, to prevent the extensive transfer of the messages, there could be multiple limits imposed on the network and it's functioning.

A. Implementation

We have developed the Epidemic routing protocol in Java with all the necessary features and giving importance to the slightest detail. When two nodes interact, each of the two nodes check if any of the messages that one node has in it is not present in another. If any message is not present, then that node transfers the message to the node it just interacted with. Upon receiving a message, the node determines if it is the message's intended destination. If not, then the node checks if it has enough buffer size to accommodate the storing of the incoming message. If yes, then the node accepts the message and continues its motion.

In epidemic routing, the messages keep moving around the network despite the message reaching the destination, as the other nodes are not aware of the fact that the message has already reached the destination. This creates a lot of traffic in the network. Hence, we have considered limiting the number of retransmissions of a message by using the message's hop-count value.

Upon every retransmission, the hop-count of a message is reduced and when the hop-count reaches 0, then the node no longer retransmits the message, even if the message that a particular node is currently interacting with is the destination of the message with hop-count 0. We have set the hop-count value in our implementation to 5.

B. Evaluation

The Epidemic routing protocol implementation was tested for various values of buffer sizes. The limit on the buffer size was varied from 10 to 100 in increments of 10.

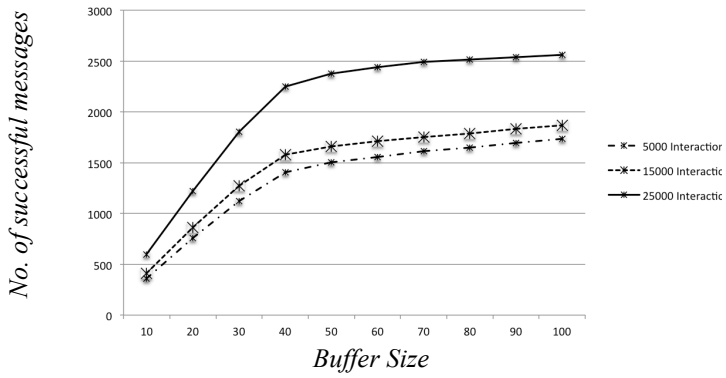

Figure 1. Graph showing the number of successful messages transmitted in the network versus the buffer size for varying number of interactions in the Epidemic routing network.

It is observed that the number of messages that have successfully reached the destination, increase with the increasing buffer size. This behavior is expected because the nodes drop the incoming messages if the buffer size goes beyond the configured limit. We can also observe that the number of successfully received messages increases with the increase in the number of interactions among the nodes. As the number of interactions increase, the probability of one node meeting another eventually increases and the probability of the message reaching the destination also goes up.

The major drawback in epidemic routing is the number of messages that is sent in the network. This causes a huge traffic when the number of nodes is high and hence results in collision and the messages not reaching the destination successfully due to the routing messages. Because every node transmits to every other node the message that is unique to itself in the interaction, the traffic is proportional to the buffer size and the number of interactions.

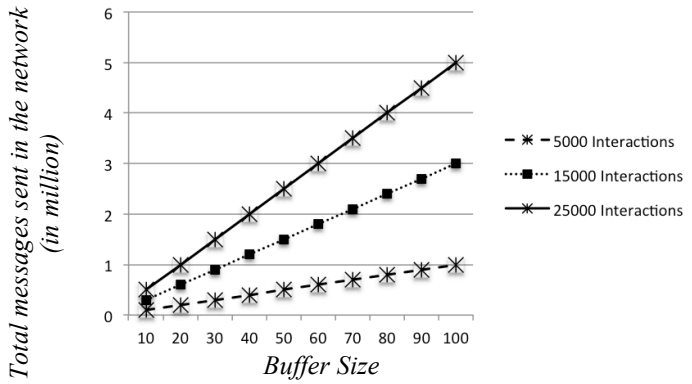

Figure 2. Graph showing the total number of messages transmitted in the network versus the buffer size for varying number of interactions in the Epidemic routing network.

IV. THE PROBABILISTIC ROUTING PROTOCOL

This protocol[4] tries to overcome the drawback of the Epidemic routing protocol by making wiser and informed decisions before transferring a message to the node it interacts with.

Any node transfers the unique messages it has to the node that it is currently interacting with if the probability of that node meeting the destination of the message is higher that the node which currently has the message. If the probability is found to be higher, then the copy of the message if sent to the interacting node and the process continues.

There are 2 parameters that are considered while updating the probability value of a node. The first one is the aging constant. Each probability reduces by the aging factor(γ) with every passing time(k units).

$$P_{(a,b)} = P_{(a,b)old} * \gamma^{\kappa}$$

The probability of a node meeting another node is increased with each interaction of the two nodes.

$$P_{(a,b)} = P_{(a,b)old} + (1 - P_{(a,b)old}) * P_{init}$$

We have also considered the transitive probabilities during the implementation and the transitive probabilities of a node indirectly meeting another node increases with every interactions between a pair of nodes.

The transitive probability is calculated by using the below formula. β is a scaling constant that decides how large impact the transitivity should have on the delivery predictability

$$P_{(a,c)} = P_{(a,c)old} + (1 - P_{(a,c)old}) * P_{(a,b)} * P_{(b,c)} * \beta$$

At the start of the simulation, each node is given a uniform probability of contacting with another node.

A. Implementation

Probabilistic routing is an extension of Epidemic protocol where the nodes consider the probabilities before transferring the message to the node it is interacting with. We have considered the probability to be varying in every unit time. This is with every passing unit of time in the dataset, we reduce the probabilities by the aging constant. The node before transferring the messages to another node, reduces its probability by the aging constant first, then updates its probability of meeting the node that it is currently interacting with, then calculates the transitive probabilities of all the nodes keeping the current node as an intermediate node and then transfers the message to the node in contact only if the probability of that node meeting the destination is higher that the node itself. We have kept the hop-count limit as 5 even in case of Probabilistic routing.

The aging constant is considered to be 0.98 and the transitivity constant is kept at 0.25. The probabilities of all the nodes meeting every other node in the network, is kept initially at 0.75.

B. Evaluation

The algorithm was executed for varying buffer sizes from 10 to 100 in increments of 10 and we observed that the number of successful messages increases with increasing buffer size.

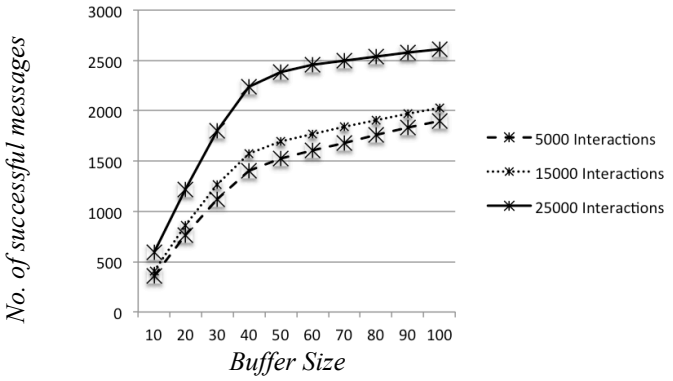

Figure 3. Graph showing the number of successful messages transmitted in the network versus the buffer size for varying number of interactions in the Prophet routing network.

Similarly, the total number of messages that were transferred in the network was plotted against the buffer size for the same number of interactions in the network.

The number of transferred messages grew proportionally to the increasing buffer size. But the total number of messages in the network was significantly lower than in the case of an Epidemic routing network. The number of successfully received messages did not vary significantly. On the other hand the successfully received messages increase due to the limited use of the available buffer size.

Hence the prophet algorithm works more efficiently than epidemic routing.

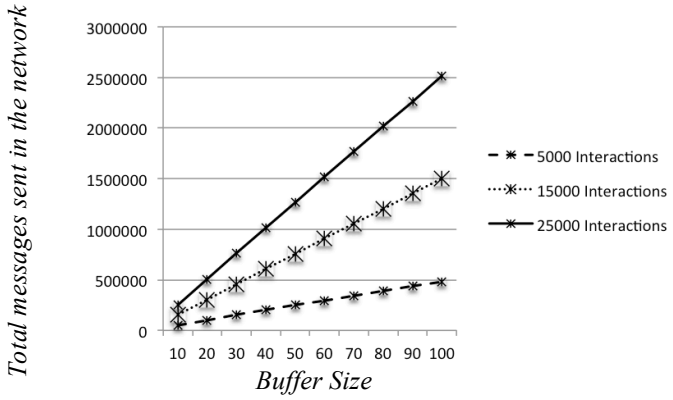

Figure 4. Graph showing the total number of messages transmitted in the network versus the buffer size for varying number of interactions in the Probabilistic routing network.

V. THE VANILLA ROUTING PROTOCOL

The Spray and Wait protocol[5][6] was developed by the researchers at the University of Southern California. It is a routing protocol that attempts to gain the delivery ratio benefits as well as the low-resource utilization benefits of forwarding-based routing. This protocol consists of two phases. The first phase is the spray phase where a copy of the message is sent to the interacting node and the next phase is the wait phase, where the node that received the message will hold on to the received message until the destination is reached.

There are 2 sub-protocols in Spray and Wait. We study the vanilla protocol in this section.

In the vanilla protocol, every node initially has, a predetermined L copies, of a message. The protocol does not consider the probabilities of every node. On the other hand, every node upon meeting another node in the network transfers a copy if the unique messages it has to that node only if the number of remaining copies in the node is not 0. After transferring the message, the node decrements the number of copies. The receiving node enters a wait phase for that message. That is that received message is transferred to a node that it meets only if it is the destination node for that message. This way the network avoids huge number of transmissions of the message and restricts it only to L.

A. Implementation

In order to implement this routing protocol, we have considered that the number of copies that a node initially has of every message is 10. Upon an interaction, a node first checks if it has any extra copy of the messages that it owns. If yes, then the node checks if the node that it is interacting with has the message or not. If this is found to be a unique message, then the message is transferred to the other node with the number of copies, L, as 0; so that the receiving node will no longer be able to transfer that message to any other node. The source node now decrements the number of copies that it has now, by the number of transfers of that message it made. This process is followed until the number of copies becomes 0 or till the destination is reached. Upon interacting with a node, if it is found that it is the destination for any of the messages that the current node owns, then irrespective of whether the L value is 0 or otherwise, the message is transferred. Hence the message eventually reaches the destination, if there is an interaction that favors it.

B. Evaluation

We implemented the Vanilla version of the Spray and Wait protocol with the number of copies as 10 at each node and we, like in the case of the first two protocols, have measured the number of successful messages and the total number of interactions by varying the buffer size.

The results that were obtained were positive and encouraging. With very less transfer of messages, the protocol was able to achieve a decent ratio between the successful messages received and the total number of messages.
The number of successfully received messages increased with the increasing buffer size and the total number of messages sent was found to be proportional to the buffer size too for varying number of interactions between the nodes in the network.

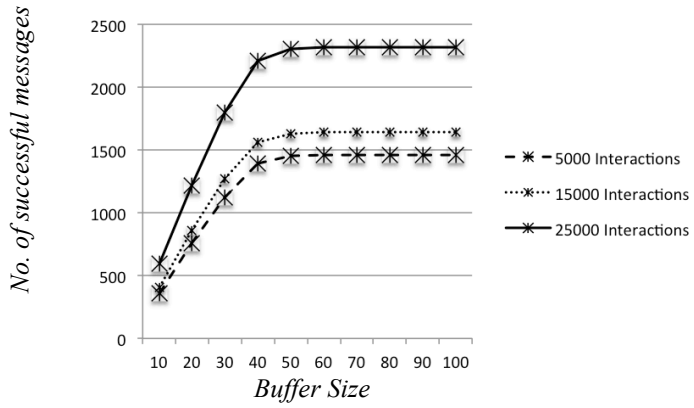

Figure 5. Graph showing the number of successful messages transmitted in the network versus the buffer size for varying number of interactions in the Vanilla routing network.

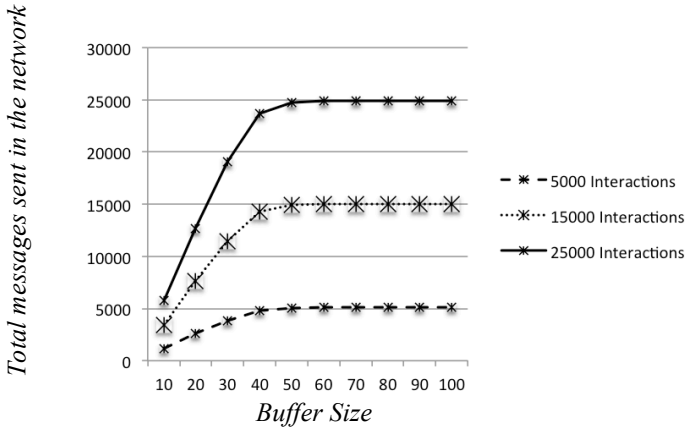

Figure 6. Graph showing the total number of messages transmitted in the network versus the buffer size for varying number of interactions in the Vanilla routing network.

VI. THE BINARY ROUTING PROTOCOL

Similar to the Vanilla version of the Spray and Wait protocol, the Binary version, aims at reducing the number of interactions by keeping the efficiency of the network as high as possible.

Every node starts with L copies of a message. But upon encountering a node, if the node is found not to have a copy of any message in the former node, then floor(L/2) copies of the message is transferred to the latter node. The latter node will now have floor(L/2) copies of the message and the former will have L-(floor(L/2)) copies. This process is followed by every node until a message reaches its destination, or the number of copies of a message in the node reduces to 0. When a node meets the destination of any of the messages it owns, the message is transferred to the node irrespective of the L value.

A. Implementation

While implementing the Binary version of Spray and Wait protocol, we used the Vanilla version of the protocol as the base

and built on it. The initial number of copies at every node was set to 10.

The hop-count does not play and role in the Spray and Wait protocols as the number of copies is limited by the value of the L field in the message.

B. Evaluation

The implementation of Binary protocol was executed on varying buffer sizes.

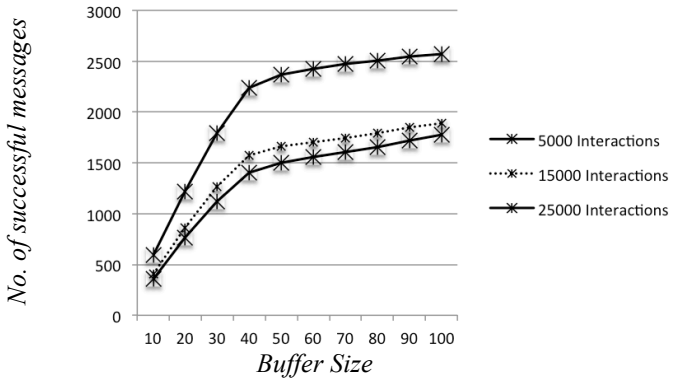

Figure 7. Graph showing the number of successful messages transmitted in the network versus the buffer size for varying number of interactions in the Binary routing network

For increasing buffer sizes, the number of successful messages delivered in the network increased and approached a constant value depending on the number of total messages in the network and the total number of interactions among nodes in the network.

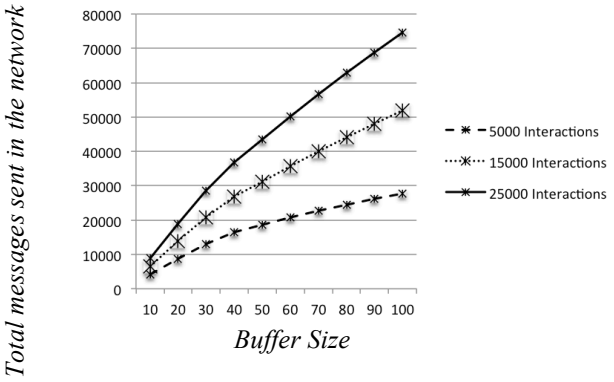

Figure 8. Graph showing the total number of messages transmitted in the network versus the buffer size for varying number of interactions in the Binary routing network

The total number of interactions in the network in order to achieve the number of successful messages received as shown in Graph 7, increases with the increase in the Buffer size. However, the number of interactions in the Binary routing protocol network is greater than that in the Vanilla network as the subsequent nodes too transmit the received messages in Binary. On the other hand, the number of successfully received messages increases, as there is a higher chance of the intermediate nodes to meet the destination.

VII. COMPARISON BETWEEN THE IMPLEMENED ROUTING PROTOCOLS

Based on the different results obtained by the individual evaluation of the different routing protocols, we compared the results of every protocol with every other with respect to the number of successful messages eventually received at the end of simulation and the total number of messages exchanged in the course of achieving this. For both the comparisons, the buffer size is kept at a constant value of 100.

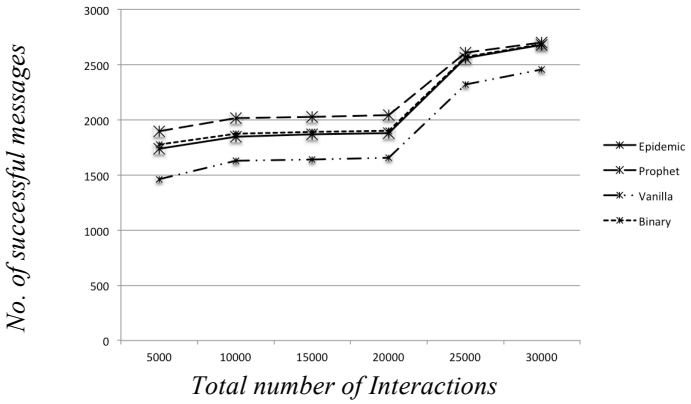

Figure 9. Graph showing the number of successful messages transmitted in the network versus the total number of interaction among all the nodes in various kinds of networks

From the graph, we can see that Prophet performs the best among the 4 protocols with respect to the number of successfully received messages. Literally, the number of successful messages in Epidemic routing has to be the highest as it transfers the message to every node that any node interacts with. But we see that the hardware of the nodes play an important role here and as we have limited the hop-count to 5 and the buffer size to 100, Epidemic routing fails to match Prophet.

Also Prophet outperforms both the versions of Spray and Wait protocol because of the forwarding limitations put in the Spray and Wait protocols. However, the Binary version of the protocol runs very close to Prophet and can be considered to provide the same performance with respect to the number of successful messages delivered in the network. We will have to now consider the amount of resources spent by nodes in each of these protocols to come up with the best one.

We would not want a protocol to consume very high number of resources and provide a result that is not too far away from the results of a protocol that consumes considerably less number of resources. In real world, the nodes will have a certain amount of resources, be it the battery power, or the storage space. We will have to make use of these resources carefully in order to make sure that to achieve the initial success in the transmission of the messages, we do not give up on the messages that may arise later in the network. If the storage space of the nodes gets fast filled up, then it will not have space to accommodate the messages from the nodes that it will meet later in the network. The problem grows bigger with increasing network size and the number of messages.

The graph below shows the total number of messages transferred in the network to obtain the number of successful messages that we saw.

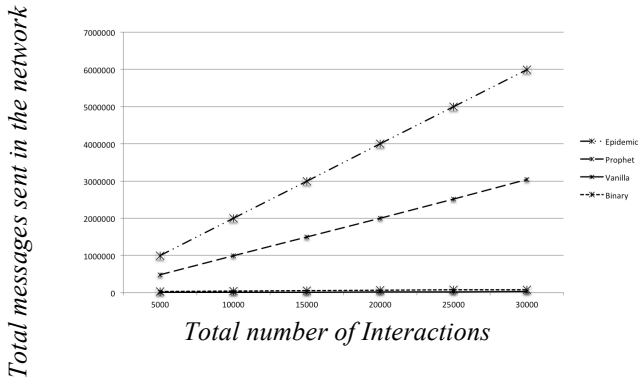

Figure 10. Graph showing the total number of message transmitted in the network versus the total number of interaction among all the nodes in various kinds of networks

From the graph we see that the total number of messages that is transferred in the network till the end of the simulation with constant terms for comparison is the highest in Epidemic routing network. This shows that the protocol, even though it provides a considerable success rate for small networks, is not scalable and cannot be deployed in larger networks.

On the other hand Prophet, which performed the best in terms of the number of successful messages received, stands second in the number of total transfers of messages among the nodes in the network. Vanilla and the Binary versions of the Spray and Wait protocol perform the best with least number of messages transferred through the course of the simulation.

The number of successful messages in the Prophet and the Binary protocol were very close. From this graph we see that Binary outperforms Prophet in terms of the efficient usage of the resources.

Hence in the simulation that we have conducted based on our implementations of Epidemic, Prophet and the Vanilla and Binary versions of Spray and Wait, the Binary version performs the best with very efficient usage of resources and high success rate of the messages reaching the respective destinations.

VIII. RESOURCES/REPOSITORY

The implementations of the different algorithms can be found in https://github.com/KVSamhitha/Delay-Tolerant-Protocols

The simulations were conducted for the total number of interactions starting from 5000 to 30000 with a step of 5000. In this paper, we have represented only 3 values (5000, 15000 and 25000) out of the conducted 6 experiments due to space constraints. The complete simulation results for every protocol can be found in this link - https://docs.google.com/spreadsheet/ccc?key=0AirCkn3kPaiXdDBsYzhlNU5mdWE1QzB2MzV4S0M4d2c&usp=sharing

IX. REFERENCES

[1] Delay Tolerant Network, http://searchnetworking.techtarget.com/definition/delay-tolerant-network

[2] CRAWDAD metadata link, http://crawdad.cs.dartmouth.edu/meta.php?name=cambridge/haggle

[3] Amin Vahdat and David Becker, "Epidemic Routing for Partially-Connected Ad Hoc Networks", Department of Computer Science, Duke University, http://issg.cs.duke.edu/epidemic/epidemic.pdf

[4] Anders Lindgren, Avri Doria and Olov Schelen, "Probabilistic Routing in Intermittently Connected Networks", http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.85.4880&rep=rep1&type=pdf

[5] Thrasyvoulos Spyropoulos, Konstantinos Psounis and Cauligi S. Raghavendra, "Spray and Wait: An Efficient Routing Scheme for Intermittently Connected Mobile Networks", Department of Electrical Engineering, University of Southern California, http://chants.cs.ucsb.edu/2005/papers/paper-SpyPso.pdf

[6] Wikipedia Link, Routing in Delay Tolerant Networking, http://en.wikipedia.org/wiki/Routing_in_delay-tolerant_networking