ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

TIK
Computer Engineering and
Networks Laboratory

Sumit Kumar
D-ITET Masters HS'09

Implementation and Evaluation of a CNA-based DTN Routing Protocol

Semester Thesis, SA-2009-30
September 2009 until January 2010

Supervisor: Prof. Bernhard Plattner

Advisors: Franck Legendre
Theus Hossmann
Thrasyvoulos Spyropoulos

Acknowledgement

It has been a wonderful experience working in DTN research area, an upcoming and challenging field of future inter-networking. It has been an interesting task to get to know the recent developments in this domain and contribute something from my side. I very much enjoyed working on the semester thesis which provided me an excellent learning curve and lots of insight into the research ongoing in this field.

I would like to thank and express my gratitude to Prof. Bernhard Plattner, head of Communication Systems Group (CSG), for providing the opportunity to work on this thesis. Special thanks to my advisors Franck Legendre, Theus Hossmann and Thrasyvoulos Spyropoulos for supporting me throughout working on this thesis.

Zurich, February 2010

Sumit Kumar

Abstract

Delay Tolerant Networking (DTN), deals with networks operating in challenged and extreme environment. These networks deal with large transmission delays, frequently disconnected paths, high link & path error and limited resources. Few examples of this kind of networks are satellite communication, ad-hoc & sensor, vehicular and military networks. Trying to model such networks into the traditional Internet architecture will lead to many issues ranging from performance degradation, rise in maintenance and operating costs and many networking overheads. Therefore, a separate research group was established to research DTN. They came up with a new DTN architecture which provides an alternative to model these challenged networks.

Routing is one of the most important aspects of any networking architecture. Lots of methods have been proposed for DTN routing, each having its pros and cons. Recently CNA (Complex Network Analysis) has been proposed as a toolset for DTN routing. The idea is to model the network as a social network where each graph edge represents one or more meetings between two nodes. Then CNA metrics, such as *similarity* and *centrality*, are used to assess the node's utility to carry a message. This semester thesis implements and evaluates a CNA based routing strategy. Thereby we focus especially on how to aggregate contacts to the social graph. It was recently proposed that density-based aggregation offers a more natural representation than time-based aggregation. The validation proves our above principle and results obtained shows that by density-aggregating contacts into a social graph and applying CNA metrics we can achieve an efficient routing between the nodes.

Contents

Table of Figures

A. Introduction

The Internet, as we know it today, has been a great success story connecting millions of communication devices across the globe. This was achieved by what we know today as TCP/IP protocol suite. All communication devices implement and run some kind of TCP/IP protocol suite to communicate.

In the Internet the connectivity is guaranteed by the fixed wired links/paths. These links have low latency, low error rates and low delay between any source and destination. With the advent of wireless networking this tenet became a little bit blurry. These wireless networks seldom use full TCP/IP internet suite and hence they are mutually incompatible. DTN, Delay Tolerant Networking [1] [2] [3], is a special class of networking area which addresses the extreme and performance-challenged environments where continuous end-to-end connectivity is not guaranteed. DTN is concerned with interconnecting highly heterogeneous networks together even if end-to-end connectivity may never be available. We cannot use traditional TCP/IP for these cases. Examples of such networks include inter space communication (interplanetary) networks, military networks, underwater networks, ad-hoc networks between mobile devices and sensor/actuator networks. Disconnection rather than connection is the '*normal*' case in these kinds of scenarios e.g. interplanetary networks. Peer-to-peer / Ad-hoc connections between mobile devices can actually extend the capacity of infrastructure (3G, Wi-Fi) networks.

The routing in DTN is achieved by using a *bundle* protocol specifications for all the messages exchanged in the network. This bundle protocol specification is defined in RFC 5050 [4]. This RFC defines how the nodes participate and cooperate in order to transfer, deliver and forward the bundles (messages) in the network. A *bundle* is defined as a single protocol data unit containing the control data and application payload across the nodes in the DTN network. All the DTN routing protocols conforms to this bundle protocol specifications.

Routing in DTN networks is a challenging research area. It has to deal with intermittent connectivity and regular link up/down. Like the traditional Internet we cannot maintain an up-to-date routing table at each and every router because of time and resource constraints. Thus in order to have routing in DTN networks researchers has proposed different methods over the time to achieve it. Normally they can be classified into 4 classes:

Schedule-based routing – The contact or '*meeting*' time between the nodes in the network are known well in advance. So we can exploit this and design the routing strategy as to schedule the message transfers during the meeting times. For e.g. Minimum Estimated Expected Delay (*MEED*) routing [15].

Probabilistic routing – Message forwarding is done at every hop whenever nodes are in contact with each other. The main idea is to increase the delivery probability at every step (greedy forwarding). No effort is made in identifying/predicting the better node or predicting the future encounters. The messages are distributed randomly to a number of nodes in the network. For e.g. Epidemic routing [7], Spray-and-Wait routing [8].

Utility-based routing – Routing is achieved by exploiting different utilities/features of the message or nodes e.g. time of last encounter, contact frequency of nodes, geographical locations and mobility patterns of the nodes. Nodes have some intelligence to decide, based on some metrics, which node is the better carrier for the message to the destination. For e.g. Age of last encounter routing [17], PROPHET [6].

CNA-based routing – Past observed contacts between the nodes are aggregated into a social graph, with graph edges representing past meeting between vertices (nodes). CNA metrics (*similarity and/or centrality*) are then applied on this social graph to arrive at the forwarding strategy. For e.g. *SimBet* [12], *BubbleRap* [13].

The objective of this semester thesis is to implement and evaluate a CNA-based DTN routing protocol based on [5]. As of now CNA based routing protocols are only evaluated by simulations and no real implementation exists. Unless we have a real implementation and validate on real mobile nodes/devices, we cannot prove their worthiness. Only after verification on real networks we will come to know the correct performance metrics of these protocols. The basic idea of all CNA-based protocols is to aggregate the contacts seen in the past to a social graph and then apply different metrics (e.g. *centrality* and *similarity*) or algorithms (*community detection*) to derive the routing decision. All the different CNA-based routing protocols proposed by researchers incorporate time window based aggregation (see below) of the social ties and contacts. But the

performance does not matter on the type of metrics or algorithm used, instead depends heavily on the aggregation of the contacts into a social graph. We can have 2 different aggregation schemes:

Time window based aggregation – The past seen contacts are collected over a time frame for e.g. consider all the contacts seen in the last 6 hours time frame.

Density based aggregation – The past seen contacts are collected over a density function for e.g. consider all the nodes seen more than 10 times in the last 4 hours.

We want to implement not from the scratch but base on implementation on existing work. PROPHET [6] is the most advanced and used implementation today. A utility based state-of-art routing protocol, PROPHET has been proposed in 2003 by researchers from Lule University of Technology, Sweden. It is the first routing protocol widely accepted by the DTN community and it has being considered for making an IETF RFC. This semester thesis takes the work of PROPHET as a baseline and work upon it to deliver a robust and efficient routing algorithm.

We structured the work on this thesis in the following 5 tasks:

1. Study and understand DTN principles and use cases
2. Study the understand different proposed DTN Routing protocols – Epidemic routing, Spray-and-Wait protocol, PROPHET, *SimBet*, *BubbleRap*
3. Understand the PROPHET code implementation
4. Implement the density aggregated CNA-based routing algorithm based on metrics Similarity and Centrality
5. Validate the implementation by emulating sample and valid topologies

In chapter B we discuss the related research work carried in DTN routing domain. It shows the different solutions proposed for the DTN routing and their pros and cons. Chapter C explains the implementation details followed by chapter D which talks about the validation. Chapter E concludes this thesis with discussing what all things needs to be achieved in the future.

B. Related Work

DTN networks overcome the problems of intermittent connectivity, long delay, symmetric data rate and high error rate by using a routing technique know as ***store-carry-and-forward*** message forwarding. The intermediate nodes stores the message until it gets a better route or node to transfer the message towards the destination. We can classify DTN routing in 4 classes:

1. Schedule-based routing

One approach towards DTN routing is when a node knows the schedule of the future contacts. This is known as scheduled contact routing. Schedule routing is possible when all the nodes in the network move along a pre defined paths and their contact schedule can be calculated. The scheduled meeting time can also be exchanged by some signaling methods. The best example can be of inter-planetary communication where we can calculate (almost precisely) the time when two planets can be at communication range so that transmission can take place. At the scheduled meeting time a node can transfer all the intended messages to the meeting node. One approach employing this technique is Minimum Estimated Expected Delay (MEED) [15]. The algorithm constructs a graph considering all the scheduled contacts and assigns a weight to every link. The link weight is a function of scheduled time, queuing time and propagation time. The main goal is to reduce the average path delay. Thus we can achieve efficient routing, under condition that we have all the schedule meeting times of the nodes in the network. But this is very challenging situation for DTN networks since the node failure and link disconnection is often not a pre defined process.

2. Probabilistic Routing

Main idea of this routing strategy is to forward the message probabilistically whenever the nodes are in contact with each other. No routing table is maintained at any node and next hop is decided at random. One of the earliest methods proposed was ***Epidemic routing*** [7]. In Epidemic routing each message is flooded across the whole network. When a node is carrying a message for some other node, it just gives a copy of that node to every other node it encounters. No assumptions are made in regard to control over node mobility or knowledge of network's future topology. Each node maintains a buffer containing all the messages it is carrying. When this node meets a new node they exchange this buffer information and decide which non-existent messages to transfer to

each other. After this interaction both nodes carry the same messages. It has the worst case performance with respect to the resources but has best delivery ratio as compared to other protocols.

Lots of optimizations have been proposed to improve the performance of Epidemic routing. The number of copies of a given message can be restricted to a finite number therefore reducing the resource usage. There was also an approach to reduce the overhead of flooding by only forwarding a copy with some probability $p < 1$, which is randomized flooding [16].

One of classical strategy proposed was ***Spray-and-Wait*** [8] routing protocol. It primarily aimed at reducing the number of copies in the network caused by epidemic flooding. The main idea is to limit the number of copies in the network to a finite small value (L). There are 2 variants of this approach

Binary Spray-and-Wait

Source Spray-and-wait

In the source spray-and-wait only the source node will transfer L copies to L relay nodes and these relay nodes, along with the source node, can forward the message only to the destination. Binary spray-and-wait mechanism uses forwarding tokens to distribute copies of the message. The source node starts with L tokens and on encountering a node, it will transfer a copy of the message as well as L/2 tokens. Now both the relay node as well as source node can copy the message to other nodes in the network unless $L = 1$. After L becomes 1 a node can transfer the message only to the destination. This technique improves the performance over its predecessors but still has many questions to answer. The main question which can be asked is how to select L? If L is very small in a very large network then whole network will not be covered with few copies and messages might not reach destinations far from the source. If L is very large for small networks then it is almost similar to the epidemic routing. Thus to overcome this problem we need to know the number of nodes in the network. But what if we don't know the number of nodes in the network i.e. we have no idea of the network size? Distributed estimation of the network size is a challenging problem.

3. Utility-based Routing

Utility-based routing tries to exploit some feature of the network topology, previous time of contacts, contact frequency of some nodes, geographical knowledge and mobility patterns. As seen in the previous section neither simple epidemic-routing nor routing based on scheduled contacts can guarantees a high delivery ratio in many scenarios. They don't fit within the requirement specs of an efficient DTN routing strategy but showed a path for further research into mitigating challenges being faced.

PROPHET [6] (Probabilistic Routing Protocol using History of Encounters and Transitivity) was one more classical protocol proposed for improving the delivery ratio and reducing the resource overhead in DTN environments. PROPHET uses the past encounters between the nodes to decide upon the routing decision. The basic operation is similar to the epidemic routing where when nodes are in contact with each other, they exchange a summary vector which contains all the message information which a node is carrying. In addition to the message information in the summary vector now the node transmits a probability value for each entry. Each node maintains a probability metric (delivery probability) $p < 1$ for every known destination (see below). On receiving a summary vector a node will calculate the new probability for all the destination and checks who is the better carrier for a given message. If this node is better carrier then it will request the message from the encountered node.

$$\mathbf{p = p_{old} + (1 - p_{old}) \times p_{init}}$$ where, p_{init} is an initialization constant [0, 1]

Whenever a node is encountered this metric is updated indicating that nodes are often encountered. So a high delivery probability means this given node is encountered more than other nodes. If a pair of nodes does not encounter each other in a while then the delivery probability must *age*, indicating less likely to carry messages to each other.

$$\mathbf{p = p_{old} \times \gamma^{k}}$$ where, γ is aging constant [0, 1] and k is number of time units elapsed since the last encounter.

The delivery probability also has a *transitivity* property that is based on the observation that if node A frequently encounters node B, and node B frequently encounters node C, then node C probably is a good node to forward messages destined for node A.

$$\mathbf{p_{(a, c)} = p_{(a, c)old} + (1 - p_{(a, c)old}) \times p_{(a, b)} \times p_{(b, c)} \times \beta}$$

where, β is scaling constant [0, 1] that decides impact of the transitivity

Few routing algorithms has been proposed which uses the so-called "time elapsed since the last encounter' or the 'last encounter age' [17] to route message to the destination. In order to route a message to a destination, it gets forwarded to neighbors who encountered the destination more recently.

Another utility-based routing technique uses location knowledge of the nodes [9]. This scheme uses the trajectories of mobile nodes to predict their future distance to the destination and transfers message to the nodes which are moving in the direction of the destination.

Another well known DTN utility-based routing proposed was *MobySpace* routing [10]. It is built on the notion that two people having similar mobility pattern are more likely to meet each other in effect will be able to communicate efficiently. The mobility pattern is formalized in a virtual Euclidean space known as *MobySpace* and each point in this space represents mobility pattern of a node called as *MobyPoint*. Routing is done by forwarding the messages towards the node which is having its *MobyPoint* closer and closer to the destination.

4. CNA-based routing

CNA-based (Complex Network Analysis) [11] routing tries to exploit the underlying social structure in the real world networks to derive upon the routing strategy. CNA as a field of research became popular with famous Milgram's experiment in 1967 which suggested that human society is a small world type network characterized by short path lengths. This experiment was aptly termed as "*six-degree of separation*". Since then there is lot of research going on in uncovering the social structures from all kinds of networks that we encounter on a day-by-day basis. Some of the examples are Internet, Paper citation network, biological networks, movie actors/actresses network, World Wide Web (WWW) and many others.

All these types of networks display substantial non-trivial topological features, with patterns of connectivity neither being purely regular or purely random. Such features include degree distribution, clustering coefficients, community & hierarchical structures, search or navigation capabilities. CNA deals with the research and empirical study of kind of networks exhibiting

these kinds of features. CNA has been proposed by different researchers as a way to predict the future contacts in DTN. Past observed contact in the network are aggregated in a social graph. Existing of the edge implies there was a past meeting between the nodes. Once we have this social graph we can apply different CNA metrics, such as *similarity and centrality*, on this graph and derive upon the routing strategy.

Similarity – Similarity is a property of social network which measures the transitivity of relationship (i.e. if nodes A & B and nodes B & C are neighbors then probability if high that nodes A & C are also neighbors). A high degree of transitivity means that there is a heightened probability of two people being friends if they have one or more other friends in common. Similarity for two nodes in question can be calculated by counting the number of common neighbors they have. This is also related to *clustering* or *network transitivity*.

Centrality – Centrality in CNA or graph theory is defined as the relative importance of a node/ vertex in the network/graph. In terms of social network terminology it can be stated as relative importance of a person within a social network. To understand it more clearly we can think how well a road is connected or how well it is used in an urban network. In the space syntax we can identify centrality as how important a room is within a building. There are four most widely used measures of centrality.

i) *Degree Centrality* – It is a measure of number of direct neighbors a node has. Or stated otherwise a node with high degree centrality maintains contacts with numerous other nodes. This implies the node in discussion holds an important structural position in the network with respect to the information exchange. And in contrast nodes having smaller degree centrality have fewer neighbors and are less involved in the information exchange in the network.

ii) *Closeness Centrality* – It measures the reciprocal of the mean geodesic distance (shortest path) between a node and all the other reachable nodes. It can be seen as a measure of how long it takes to disseminate information from one node to all the other nodes in the network.

iii) *Betweenness Centrality* – It measures the extent to which a node lies on the path of information exchange linking other nodes. In other words we can state it as a measure of the extent to which a node has control over information flowing between other nodes.

iv) *EigenVector Centrality* – It assigns relative score to all the nodes in the network based on the principle that connections to high-scoring nodes contributes more to the score of the node than connections to equal or less scoring nodes. Most widely known example of this centrality is Google *PageRank* technology.

Based on the above CNA metrics two protocols have been proposed – *SimBet* and *BubbleRap*. *SimBet* protocol derives the routing decisions based on Similarity & Betweenness calculation whereas *BubbleRap* partitions the network into communities and then use the ranking (centrality or node importance) to forward the messages.

SimBet [12] – A social graph is constructed by aggregating all the past contacts between the nodes. On this graph each node calculates the similarity and centrality values for each of the destinations for which the node is carrying a message. A *SimBet* utility is defined (as below) and its value is calculated for all the destinations at every node. When nodes encounter each other they exchange this utility and checks who the better carrier for a particular message is. The node having the higher value is the eventual carrier of the message.

$$SimUtil_n(d) = \frac{Sim_n(d)}{Sim_n(d) + Sim_m(d)}$$

$$BetUtil_n = \frac{Bet_n}{Bet_n + Bet_m}$$

And we calculate the *SimBet* coefficient as:

$$SimBetUtil_n(d) = \alpha \cdot SimUtil_n(d) + \beta \cdot BetUtil_n$$

SimBet is the protocol implemented in this thesis along with the density based aggregation techniques as described in [5].

BubbleRap [13] – BubbleRap tries to detect the communities in the underlying social structure of the network. After forming the communities nodes are ranked according to their popularity which is nothing but the centrality. It works on 2 ideas, first people have varying roles in the society and these should be true in networks also – so forward the messages to nodes which are more popular than other nodes. Second people form communities in their social lives which can be observed in networks as well – so identify the communities and mark members of the communities to act as relay nodes for the message.

C. Implementation

This chapter deals with the implementation details of the CNA-based routing in this thesis. Since PROPHET is a widely accepted DTN routing protocol and conforms to DTN bundle protocol specification, it has been used as the base for the implementation. All the bundle transfer messages and control flows implemented in PROPHET have been implemented to achieve a conformance to PROPHET. *SimBet* routing protocol is implemented with the density-based aggregation. In the following subsections we will see all the relevant details of the implementation.

1. PROPHET Implementation

One of the first complete, most talked about and widely implemented DTN routing protocol is PROPHET [14]. It was proposed by researchers from Lule University of Technology, Sweden. It is being considered to propose PROPHET as an internet RFC and discussions are currently undergoing at IRTF for the same. The routing is inspired by the mobility patterns of the mobile nodes and studying underlying behavior patterns. The operation of PROPHET is similar to Epidemic routing; only difference being that in PROPHET the routing decision is taken in an intelligent and probabilistic manner. Messages are forwarded to only those nodes which have a higher delivery probability than the current node. For routing protocol under consideration of this semester thesis, PROPHET was taken as a basis. Since PROPHET adheres to the DTN bundle protocol specifications [4] as well, it was ideal to choose it for the starting point. The main implementation details of PROPHET are as follows:

Written in C++

Uses the QT framework and QT libraries [18]

Each node is represented as a independent process having its configuration files which on initialization reads the node IP address, all protocol coefficients, different timer specifications and DTN bundle protocol constants

There is graphical widget to display all the packets exchanged and all the debug and log messages

Multi-threaded implementation in the sense that all the nodes are running in parallel and independently

The PROPHET implementation can be summarized as follows:

i) Each node maintains a list of all the nodes it has encountered in the past and calculates the delivery probability for each of the nodes

ii) On encountering a node it checks whether the node already exists in its node list. If new then add to the list and update the delivery probability; otherwise just update the delivery probability

iii) Timer is maintained to regularly age off the old nodes

iv) Also maintains the transitive property and updates all the nodes periodically

v) Doesn't take care of the multiple copies of the same message in the network. This means after transferring a message to a intermediate node, the message source will not delete the message. So after some time there will be multiple copies of the same message in the network.

In the following section we present some state diagrams to visualize the operations of the PROPHET implementation:

PROPHET Connection States:

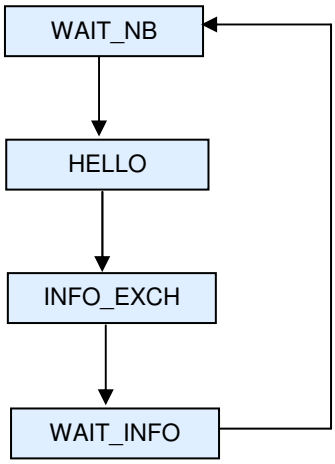


Figure 1: PROPHET Connection states

When a node is initialized it is in WAIT_NB state, which means it is waiting for a neighbor to connect. Once a node encounter a node in its vicinity a HELLO message is exchanged to indicate about each other's presence (see Fig. 2). After the HELLO state the nodes enter INFO_EXCH

and WAIT_INFO states where they exchange the entire summary vector information as well as final bundles. All the routing decisions and messages are transferred at this stage. The node goes back to the initial state, WAIT_NB, after the timeout or after all the information exchange has taken place.

HELLO Procedure States:

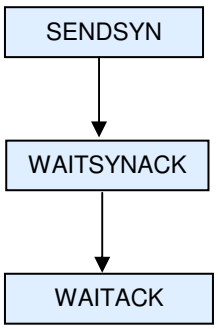


Figure 2: HELLO Procedure states

After a neighbor is discovered, the node goes for the HELLO message exchange. During the HELLO message exchange, a node first sends the SYN message which is the SENDSYN state. After this it enters the WAITSYNACK state where it waits for the neighbor to reply with its own SYN+ACK message. Once the neighbor replies it enters into WAITACK state. Once the initiator node receives this SYN+ACK message it sends the ACK message and then a connection is established between the nodes. The initiator becomes the MASTER node and receiver becomes the SLAVE node. To start the HELLO procedure a neighbor discovery routine is running in the background. Whenever this routing senses a neighbor it sends a signal to the upper application layer to trigger a HELLO message exchange.

MASTER (Initiator) States:

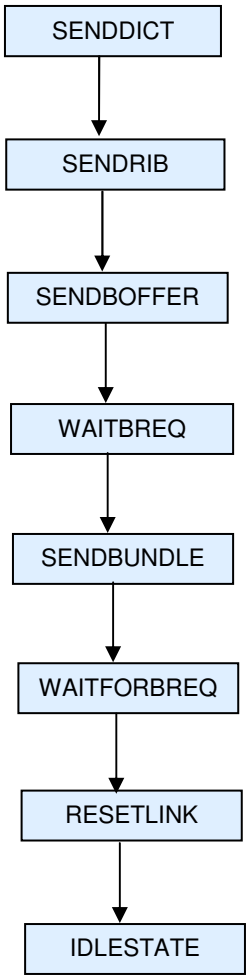


Figure 3: MASTER (Initiator) states

The MASTER node is the one which starts all the communication. The first state is SENDDICT where the node sends the *dictionary* message to the listener. It contains the list of all the nodes which this node has encountered in the past. SENDRIB is the state where this node sends the Routing Information Base to the neighbor node to calculate all the delivery probability and transitive delivery probability. RIB message contains the delivery probabilities values calculated at a node for all its contacts. After this the nodes enters SENDBOFFER where it sends the *bundle offer* message. Bundle offer contains information of all the destinations for which this node is carrying message along with the locally calculated delivery probability value. It then enters the WAITBREQ state waiting for the neighbor to send the list of messages for which it thinks to be the better carrier. Finally it enters SENDBUNDLE where it sends all the requested bundles and RESETLINK to reset the communication. After that the node is in IDLESTATE until it finds a new neighbor.

SLAVE (Listener) States:

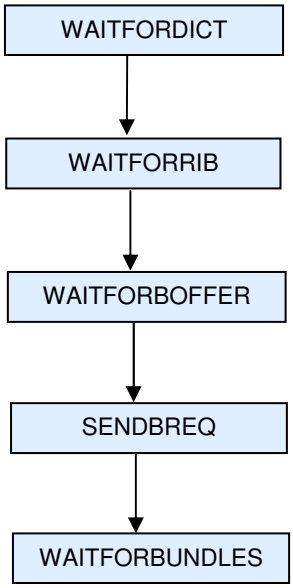


Figure 4: SLAVE (Listener) states

SLAVE node responds to all the communication sought by MASTER node. It first waits in the WAITFORDICT state waiting for the master to send the *dictionary* message. Then similarly it waits in WAITFORRIB and WAITFORBOFFER states waiting for RIB and Bundle Offer message respectively. On reception of Bundle offer message and checking for which messages it is a better carrier, it requests the master for the message in SENDBREQ state and then enters WAITFORBUNDLES state.

So for a mutual exchange of messages a node is in MASTER role for all the messages he is transmitting to the neighbor and in the SLAVE role for all the messages he is receiving. This constitutes the PROPHET connection states and roles for the mutual exchange of information.

2. Time Window and Density-based Aggregation

This semester thesis main goal is to implement and evaluate a CNA-based density aggregated DTN routing scheme. We have seen CNA based routing techniques in previous section as *SimBet* and *BubbleRap*. Both these well-known routing protocols use CNA metrics, *similarity* and *centrality*. However, the optimal performance can not only be achieved by the choice of metrics or algorithms, in fact it depends on the mapping from the mobility process generating contacts to the aggregated social graph [5]. To have a prediction of the future contacts and to properly evaluate the better carrier of the message, CNA-based routing techniques should best reflect the underlying social structure. Thus we need a proper way to aggregate the contacts in the social graph and must remove the old or stale edges.

We have two different ways how to aggregate the edges in a social graph:

i) Time Window based Aggregation

ii) Density based Aggregation

<u>*Time Window based aggregation*</u> – Most of the algorithms proposed so far uses time window aggregation scheme. If we order the contacts from time 0 to n as $C_{0,\,n}$ then an aggregation mapping can be defined at time n as:

$$f \;:\; C_{0,\,n} \;\rightarrow\; G_n(V, E_n)$$

where, G_n is the output social graph at time n

V set of all the network nodes

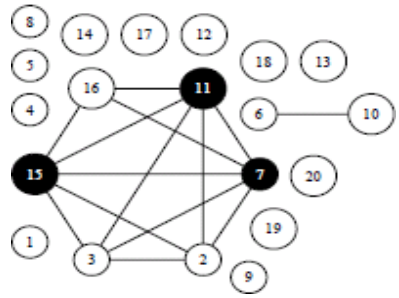
E_n subset of edges present in the graph at time n over the complete edge set E

We can have two different approaches in time window scheme:

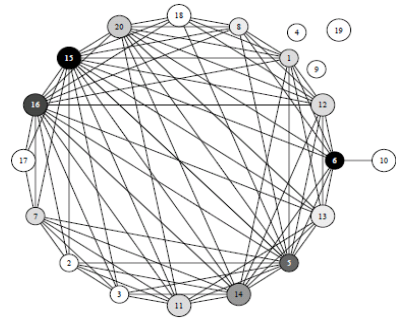
- *Growing time window* – All the contacts happened in past are taken into consideration. No effort is made to remove any of the edges in the graph, however old the edge is and when it was created.

- *Sliding time window* – A fixed time window of some finite duration is used to aggregate the contacts. Only the contacts happened in this finite time duration T are represented as edges in the social graph. For e.g. all contacts in 5 hrs time window.

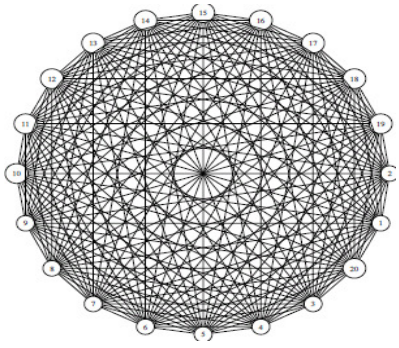
But time window aggregation can cause loss of heterogeneity in the social structure derived. For a very small time window all the nodes have similar centrality and similarity values since they do not have many contacts and hence metrics are not properly defined. Similarly, for large time windows i.e. over a long period of time, all the nodes have seen almost all other nodes and hence have again similar values for centrality and similarity values. This might be a very highly dense network graph. Figure 5 below depicts the situation for a sample ETH trace.



t = 1 hr



t = 2 hr



t = 72 hr

Figure 5: Time Aggregation of the ETH trace data

Density based aggregation – The choice of number of edges and which edges to include, determines the quality of the aggregated social graph. There is an optimal *density* for the social graph where most "regular" edges are in the graph but only few "random" edges. The density *d* of an aggregated social graph G_n is defined as fraction of aggregated edges over all the possible edges.

$$d(G_n) = \frac{|E_n|}{|E|}$$

We have to decide this optimal density carefully such that the underlying social structure is best possible set of edged depicting most relevant and regular nodes/edges. There are two methods to achieve this optimal density:

- *Most Recent contacts (MR)* – Each edge in the graph is given a timestamp according to the contact occurrence. In MR contacts we keep all the recent contacts which happened after some pre-specified timestamp, $t_{oldest,\ n}$

- *Most Frequent contacts (MF)* – Each occurrence of an edge is counted i.e. whenever two needs encounter each other a counter is incremented to indicate the frequency of contact. Only those edges are retained in the final graph G_n which has frequency at least equal to a pre-defined frequency value, $c_{least,\ n}$

The paper [5] states that "*performance of a DTN routing protocol not only depends on the aggregation but also on how the aggregation is done*", and, because of the results of the paper we will implement such a density based aggregation.

3. Routing Algorithm

The algorithm implemented in the semester thesis uses CNA metrics as used in the *SimBet* protocol i.e. *similarity* and *degree centrality*. The communication pattern between two nodes is similar to the PROPHET protocol implementation since it satisfies DTN bundle protocol specifications. When two nodes encounter each other, they exchange a series of control messages before the actual message transfer. These control messages at each node are used to determine which bundles to forward. The Figure 9 explains the interaction and messages exchanged between two nodes in contact:

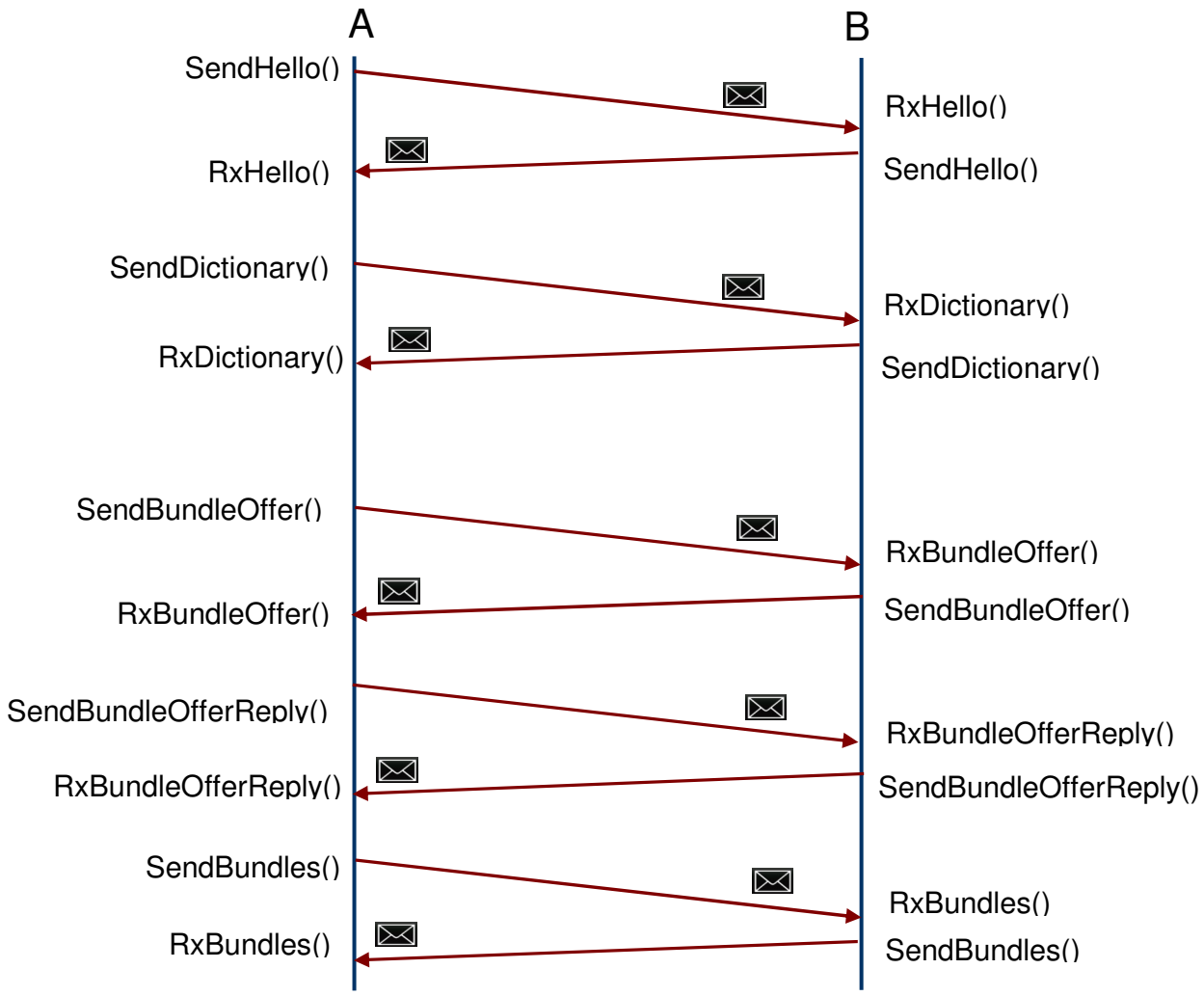


Figure 6: Interaction between two neighbor nodes

We can classify the interaction between two nodes in 5 stages:

1. *Hello* – When 2 nodes encounter each other, the first message which they exchange is a *Hello* message. This allows for nodes to know each other and initiate the further communication. Each node sends to each other a *Hello* message indicating its presence. On getting a *Hello* message, each node adds the other node to its nodelist and updates the time and frequency of contact. If the node has already encountered each other then each node will just update the time of contact and increment the frequency of contact. This condition is pre-defined by the DTNRG in the Bundle Protocol Specification as the first control message which has to be exchanges between two encountered nodes.

2. *Dictionary* – After the *Hello* message exchange nodes know that they have to perform the routing algorithm. To exchange the current state of a node with the neighbor, each node sends a message known as *Dictionary* to its neighbor. A Dictionary message contains information of all the encounters which a node has seen in the past. It contains a structure containing a list of encountered nodes, time of their encounters and their frequency of encounters. On receiving the *Dictionary* message a node will go through all the nodes in the message and add it to its own nodelist. But the difference is that all the nodes will be stored as *indirect* contact which means that the node has not met them directly but through some other contact. This is useful since we use this "indirect nodes" information to calculate the CNA metrics. Later, if some of the nodes, currently designated *indirect* contacts are seen directly (i.e. through a direct *Hello* message) then we change the flag and set it to *direct*. After updating the nodelist with the nodes in the dictionary message, a node will apply the MF and MR filter. Here we update the nodelist with all the recent contacts and which we feel are useful in determining the routing decision. The MF and MR values are configurable and support something like "*Keep only the contacts which you have seen in the last 30 minutes*". After applying the MF and MR filters we need to calculate the *similarity* and *centrality* coefficient of all the destinations for which a node is carrying a message. For calculating the centrality we take the number of direct contacts which a node is having, which is degree centrality. For similarity we check for each destination that how much similar is the encountered node is to it. We count the number

of neighbors which the destination and the encountered node have in common. To summarize in *Dictionary* stage we do following steps:

a. Add the all the nodes in the encountered node's nodelist
b. Mark them indirect contacts
c. Update the latest time and frequency of encounter for every node
d. Apply MR and MF filter
e. Calculate *similarity* and *centrality* for all the destinations

3. *BundleOffer* – After the *Dictionary* stage we need to tell the neighbor which all bundles a node is carrying along with the calculated *similarity* and *centrality* coefficients. We send this information in a different control message known as *BundleOffer* message. After receiving the bundle information list from the neighbor a node has to calculate the *SimBet* coefficient for itself and the neighbor node for all the destinations.

$$SimUtil_n(d) = \frac{Sim_n(d)}{Sim_n(d) + Sim_m(d)}$$

$$BetUtil_n = \frac{Bet_n}{Bet_n + Bet_m}$$

And we calculate the *SimBet* coefficient as:

$$SimBetUtil_n(d) = \alpha \cdot SimUtil_n(d) + \beta \cdot BetUtil_n$$

Then a node checks who is having higher value of *SimBet* coefficient. The node having the higher value for the *SimBet* coefficient will be the eventual carrier of the bundle. This information is transmitted in the next message. α and β are the constants needed to fine tune the similarity and centrality values according to their importance. For e.g. if we want to give both the coefficients equal importance then we can set α and β to 0.5

4. *BundleOfferReply* – After deciding that for which bundles a node is a better carrier, a node will send this information to the neighbor node. It will send the list of bundles which it thinks is a better carrier in *BundleOfferReply* message. On receiving the list of requested bundles from the neighbor, a node will remove those bundles from its list and transfer to the neighbor in the next stage.

5. *Bundles* – This stage just deals with the final list of bundles which has to be exchanged. After sending the bundle to the neighbor either a node can delete the bundle or can keep a copy to forward to other potential carriers.

Figure 6 shows the pseudo code of the implementation.

```
1   upon reception of Hello message h from node m do
2       if (newNeighbor(m) == true) {
3           addtoNodelist(m)
4           updateTime&Frequency(m)
5       }
6       else {
7           updateTime&frequency(m)
8       }
9       sendHello(m)
10
11  upon reception of Dictionary message dict from node m do
12      addNodeEncounters(m, dict)
13      updateTime&Frequency(m)
14      updateDensityBasedAggrMF&MR()
15      updateSimilarity()
16      updateCentrality()
17      sendBundleOffer(m)
18
19  upon reception of BundleOffer bo from node m do
20      Vector requestMsgs
21      for all destinations in bo do {
22          if (m.Simbet(d) < simBet(d)) {
23              requestMsgs.add(d)
24          }
25      }
26      sendBundleOfferReply(m, requestMsgs)
27
28  upon reception of BundleOfferReply bor from node m do
29      Vector transferMsgs
30      for all messages in bor do {
31          transferMsgs.add(bundleList.get(d))
32      }
33      sendBundles(m, transferMsgs)
34
35  upon reception of Bundles b from node m do
36      bundleList.add(d)
```

Figure 7: Routing algorithm, pseudo-code for a node

4. Class Diagram

Figure 7 shows the class diagram for the implementation. For testing and validation we run several nodes on one computer.

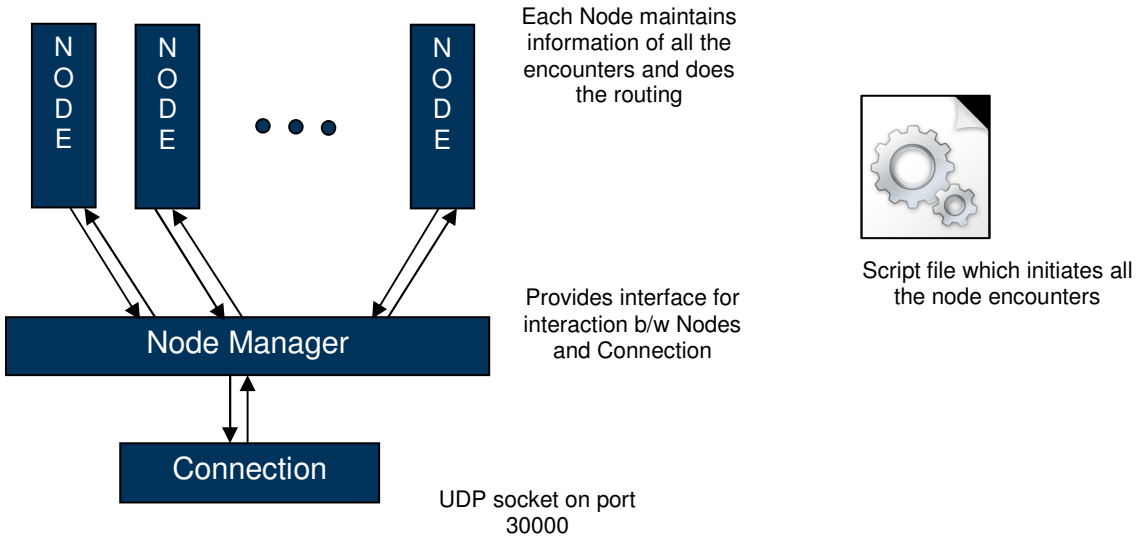


Figure 8: Class Diagram of the implementation

Node – The Node simulates individual nodes in the network. Each node class maintains a list for all nodes whom it has met in the past. It also maintains the time as well as the frequency of encounters. The node class stores all the bundles and marks them with *centrality* and *similarity* values. These values are updated whenever a node encounters a different node. The nodes exchanges all the control messages (*DICTIONARY*, *BUNDLEOFFER* and other messages) with the encountered node to decide upon the routing strategy as discussed in the previous section. Figure 8 and 9shows the different bundle headers used in the implementation. Each node is uniquely identified by a 32-bit integer. This ID/number is assigned at the time of node creation. Similarly all the messages are also identified by a unique message ID which is assigned while creating the message.

Node manager – The Node manager class provides the interface to all the node classes to transfer the bundles to the connection class. Also node manager will transfer the bundles received from the connection to the responsible nodes. It is like an abstraction between the Connection and

Node class. During initialization Node manager is initialized first and then it instantiates the Connection and Node classes. It also interacts with the script file for the initialization of the nodes and defines all the encounters.

Connection – The Connection class provides the interface to the node manager to transfer the bundles/messages to the network. It also captures the bundles from the network and gives those bundles to the node manager for the further processing.

Script file – This file behaves similar as the trace file which contains all the initialization constants and defines the nodes, their encounters and time of encounters. It also creates the messages in the network.

```
struct nodeInformation
{
   int contactFlag;              // 0 == direct contact; 1 == indirect contact
   int nodeID;                   // stores the node ID
   QDateTime timeStamp;          // stores the time of contact
   QMap <int, QDateTime> mapEncounters;      // stores the frequency of encounters and via
which node
};

struct Bundle
{
      int msgID;          // message/bundle unoque ID
      int similarity;     // similarity value
      int betweenness;    // centrality value
      int sourceID;       // bundle source ID
      int destinationID;  // bundle destination ID
};

// Different message types
#define MSG_HELLO                    1
#define MSG_DICTIONARY               2
#define MSG_BUNDLEOFFER              3
#define MSG_BUNDLEOFFERREPLY         4
#define MSG_BUNDLES                  5

/*common header for all types of messages to identify
   the intemediate source node and destination node */
struct Header
{
      qint32 h_type;                 // message Type
      qint32 h_srcnodeId;            // intermediate source Node ID
      qint32 h_dstnodeId;            // intermediate destination Node ID
      qint32 h_entryCount;           // Number of entries in the following message digest
};
```

Figure 9: Different data structures used in the implementation (a)

```
/*dictionary header for list of node
   alongwith time and frequency information */
struct DictionaryHeader
{
      qint32 h_length;                  // length of the encounter list
      qint32 h_stringId;                // node ID of the encountered node
      qint32 h_contactFlag;             // type of contact
      qint32 reserve;                   // reserve, used for structure packing
};


/*bundle offer header */
struct BundleOfferHeader
{
      qint32 h_msgID;           // message ID
      qint32 h_similarity;      // similarity value for this bundle
      qint32 h_betweenness;     // betweenness value for this bundle
      qint32 h_destinationID;    // destination ID
};

/*bundle offer reply header */
struct BundleOfferReplyHeader
{
      qint32 h_messageID;       // message ID
      qint32 reserve;           // reserve byte to balance the structure
};

/*bundle transfer header */
struct BundleTranfHeader
{
      qint32 h_msgID;           // message ID
      qint32 h_similarity;      // similarity value for this bundle
      qint32 h_betweenness;     // betweenness value for this bundle
      qint32 h_sourceID;        // Source ID of this bundle
      qint32 h_destinationID;   // destination ID
      qint32 reserve;           // reserve byte to balance the structure
};
```

Figure 10: Different data structures used in the implementation (b)

D. Validation

The validation was planned to do as emulation of sample topology and verify the basic functionality. Some regular topologies (chain, star and mesh) and specific topologies were emulated and the protocol was tested on them. Bundles were generated for different scenarios in different topologies. Validation was done on a single computer by emulating multiple nodes and sending messages between them.

For e.g. in the chain topology (see Fig. 10) we generate a bundle at first node for the last node in the chain. Also bundle was generated at each node for every other node in the scenario. Then we initiate the encounters at a specific time and check which all bundles are transferred to the destinations or transferred to a relay/intermediate node. The same procedure for all the different topologies and bundles were verified whether they reach the destination or a relay/intermediate node. This relay/intermediate node should be a better carrier of the message than the current node.

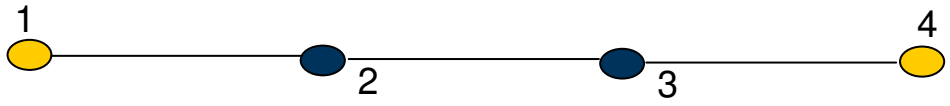


Figure 11: Chain topology

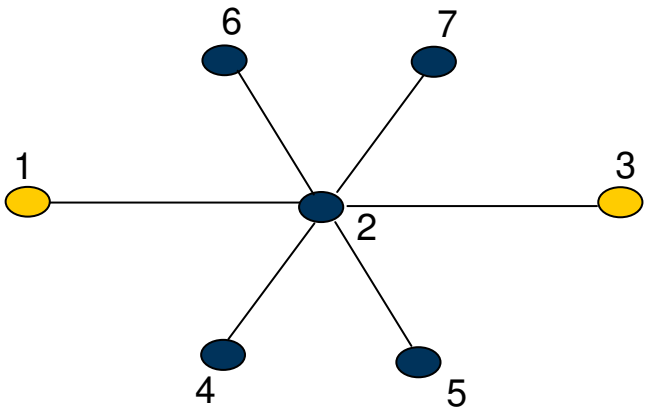


Figure 12: Typical topology to validate centrality

The topology as described in figure 11 depicts a case where bundle forwarding is done solely on *centrality* value. A bundle at node 1 destined to node 3 will go through node 2. After the initial

Hello exchange among all the nodes, every node will perform similarity and centrality calculation. Node 1 has a centrality value of 1 since it has only one neighbor. Node 2 has a centrality value of 6 since it has 6 neighbors. Both nodes 1 and 2 will have same value of similarity since both of them do not have any node common to destination node 3. So, after the node 1 and node 2 meeting, the bundle for the node 3 is transferred to node 2. Node will forward the bundle to node 3 in the next meeting time.

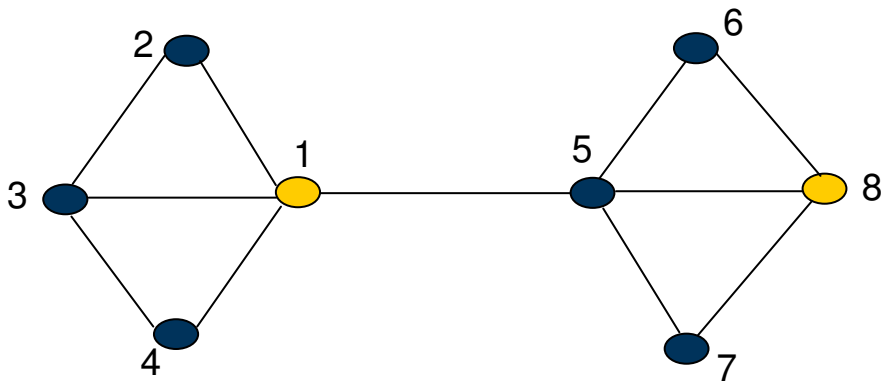


Figure 13: Typical topology to validate similarity

The topology as described in figure 12 depicts a case where bundle forwarding is done solely on *similarity* value. A bundle at node 1 destined to node 8 will go through node 5. After the initial Hello exchange among all the nodes, every node will perform similarity and centrality calculation. Node 1 has a centrality value of 4 which is total count of its neighbors. Node 5 also has centrality value of 4. So both the nodes have the same value of centrality. But node 5 is more similar to the destination node 8 than nose 1. Node 5 has a similarity value of 2 since it has two nodes as common neighbor to node 8. So after the node 1 and node 5 meeting, the bundle for the node 8 is transferred to node 5. Node will forward the bundle to node 8 in the next meeting time.

In all these test scenarios we have seen that that the implementation behaves as expected. So using the *similarity* and *degree centrality* values we decide the routing strategy. And by applying density-based aggregation (MF – Most Frequent and MR – Most Recent) we take only the nodes which have more recent information of the network.

E. Conclusions and Future Work

DTN is a very challenging research area and calls for more and more research. There are many open challenges in DTN routing which needs to be addressed. In order to have a proper understanding of the problems we first have to test the existing proposed solutions on the real mobile devices and test. The semester thesis achieves the first step towards this overall goal.

Over the period of 5 months we achieved a working implementation of *SimBet* routing protocol along with the density-based aggregation. This implementation was then verified using sample network topologies for the correctness. Due to shortage of time and effort required to finish everything, this semester thesis takes the first step towards the successful implementation and evaluation of a CNA-based DTN routing protocol. Looking at the broader aspect we need to evaluate the protocol against real world scenarios and compare with other state-of-art protocols. It has to be installed on the mobile devices like N95, HP iPaq etc. and real-time performance evaluation needs to be done. This semester thesis verifies the basic routing algorithm testing under light load and sample topologies.

In the future we need to deal with following work:

1. Run the trace file e.g. ETH Trace, MIT Trace etc
2. Real-world performance evaluation on mobile devices
3. Compare with other state-of-art protocols like PROPHET, SimBet, BubbleRap
4. Varying the number of copies of a message in the network to check the robustness and delivery ratio of the protocol
5. Evaluating the routing algorithm for long durations (weeks) to get the exact performance and working behavior

F. References

[1] http://www.dtnrg.org
Last checked: 06.05.2010

[2] A Delay-tolerant network architecture for challenged internets, K.Fall

[3] DTN Tutorial http://www.dtnrg.org/docs/tutorials/warthman-1.1.pdf
Last checked: 06.05.2010

[4] Bundle Protocol Specification – IETF RFC 5050 http://www.rfc-editor.org/rfc/rfc5050.txt
Last checked: 06.05.2010

[5] T Hossmann, T Spyropoulos and F Legendre - Know thy Neighbor: Towards optimal Mapping if contacts to social graphs for DTN routing. IEEE Infocomm 2010, San Diego CA, USA

[6] A Lindgren, A Doria, O Schelen - Probabilistic routing in intermittently connected networks. SIGMOBILE Mobile Computing and communication review July 2003

[7] A. Vahdat, D. Becker - Epidemic routing for partially connected ad hoc networks. Technical report CS-200006, Duke University (2000)

[8] T Spyropoulos, K Psounis and C S Raghavendra - Spray and wait: An efficient routing scheme for intermittently connected mobile networks. ACM WDTN 2005

[9] J Lebrun, C N Chuah et al. - Knowledge based opportunistic forwarding in vehicular wireless ad hoc networks. IEEE VTC 2005

[10] J Leguay, T Friedman et al. - Evaluating mobility pattern space routing for DTNs. In proc. INFOCOMM 2006

[11] M E J Newman - The structure and function of complex networks. SIAM review March 2003

[12] E M Daly and M Haahr - Social network analysis for routing in disconnected delay tolerant manets. ACM MobiHoc 2007

[13] P Hui, J Crowcroft and E Yoneki - Bubble-Rap: Social based forwarding in delay tolerant networks. ACM MobiHoc 2008

[14] http://prophet.grasic.net/
Last checked: 06.05.2010

[15] Hussein G. Badr, Sunil Podar – An Optimal Shortest-Path Routing Policy for Network Computers with Regular Mesh-Connected Topologies.

[16] Y.-C. Tseng, S.-Y. Ni, Y.-S. Chen, and J.-P. Sheu – The broadcast storm problem in a mobile ad hoc network.

[17] Henri Dubois-Ferriere, Matthias Grossglauser, Martin Vetterli – Age Matters: Efficient Route Discovery in Mobile Ad Hoc Networks Using Encounter Ages

[18] http://qt.nokia.com/
Last checked: 06.05.2010