2012

Opportunistic lookahead routing procedure for delay tolerant networks

Priyanka Rotti
Louisiana State University and Agricultural and Mechanical College

OPPORTUNISTIC LOOKAHEAD ROUTING PROTOCOL FOR DELAY TOLERANT NETWORKS

A Thesis

Submitted to the Faculty of the
Louisiana State University and
Agricultural and Mechanical College
in partial fulfillment of the
requirements for the degree of
Master of Science in System Science

in

The Department of Computer Science

by
Priyanka Gururaj Rotti
B.E. Information Science and Engineering,
Visvesvaraya Technological University, 2008
December 2012

Acknowledgements

> As we express our gratitude, we must never forget that the highest appreciation is not to utter words, but to live by them
>
> John F. Kennedy

A simple and profound thought, but very true. However, my thesis document cannot be complete without my expression of gratitude to all those who made it possible.

I first fold my hands in gratitude to Almighty, for giving me the strength to make every dream come true, and for giving me the chance to live my dreams.

Next, I would like to thank my advisor **Dr. Rajgopal Kannan** for having guided me through my Masters course here at LSU. His support and help during the course of my thesis has been invaluable. The guidance, support and help that I have received from **Dr. Amit Dvir** cannot go unacknowledged. His untiring and timely help has been key to the completion of my thesis. I also thank Dr. Konstantin Busch and Dr. Jian Zhang for consenting to be on my thesis committee and supporting me through it.

I now acknowledge the support that I have received from my parents and my sister. They have been a constant source of encouragement at all times during my stay at school. This thesis would not have been possible without their support.

I would like to thank Maggie Edwards for all the administrative help she has given and Umesh Satish for being my critic and proof reading my thesis. A special note of thanks to Hari, Arpitha, Sashankh, Pratap, Sunada, Madhavi, Praveen, Narendra and Abhishek for

being a constant source of support though my thesis and masters program.

I am sure I have missed mentioning a lot of other people who made this possible for me. I express my sincere gratitude to every one who have been with me through this journey.

Contents

Abstract

Delay Tolerant Networks are wireless networks that have sporadic network connectivity, thus rendering the existence of instantaneous end-to-end paths from a source to a destination difficult or impossible. Hence, in such networks, message delivery relies heavily on the store-and-forward paradigm to route messages. However, limited knowledge of the contact times between the nodes poses a big challenge to effective forwarding of messages.

In this thesis, we discuss several aspects of routing in DTNs and present one algorithm and three variants for addressing the routing problem in DTNs: (i) the Look-ahead Protocol, in which the forwarding decision at each node to its immediate or one-hop neighbor is based on the position of the packet / message in the queue of the neighboring node(ii) Backpressure based lookahead, where a lookahead factor is introduced with the basic backpressure equation. This factor takes into account the difference of queue lengths from the neighbors, (iii) a two-step lookahead protocol, where the forwarding decision is sometimes based on the instantaneous one-hop neighbors of the neighboring node.

We also present simulation results of these protocols and compare these results to the existing standard routing protocols for DTNs. In all the algorithms, we look to optimize the amount of network bandwidth used by looking one step ahead before making a forwarding decision. By considering the queue in the neighboring nodes, the amount of network resources consumed decreases. The protocols that we propose may come with a slightly higher hop-count per packet than most protocols, but we have tried to maintain a comparable delivery ratio with the existing standard protocols.

Chapter 1

Introduction

The pervasive nature of wireless communications in the present day communication scenario has seen a rapid rise in heterogeneous networks. A heterogeneous network is defined as a network that connects computers and other devices which run different operating systems and operate on diverse communication protocols or access technologies. For example, a wireless network that provides a service through wireless LAN and is able to maintain its service when switching to a cellular network is a good example of a wireless heterogeneous network. Some heterogeneous networks that operate in mobile or extreme terrestrial environments lack continuous network connectivity. Delay Tolerant Networking is an approach that seeks to address the issues that render communication in heterogeneous networks difficult [22].

The advent of wireless protocols and their increased use in common communication environments invigorated research in the field of mobile ad-hoc networking (MANETs) and vehicular ad-hoc networks. The motivation and necessity for a Delay Tolerant Network(DTN)

was first discussed by Kevin Fall [7]. Since then, there has been a tremendous amount of research in making DTNs more suitable for usage in real world scenarios [22].

1.1 Challenges with Routing in DTNs

The ability to successfully route data from a source to a destination is a fundamental property desired in any communication protocol. DTNs are characterized by their lack of connectivity, thus a lack of instantaneous end-to-end paths. Traditional routing protocols like Optimized Link State Routing Protocol (OLSR) [10] and Ad-hoc On Demand Distance Vector Routing (AODV) [16] rely on the existence of end-to-end paths between source and destination. These protocols use *deterministic routing* approaches which perform well when the routes are known a priori. They first establish a complete route and then forward the actual data.

Consider the sample DTN as shown in Figure 1.1. If D has a packet to send to B, there is no path from D to B. However, at T_4, the packet can be forwarded to B by C if D forwards the packet to C in T_1. An important consideration here is a prediction by D about its probability of meeting C.

In a disconnected network like in the case of DTNs, end-to-end paths are difficult or impossible to establish. This established a need for a new approach to routing, that works around the limitation of a lack of instantaneous end-to-end paths.

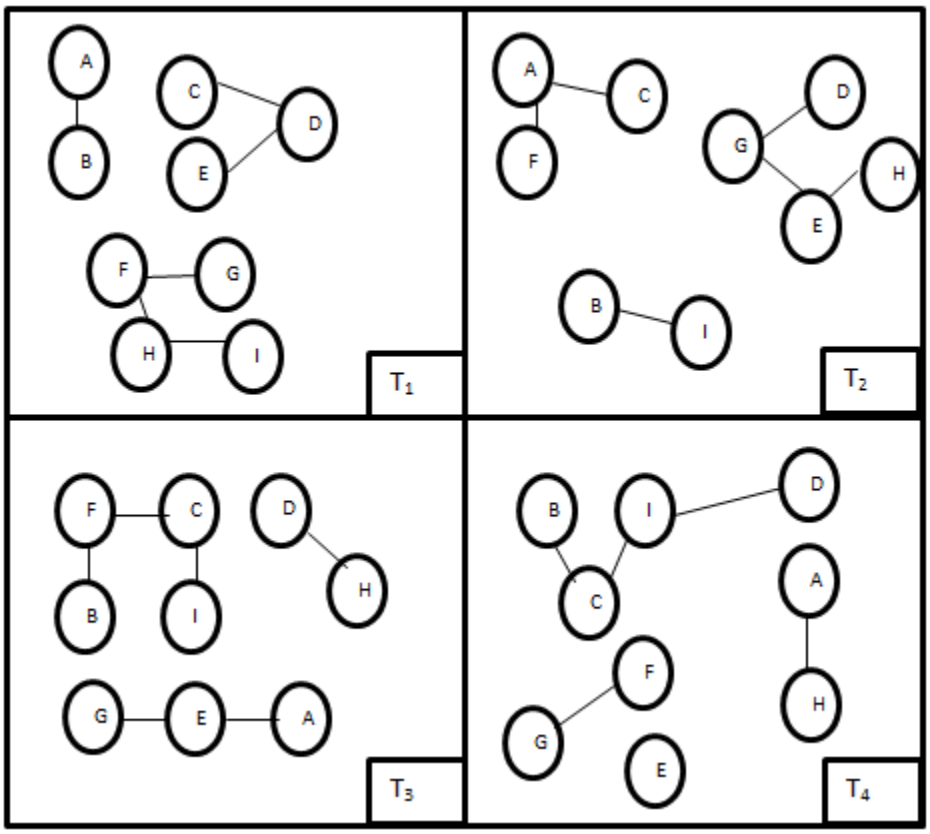

Figure 1.1: Sample DTN Scenario

1.2 Approaches to Routing in DTNs

In DTNs, *opportunistic routing protocols* are used. The concept of opportunistic routing is well-suited to mobile networks where there are significant disruptions to node availability. These protocols use redundancy among the nodes. They use the node that is available for routing *at the time of transmission* as opposed to pre-defined routes from source to destination. The forwarding decision is made by the protocol that is employed for routing. These protocols use the *store-and-forward approach* where data is stored and moved through the network incrementally, in the hope that the data will eventually reach the destination.

The most basic form of opportunistic routing is Epidemic dissemination [21]. This protocol preceded many other protocols proposed to address the issues of routing in DTNs.

In most protocols, there are multiple copies of the message present in the network, which increases the probability that the message is delivered to its destination and reduces delay. However, these protocols are expensive when the local storage and inter-node bandwidth are both limited. When the local storage and the inter-node throughput are more tightly constrained, more discriminate algorithms are needed. Therefore, while designing a routing protocol for a delay tolerant network, the important considerations must be

- The number of copies that are distributed in the network for each message
- The selection of nodes to which the message is replicated and forwarded

Routing in delay tolerant networks is seen as a trade-off between the message delivery ratio, the network overhead and the delivery delay in the network.

1.3 Our Contribution

In this thesis, we propose a new opportunistic routing protocol for DTNs and three variants of the protocol. This protocol uses just one copy of the packet in the network for routing instead of multiple copies. We analyze the problems of routing on homogeneous and heterogeneous environments. The protocol and its variants that we propose try to reduce the cost of delivering messages to their destination. The basic features of the protocols are the following:

- We propose a look-ahead factor in the protocols. This factor takes into account the number of packets with greater priority in the queues of the neighboring nodes and the position of the current packet in the queue of the neighbor. None of the preceding protocols have a lookahead property in them. In this protocol and its variants, there is only one copy of the message in the network as opposed to the multiple copies as with the other opportunistic protocols.

- The first variant of the protocol is one in which, depending on the queue information received, the neighbors are placed into discrete buckets. From the bucket of least value, a node is picked randomly. We call this the Random Look-ahead Protocol.

- We propose another variant of the Lookahead protocol. In this variant, we check if the destination of the packet is a one-hop neighbor of one of the neighbors. In other words, we check if the final destination of a packet is two-hops from the current node.

- A third variant is the Backpressure Look-ahead protocol. The Backpressure Collection Protocol [15] is one of the recent protocols that has been proposed for routing in sensor networks. To the native Backpressure equation that deals with the difference in queue lengths of two nodes and the Packet Reception Rate (PRR) between the two, we add a new factor, LA (for lookahead).

1.4 Organization of the Thesis

This thesis is organized as follows: Chapter 2 discusses the previous work that led to this thesis. Chapter 3 explains the Look-ahead protocol and its variants. Chapter 4 gives the simulation results and discusses the behavior of the protocol. Chapter 5 looks at the conclusions and the future work that can be done to improve the protocol.

Chapter 2

Related Work

More than a decade has passed since Kevin Fall [7] discussed the need for DTNs. The primary focus of the research on DTNs has been the routing problem. There has been a lot of work done to handle the intermittent connectivity between the nodes in a DTN and provide for successful and efficient delivery of messages.

2.1 Classification of Routing Protocols for DTNs

2.1.1 Classification Based on the Knowledge Available at Nodes

The routing protocols for DTNs can be classified based on the knowledge of the network available at the nodes. Figure 2.1 gives a good representation of the categories of routing protocol based on the knowledge of the network available at the node.

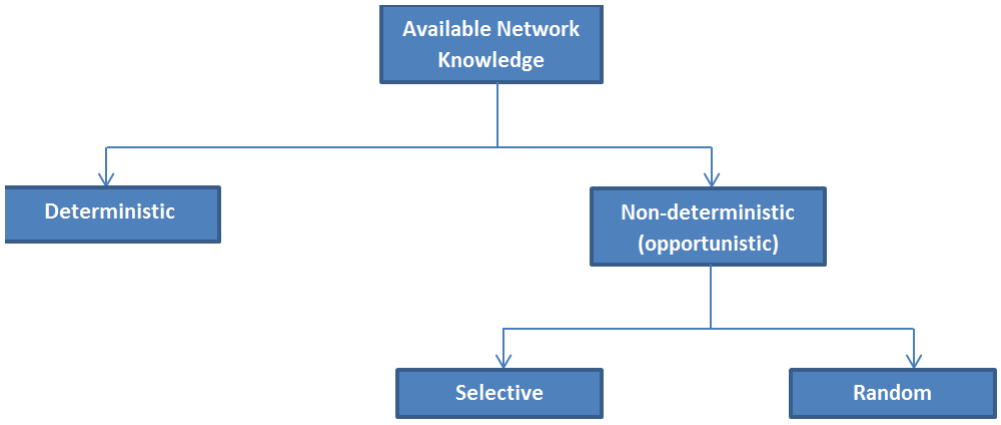

Figure 2.1: Classification of DTN Routing Protocols based on Knowledge of Network at the Nodes [3]

In some studies, it is assumed that each node in the network has absolute knowledge of node meeting times and durations. Based on this knowledge, messages were routed from a source to a destination using pre-determined paths. This type of routing is called as *deterministic routing*.

However, in DTNs, due to sporadic connectivity between the nodes, deterministic information about a path from a source to a destination is not available. There are a significant number of studies that assume zero knowledge of a pre-determined path between a source and destination. Epidemic Routing [21], Prophet Router [14], Spray and Wait [19] are some examples of non-deterministic or opportunistic protocols. Some of the protocols randomly forward the messages to neighboring nodes while some of the protocols use some decision making to choose the nodes to which the messages are forwarded.

2.1.2 Classification Based on Number of Carriers

In some DTN routing protocols, there is only one node that carries the message. This node forwards the message to the neighbor that has the highest probability of meeting the

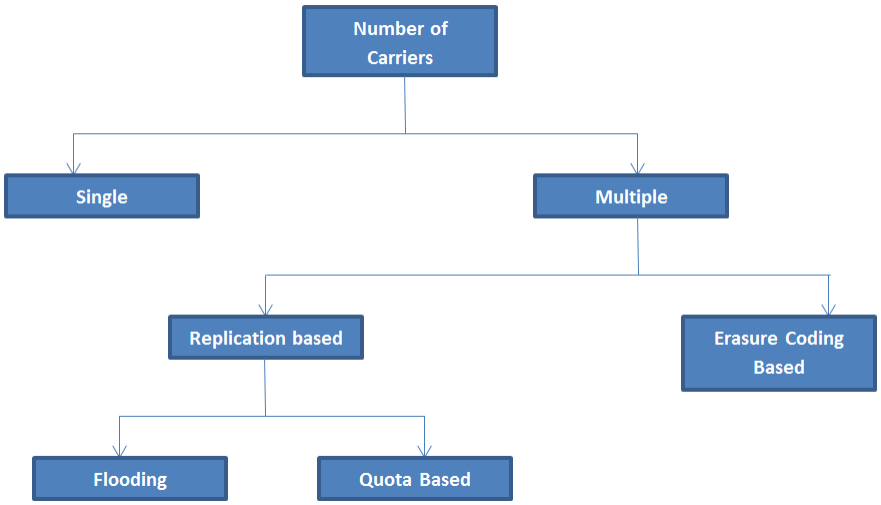

Figure 2.2: Classification of DTN Routing Protocols based on Number of carriers [3]

destination. Examples of such protocols are Prophet [14], MaxProp [4]

Another approach to routing is when there are multiple carriers that carry the message. In some of these protocols, the message is replicated and distributed to different nodes. Some protocols process k data blocks into a large set of ϕ blocks. Each of these blocks is distributed in the network. If a node receives sufficient number of blocks, then it can reconstruct the original message.

2.2 Some Popular DTN Routing Protocols

The Epidemic Routing protocol [21] is the pioneering protocol for routing in DTNs. This protocol is a flooding algorithm. The nodes continuously replicate and transmit messages. When a node comes within transmission range of another node, it checks whether its new neighbor has a copy of the message that needs to be transmitted. If it does not, then a replica of the message is forwarded to that node. This is done using *Summary Vectors*.

Each host has a buffer of messages that it originated as well as messages that it is buffering on behalf of other nodes. It maintains a hash table that indexes this list and keys that uniquely identifies each message. Each host maintains a bit vector called *summary vector* that indicates which entries in the hash table are set. When two hosts come in communication range of one another, the hosts exchange their summary vectors to determine which messages stored remotely have not been seen by the local host. Each host then requests a copy of the messages that it has not seen yet. The receiving host has complete autonomy to accept or reject a message.

The figure 2.3 is a representation of the exchange between two hosts when they come into transmission range of one another.

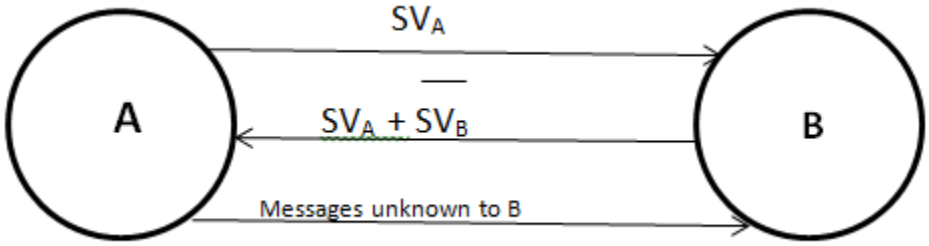

Figure 2.3: Epidemic Routing Protocol when two hosts, A and B, come into transmission range of one another

The PROPHET routing algorithm [14] address this issue. Lindgren et al. believe that the movement of nodes in a typical ad hoc network is not random as assumed. They claim that the nodes move in a predictable fashion. The PROPHET router uses this observation and aims to reduce the amount of local storage and network bandwidth being used. In this protocol, when two nodes meet, they exchange summary vectors and delivery predictability information. The delivery predictability $P_{(a,b)} \in [0, 1]$ is a probabilistic metric at node A for every known destination B.

The predictability is updated using three rules. These are:

- When node A encounters node B, then, the delivery predictability is updated as:

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

  $P_{init} \in [0, 1]$ is the initialization vector.

- This protocol runs periodically. Hence, if two nodes have not encountered each other in a while, then there is an aging factor that will reduce the delivery predictability for the source destination pair.

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

  $\gamma \in [0, 1)$ is the aging constant. k is the number of time units since the encounter between a and b.

- If A encounters B and B encounters C, then C has good forwarding predictability from A. To calculate this,

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

  $\beta \in [0, 1]$ is the scaling constant that represents how much the delivery predictability is affected by the transitivity.

The values for $P_{(a,b)}$ for each node is updated regularly. When a node A encounters another node B, then the message is forwarded to B only if $P_{(a,b)}$ for the destination at B is greater than the value at A.

Burgess et al. [4] proposed a method based on prioritizing both the schedule of the packets being transmitted and the schedule of the packets being dropped. MaxProp lets each node keep track of the probability of meeting other nodes. f_j^i denotes the probability that node i is connected to node j. This is initialized to $\frac{1}{|s|-1}$ where s is the is the total number of nodes. In each meeting of i with j, the value of f_j^i is incremented and then all the values of f are normalized. Every time two peers meet, these values are exchanged.

Once a node has the values of the delivery likelihood for the other nodes, then it calculates the cost of each possible path, c(i, i+1, ..., d) to a destination upto n (defined by protocol) hops according to:

$$c(i, i+1, ..., d) = \sum_{x=i}^{d-1} [1 - (f_{x+1}^x)]$$

Here, n represents how long the path must be. In other words, n is the number of hops upto which the value of c needs to be calculated.

After the costs to all the paths are calculated, the path with the lowest cost is selected to be the cost of the reaching the destination.

Another protocol presented by Spyropoulos et al. [19] is significant in the area of routing in DTNs. The authors propose a routing scheme that has fewer transmissions than flooding based protocols and deliver a message faster than an existing single or multi-copy scheme. There were two algorithms proposed for this:

1. *Spray and Wait Protocol* : This protocol works in two phases:

    - *Spray Phase* : A limited number of copies, say L is spread in the network by the source node and some other nodes which have received copies from the source.

An important consideration here will be the number of copies that are spread and the neighbors to which the copies of messages are forwarded.

- *Wait Phase* : After the spreading of all the copies of the message is done, and if the destination has not received the message, then each node that has the message tries to deliver the message to the destination by direct transmission independently.

2. *Spray and Focus* : A limitation of the Spray and Wait protocol is that, once all the copies of the message are spread to some nodes and the wait phase starts, but if the mobility of each node is restricted to a small local area, then it may not be possible to deliver one of the copies to the destination. This algorithm attempts to overcome this limitation [20]. It has two phases of operation:

   - *Spray Phase* : For every message originating at a source, L copies of the message are spread in the network to L nodes.
   - *Focus Phase* : Once the spraying phase is done, the nodes start to roam around to find the destination. Each copy in a single node is routed to a closed node via a utility based scheme. If $U_X(Y)$ denotes the utility of node X for node Y, then a node having a message for node D (with immediate neighbors node A and node B) forwards it to a new node B in its range if and only if $U_B(D) > U_A(D) + U_{th}$. U_{th} is the utility threshold parameter.

Moeller et al.[15] presented the Backpressure Collection Protocol for sensor networks. In this protocol, when the forwarding queue is non-empty, weights are calculated for each link using the following equation:

$$w_{(i,j)} = (\delta Q_{(i,j)} - V * ETX_{(i,j)}) * R_{(i,j)}$$

Here $w_{(i,j)}$ is the link weight, $\delta Q_{(i,j)}$ is the difference in queue length between i and j, V is the parameter that trades system queue occupancy for penalty minimization, $ETX_{(i,j)}$ is the expected time of transmission between i and j for a packet to get delivered as described in [5] and $R_{(i,j)}$ is the estimated link rate.

Dvir and Vasilakos [6] suggested using the Packet Reception Rate (PRR) as a measure of the link rate. They proposed the following forwarding decision making policy.

$$w_{(i,j)} = (\delta Q_{(i,j)} - \frac{V}{PRR_{(i,j)}})$$

Here $w_{(i,j)}$ is the link weight for the link between node i and node j, $\delta Q_{(i,j)}$ is the difference in the queue lengths of i and j, PRR is the Packet Reception Rate between the two nodes which represents the channel condition between the two nodes. It is assumed that the two nodes are capable of exchanging queue length information before a forwarding decision is made by the sending node.

PRR takes into account the distance between the neighbors, power, path lost etc. When PRR is taken into consideration, the "bad" channels get a low link weight and thus have lower probability of transmitting the packet. PRR can be calculated using the following formula as proposed in [24]:

$$PRR(d) = (1 - \frac{1}{2}^{(-\frac{\gamma(d)}{2}\frac{1}{0.64})})^{\rho 8 f}$$

where d is the transmitter-receiver distance, γ is the signal to noise ratio, ρ is the encoding ratio and f is the frame length.

Once the source node calculates the $w_{(i,j)}$ for all its immediate neighbors, the message at the head of the queue is forwarded to the node with the highest value of $w_{(i,j)}$.

This algorithm showed an improvement compared to the Collection Tree Protocol [8]. Also, the authors empirically demonstrated that the Backpressure Collection Protocol had superior delivery performance even in dynamic network conditions.

Ying et al. [23] presented a new routing/scheduling algorithm based on the backpressure protocol that guarantees network stability and minimizes the average path lengths between the source and destination. Ryu et al.[18] presented a backpressure routing algorithm for a network with groups or clusters of nodes intermittently connected via mobile carriers (the carriers provide connectivity over time among different clusters of nodes). Pujol et al. [17] presented the Fair-Route, a routing algorithm for DTNs inspired by the social processes of the perceived interaction strength, where messages are forwarded to users that have a stronger social relation with the target of the message. This limits the exchange of messages to those users with similar social status.

A more detailed routing protocol for routing in socially selfish networks was proposed by Li et al. [13]. There are three steps to this protocol design:

- When a node receives and buffers a packet, a priority p is assigned to the packet. This is stated as :

$$p_i = p_{i-1} * \omega$$

  Here, p_i is the priority of the packet in the ith hop and ω is the ith hop's willingness to forward the packet. The initial priority p_0 is set by the node where the packet originated.

- Next, the delivery probability is estimated using the equation:

$$P_{delivery} \geq (1 - P(t_{exp} \leq t_c))(1 - P(t_{over} \leq t_{exp}))$$

Here, $P_{delivery}$ is the overall delivery probability, t_{exp} is the expiration time for the packet, t_c is the time when the nodes will come in contact, t_{over} is the time at which there will be a buffer overflow at the node where the packet is buffered. $P(t_{exp} \leq t_c)$ is the probability that the packet expires before the nodes come in contact. $P(t_{over} \leq t_{exp})$ is the probability that the buffer overflows before the packet expires.

The probability that the packet is dropped because it expired is denoted by P_{exp}. This is calculated as:

$$P_{exp} = \frac{P(X > t_{exp} - t) \leq E(X)}{(t_{exp} - t)}$$

Here, X is a random variable that denotes the inter-contact time between source and destination, *E(X)* is the mean of X and t is the most recent contact time between the source and destination.

The probability that a packet is dropped because of buffer overflow is given by:

$$P_{over} = \frac{|S_{drop}|}{|S|}$$

Here, S is the set of similar packets and S_{drop} is the subset of S that is dropped.

- The node then calculates the subset of neighbors to transmit the message and the order in which to forward the packets to the neighbors. If a neighbor does not have sufficient buffer space, then the packet is not forwarded to that neighbor. Then the sending node tries to maximize its *selfish gain* through this contact.

In all the studies that we have surveyed here, there is no discussion of a decision based on the forwarding probability for the message after it has been sent to a neighbor. We propose a novel algorithm where we use information about the queue length and the position of the packet in the queue of the neighbor for making routing / forwarding decision. This algorithm aims to reduce the latency in the network and the buffer time while sustaining the delivery ratio.

Chapter 3

The Look-ahead Protocol

In this chapter, we discuss the Look-ahead Protocol and its variants. We will first list a few assumptions that we make and then move to the actual description of the protocol.

We assume that there are M nodes moving on a $\sqrt{N}$ by $\sqrt{N}$ 2D torus finite lattice. All the nodes move according to some stochastic mobility model. The meeting times of the nodes are approximately exponentially distributed. It has been shown that some of the popular mobility models like Random Walk [9], Random Waypoint [11] and Random Direction [2] exhibit (approximately) exponential meeting probability [6]. There is no contention in the network. We assume that any communicating pair of nodes do not hinder communication at the same time between another pair of nodes. The nodes do not have unlimited buffer space, old messages are dropped to make room for new ones if there is a buffer overflow. All the nodes in the environment have a queuing model of priority queuing, where the packets in the queue are sorted in decreasing order of priority. At any point of time, the packet in the queue with the highest priority is at the front of the queue.

3.1 Algorithm for the Look-ahead Protocol

As discussed before, since there is a lack of instantaneous end-to-end paths between a source and a destination in DTNs, there is no possibility of knowing the path traversed by a packet before the packet reaches the destination. The concept of *look-ahead* makes the forwarding decision based on the priority of the packet that needs to be forwarded and the queue of the neighbor.

```
priority = Priority of message to be forwarded;
count = 0 ;
while Current node has an unchecked node do
    Get a list of messages of the neighbor;
    while List is not fully traversed do
        temp = Next Message in the List;
        if temp.priority > priority then
            increment count
        else
            continue traverse
        end
    end
end
Forward packet to neighbor with lowest count
```

Algorithm 1: The Look-ahead Protocol

Descriptively, every time a node has a message that needs to be forwarded, the following actions happen at the node:

- The priority of the packet that needs to be forwarded is obtained from the packet header
- In each of the immediate neighboring nodes, the queue information, that is the priority of the packets in the queue, is fetched
- In the queue of each node, the number of packets that have a priority greater than the packet to be forwarded is counted

- The packet is forwarded to the node that has the least number of packets (or no packets) with a priority greater than that of the packet that needs to be forwarded

In this algorithm, it is a possible scenario that a packet is forwarded to a neighbor that has a queue length larger than that of the other neighbors. This can be demonstrated better by an example. Consider a simple three node network as shown in Figure 3.1

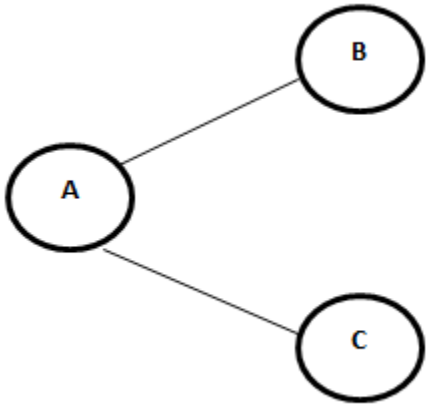

Figure 3.1: A three node network

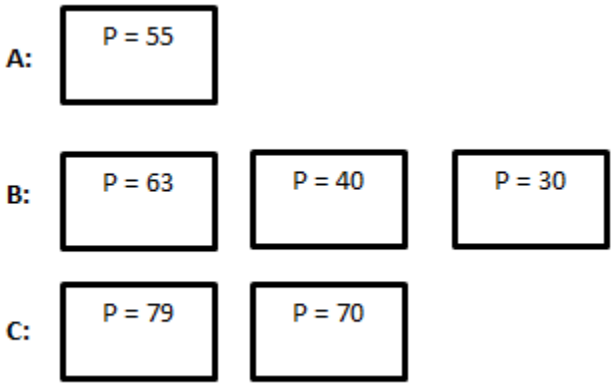

Figure 3.2: Message Queue Status for Network in 3.1

In this scenario we have three nodes A, B and C. The queues in the nodes are as shown in the figure 3.2. If A has to transmit the packet that is at the front of the queue, it will

check the queue status of both its immediate neighbors, B and C. In B, the number of packets with priority greater than the packet to be forwarded(55) is 1. In C, the number of packets with priority greater than the packet to forwarded is 2. Hence, although the queue length of C is lesser than that of B, packet is forwarded to B since the packet will be forwarded by B quicker than by C.

When Look-ahead protocol is used in nodes, sometimes, the path traversed by a packet from source to a destination is not the intuitive path. The path traversed is generally longer. For example, let us consider the scenario in figure:

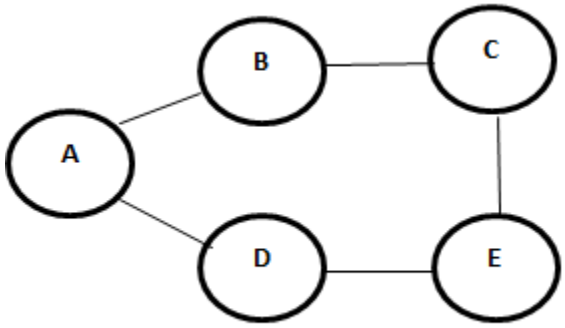

Figure 3.3: A five node network

If node A has a packet to send to node E, the two paths available are $A \rightarrow D \rightarrow E$ and $A \rightarrow B \rightarrow C \rightarrow E$. Intuitively, we would expect that the path that the packet traverses is $A \rightarrow D \rightarrow E$ since it is the shortest path and hence the path that has the least delay from node A to node E. However, this assumption does not consider the time that the packet is buffered at the intermediate nodes. The time that a packet is buffered adds a significant amount of delay that is experienced by the packet in going from the source to the destination. The protocol that we propose aims to reduce the amount of time that a packet spends in the buffers of intermediate nodes.

For the network scenario in the Figure 3.3, let us consider the following queue scenario at each node:

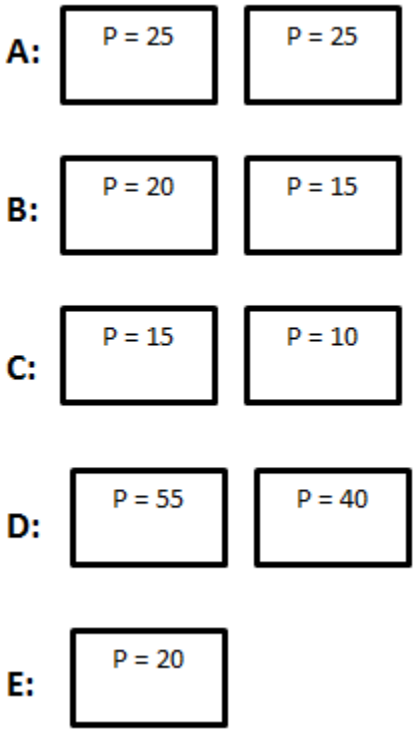

Figure 3.4: Queue Status for the Network in 3.3

If node A has to send the packet with priority 25 to node E, using the Look-ahead protocol, it first checks the status of the queues in its immediate neighbors, node B and node D. In node B, there are no packets that have a priority greater than 25. In node D, there are two packets that have a priority greater than 25. Hence, the packet will be forwarded to node B. As the node uses the information of the neighboring node at the time of transmission, this protocols fits the definition of an opportunistic protocol. From node B, using the Look-ahead protocol, the packet is forwarded to node C and then to E.

One of the issues that can arise from a queuing model like this is that, if a packet has a low priority, then, there is a possibility that the packet is never delivered. The packet

may be deleted from the buffer to make room for a new packet or is dropped because its Time To Live (TTL) expired. To avoid this from happening, another parameter is added to the packet. This is the *delay in queue* factor. Periodically, the priority of the packets in the queue of every node is updated. The frequency at which the update is performed is a parameter that is specified with the system.

$$priority_{new} = priority_{old} + delayinqueue$$

Updating the priority with the queue delay ensures that the packets with low priority are not dropped or lost.

To explain the second issue, let us consider the five node network in Figure 3.3. For this network, let us consider the queue status at each node as shown in Figure 3.5:

If node A has a packet that has to be sent to node E. The immediate neighbors are node B and node D. Node B has no packets with priority greater than 25. Node D has two packets with priority greater than 25. Hence, the packet is first forwarded to node B.

At node B, the immediate neighboring nodes are node A and node D. Node B performs the check for queue status. In node A, there are no packets that have a priority greater than 25. In node D, there are two packets that have a priority greater than 25. Thus, node B forwards the packet to node A. This leads to an infinite loop where the packet is continuously going back and forth between node A and node B. The solutions to this issue are discussed in the next section.

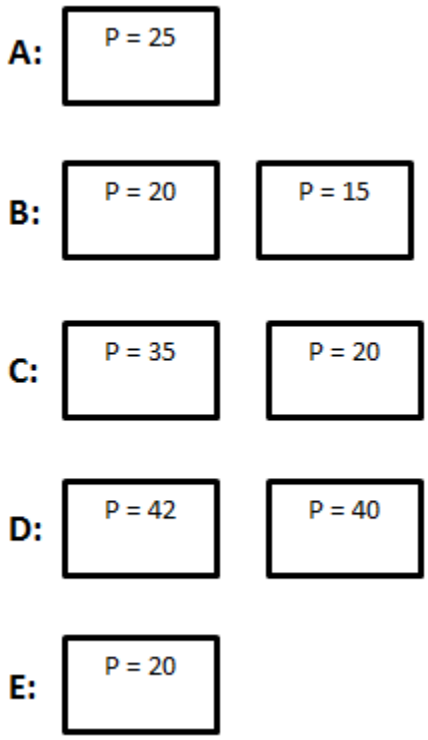

Figure 3.5: Queue Status for the Network in 3.3

3.2 Random Node Selection - A Variant

In the Look-ahead protocol, we discussed a scenario where a packet can be stuck moving back and forth between two nodes. To address this issue, we propose the following two solutions:

- Firstly, before forwarding a message to a node, a check is performed to ensure that the packet was not received from the destination node. This means, the protocol checks that the packet is not sent back to the same node from which it was received. In the above example, if this condition is applied, node B will first check if the packet came from node A. If it did (which is true in this case), then the packet is not forwarded to that node.

- A second solution to this issue would be a random selection of neighboring nodes to forward the packet.If there are two or more neighboring nodes that have the same link weight or have the same position for the packet in their respective queues, then, among the nodes with equal weight, one is chosen at random. This reduces the probability that the packet goes back and forth between the same two nodes.

To make the random selection variant of the look-ahead protocol more general, we can state the condition of link weight assignment as follows:

$$w_{(i,j)} = n$$

if $QueuePosition > 2^{(n-1)}$ and $QueuePosition < 2^n$

Here, *QueuePosition* is the number of packets in the neighbor's queue that have a priority greater than the packet to be forwarded.

Consider the example in Figure 3.4. At node B, the immediate neighboring nodes are node A and node C. Node B performs the check for queue status. In node A, there are no packets that have a priority greater than 25. In node D, there are two packets that have a priority greater than 25. Thus, node B forwards the packet to node A. This leads to an infinite loop where the packet is continuously going back and forth between node A and node B.

When random selection is used, then, the link weights for both node A and node D is 1. In this case, one of node A or node D is selected. This reduces the probability that the packet will be stuck going back and forth between node A and node B.

In the simulation of the protocols, this method of choosing neighbors randomly has proved that the probability of a packet getting stuck going back and forth between a pair of nodes is indeed reduced.

3.3 Double Look-ahead - A Variant

In both the previous versions of the protocol, the forwarding decision of the sending node does not have any knowledge about the one-hop neighbors of its own neighbors. This sometimes can lead to a reduced efficiency of delivery.

Consider a network scenario like in Figure 3.6 and the respective queue status like in Figure 3.7,

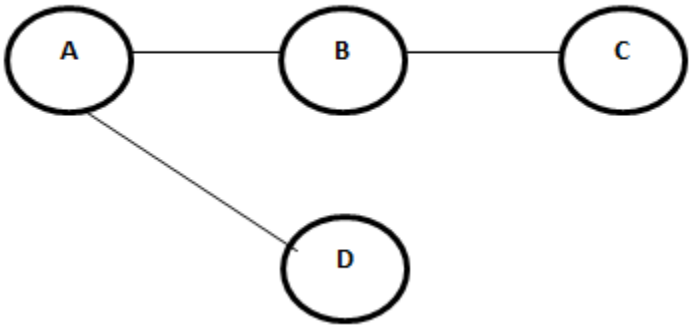

Figure 3.6: A four node network

Using either the Look-ahead protocol or the protocol with random node selection, the node that will be chosen to forward the packet from node A is node D. In this case, since there is no other path forward from node D. Since the protocol does not send the packet back to the node that the packet originally came from (in this case, the packet is not forwarded to node A from node D), the packet is queued in node D. If the message buffer of node D

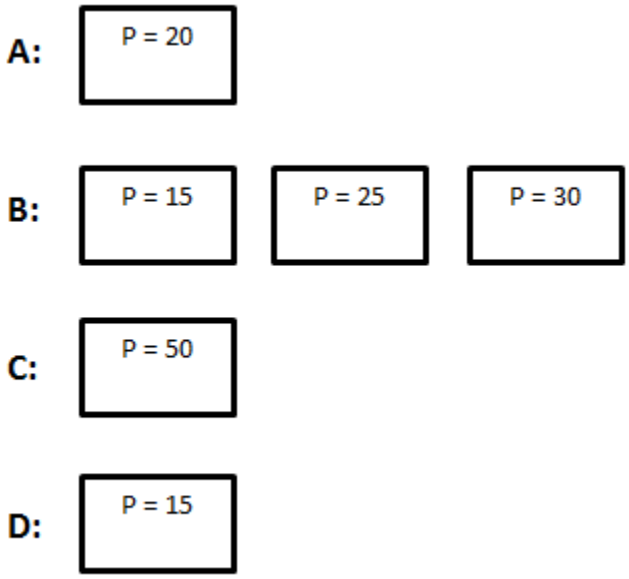

Figure 3.7: Queue Status for the Network in 3.6

is full the packet is dropped or it is deleted once its TTL expires.

To reduce the probability of a packet being dropped from a condition like the one above, we propose a variant of the look-ahead protocol. In this, when a node has to make a decision to forward the packet, the node also gets the one-hop neighbors of the neighboring node. From this information, the node determines whether the final destination of the packet is an immediate neighbor of one of its own neighbors. In other words, it checks if the final destination of the packet is two hops away. If it finds that the destination of the packet is a one-hop neighbor of one of its neighbors, the neighbor is assigned a very high link weight so that the packet is forwarded to this neighbor. When a packet is forwarded from a node, before any protocol is used, the node checks if the packet at the front of the queue has one of its immediate neighbors as its destination. If it does, then the packet is immediately forwarded to this node without applying the routing protocol.

```
priority = Priority of message to be forwarded;
while Current node has an unchecked node do
    Get a list of neighbors of the neighbor;
    Check if destination of the packet is a neighbor;
    if destination is not a neighbor of current node then
        Get a list of messages of the neighbor;
        while List is not fully traversed do
            temp = Next Message in the List;
            if temp.priority > priority then
                increment count
            else
                continue traverse
            end
        end
    end
end
Forward packet to neighbor with lowest count
```

Algorithm 2: The Double Lookahead Protocol

In the example network in Figure 3.6, if node A has to forward the packet to node C, it first checks if either node B or node D has node C as an immediate neighbor. In this case, since node C is an immediate neighbor of node B, the packet is forwarded to node B even though node D has just one packet that has a priority greater than the packet being forwarded.

3.4 Backpressure with Look-Ahead

Moeller et al. [15] proposed the Backpressure Collection Protocol to make forwarding decisions.

$$w_{(i,j)} = (\delta Q_{(i,j)} - V.ETX_{(i,j)}) * R_{(i,j)}$$

$w_{(i,j)}$ is the link weight assigned to each immediate neighbor, $\delta Q_{(i,j)}$ is the difference in queue lengths for node i and node j, $ETX_{(i,j)}$ is the expected time of transmission between node i and j, $R_{(i,j)}$ is the link rate and V is a parameter that trades system queue occupancy for penalty minimization. The packet is forwarded to the neighbor with the highest value of the link weight.

Dvir and Vasilakos [6] suggested using the Packet Reception Rate to assess the channel quality and eliminate the "bad" channels. The policy that they used for making the packet forwarding decision is represented by this equation:

$$w_{(i,j)} = (\delta Q_{(i,j)} - \frac{V}{PRR_{(i,j)}})$$

$w_{(i,j)}$ is the link weight assigned to each immediate neighbor, $\delta Q_{(i,j)}$ is the difference in queue lengths for node i and node j, V is a parameter that trades system queue occupancy for penalty minimization and $PRR_{(i,j)}$ is the Packet Reception Rate between the two nodes.

In the protocol that we propose, we look to include the look-ahead factor to this back-pressure routing equation. The look-ahead factor $LA_{(i,j)}$ is information obtained from the neighbors about the number of packets in their queue which have a priority greater than the packet that needs to be forwarded. The equation can be written as:

$$w_{(i,j)} = (\delta Q_{(i,j)} - \frac{V}{PRR_{(i,j)}}) * LA_{(i,j)}$$

$w_{(i,j)}$ is the link weight assigned to each immediate neighbor, $\delta Q_{(i,j)}$ is the difference in queue lengths for node i and node j, V is a parameter that trades system queue occupancy for penalty minimization, $PRR_{(i,j)}$ is the Packet Reception Rate between the two nodes

and $LA_{(i,j)}$ is the look-ahead factor calculated for that node. The look-ahead factor here is the number of packets in the queue of the neighbor that have a priority lesser than the packet that has to be forwarded.

When the link weight is calculated for each node, the number of packets that are forwarded before the current packet is considered. Adding this parameter to the forwarding policy ensures that the packet is not queued for a very long time. In a neighboring node, if there are more packets that have a priority lesser than that of the packet that needs to be forwarded, then, the link gets a higher weight. If the queue length is very long or there is a high difference in queues of the two nodes, then, that too is accounted for in the equation with the $\delta Q_{(i,j)}$.

```
priority = Priority of message to be forwarded;
while Current node has an unchecked node do
    Get a list of messages of the neighbor;
    while List is not fully traversed do
        temp = Next Message in the List;
        if temp.priority < priority then
            increment count
        else
            continue traverse
        end
    end
    Calculate link weight for current neighbor;
end
Forward packet to neighbor with highest link weight;
```

Algorithm 3: Backpressure Protocol with Look-ahead

3.5 Implementation Specifics

When this protocol has to be deployed in a real world scenario, some of the specifics have to be altered.

- When the information about the position of the packet has to be determined, the sending node cannot process this information. The sending node will not have access to the queue information of its neighboring nodes. The sending node should first send a *Request message* to the neighbor. Once the neighbor receives this message, it appropriately calculates the position of the packet in its queue. The result is sent back to the sending node using a *Reply message*. The sending node collects all (or a percentage) of the replies from its neighbors before the forwarding decision is made.

- The *Request* message is used by the nodes to send a request to its neighbors. This message contains the priority of the packet to be forwarded. To specify this piece of information, a new field has to be added to the packet structure. We call this the *la_request* field. The sending node puts the priority of the packet to be forwarded in this field. When a node receives a *Request* message, it checks this field to determine the position of the packet in its queue.

- Once the position of the packet in the queue is determined, this information has to be sent back to the sending node. This is done using the *Reply* message. We use the *la_reply* field to communicate this information. The position of the packet in the neighbor's queue is added in this field. When a node receives a *Reply* message, the *la_reply* field is checked for the queue position.

- The *Request* and the *Reply* messages should have the highest priority. To ensure this, the regular data packets are assigned a priority only between 1 - 97. The *Request* message is given a priority of 98 and the *Reply* message is given a priority of 99. When the priorities of the data packets are updated periodically, it is ensured that the packet is assigned a priority not higher than 97.

$$priority_{new} = max(priority_{new}, 97)$$

3.6 Summary of Contributions

In this chapter, we have introcuded a novel protocol for routing in DTNs, the Look-Ahead Protocol. This protocol is novel in the sense that it gets information from its neighbor about its queue before a forwarding decision is made.

The look-ahead protocol first determines the priority of the packet to be forwarded. Then it determines all the nodes whicg are within transmission range at that instant. In each neighbor's queue, it checks the number of packets that have a priority greater than that of the packet the nodde needs to forward. Once it has this information from all the nodes that are in transmission range, it forwards the packet to the node with the *least* number of packets of a higher priority. This way, the protocol ensures that the packet does not stay in the buffer of a single node for a long time.

This protocol inherently forwards packets that are of high priority. Sometimes, packets with lower priority get stuck in the queues for ever and are eventually dropped to make room for newer packets. To prevent this from happening, the *delay in queue* factor is used to update the priority of the packets so that lower priority packets are also forwarded.

In certain scenarios, packets can get stuck moving back and forth between a pair of nodes thus having a negative impact on the overall behavior of the protocol. To prevent this, we propose three variants of the protocol.

In the first variant, the look-ahead factor for all the immediate neighbors is determined. Once the information is available for all the nodes, the information is put into *buckets*. Each bucket has a range of 2^{n-1} to $2^n - 1$, where n is the bucket number. Once the information is in the buckets, the protocol chooses the non-empty bucket with the *lowest* value of n. From this bucket, it chooses a random node to forward the packet to.

The second variant adds an additional check for the forwarding policy. While checking in an immediate neighbor about the number of packets in the queue with higher priority, the node also checks if the final destination of the packet is an immediate neighbor of one of the neighbors. If the node finds that one of the immediate neighbors is one hop away from the final destination, then, the packet is forwarded to that node irrespective of the queue information of that node.

The last variant adapts the backpressure protocol for making the routing decision. We use the variant of the backpressure equation that uses packet reception rate as an estimate of the channel quality. Once the node has the information from its immediate neighbors about the number of packets in their queue with higher priority, then, it multiplies this number with the backpressure equation to calculate the link weight for each neighbor. It forwards the packet to the node with the highest link weight.

Chapter 4

Simulation Results and Discussion

In this chapter, we discuss the simulation of the Lookahead protocol and its variants. We have used the ONE (Opportunistic Networking Environment) simulator to test the performance of these protocols. We first describe the simulator and then proceed to an analysis of the data we have collected from the simulations.

4.1 The ONE Simulator

The ONE simulator [12] is a Java based tool that has a broad range of DTN protocol simulation capability. This simulator supports mobility and event generation, DTN routing and application protocols, visualization and analysis interfaces, importing and exporting mobility traces and a basic notion of energy consumption at each node. The ONE simulator is very modular. This allows for extending almost all the functions using well defined interfaces. The differentiating factor of this simulator is modeling the store-and-forward paradigm of routing.

The main functions of the ONE simulator are the modeling of the node movement, inter-node contacts, routing and message handling. The source code for the simulator can be downloaded from [1].

4.1.1 Nodes

A node is a basic agent in the simulator. It is a mobile endpoint that is capable of store-and-forward paradigm of routing. Each node has a radio interface, persistent storage, movement, energy consumption and routing. The radio interface and the storage are configured through parameters that are provided from an external text file. Movement and routing are configured through specialized modules that implement a particular behavior for the capability.

4.1.2 Node Movement

Node movement is implemented through mobility models. Mobility models define the algorithms that generate the node movement paths. The three types of synthetic movement that are included are:

- Random Movement
- Map-constrained random movement
- Human behavior based movement

The ONE simulator includes the implementation of the Random Walk[9] and the Random Waypoint[11] movement models. It also includes three map-based mobility models:

- Random Map-Based Movement (MBM): The nodes move randomly but follow the path defined by the map data.
- Shortest Path Map-Based Movement (SPMBM): The nodes choose a random point on the map and then follow the shortest path to that point using the map data.
- Routed Map-Based Movement (RMBM): The nodes have pre-determined routes that they follow.

The ONE simulator also includes the Working Day Movement model which tries to increase the reality of human node mobility.

4.1.3 Routing

The simulator has a framework for defining the algorithms and rules used in routing. It comes with ready implementations of some of the popular DTN protocols. The protocols that are included with the package are: Direct Delivery, First Contact, Spray-and-Wait, PRoPHET, Max-Prop and Epidemic.

To add a new protocol to the ONE simulator, a new routing module needs to be added for the protocol. The new module can inherit the basic functionality like simple buffer management and callbacks for message-related events from the *MessageRouter* module. The basic functionality for all the protocols is common for all the currently implemented protocols with internal routing logic. This can be re-used using the *ActiveRouter* module. This module provides functionality like checking for messaged that have to be delivered to immediate neighbors and dealing with message transfers and aborts.

4.1.4 Application Support

In the ONE simulator, there are two ways to generate application messages:

- Message Generators: Messages are created with a random or fixed source, destination, size and interval.
- External Event Files: The messages are generated with specified messageID, fixed source and destination and messages generated at a specified time.

4.1.5 Reporting and Visualization

The results collected from simulation in ONE simulator can be visualized using the Graphical User Interface as shown in Figure 4.1 or by generating images using the information gathered in the reports.

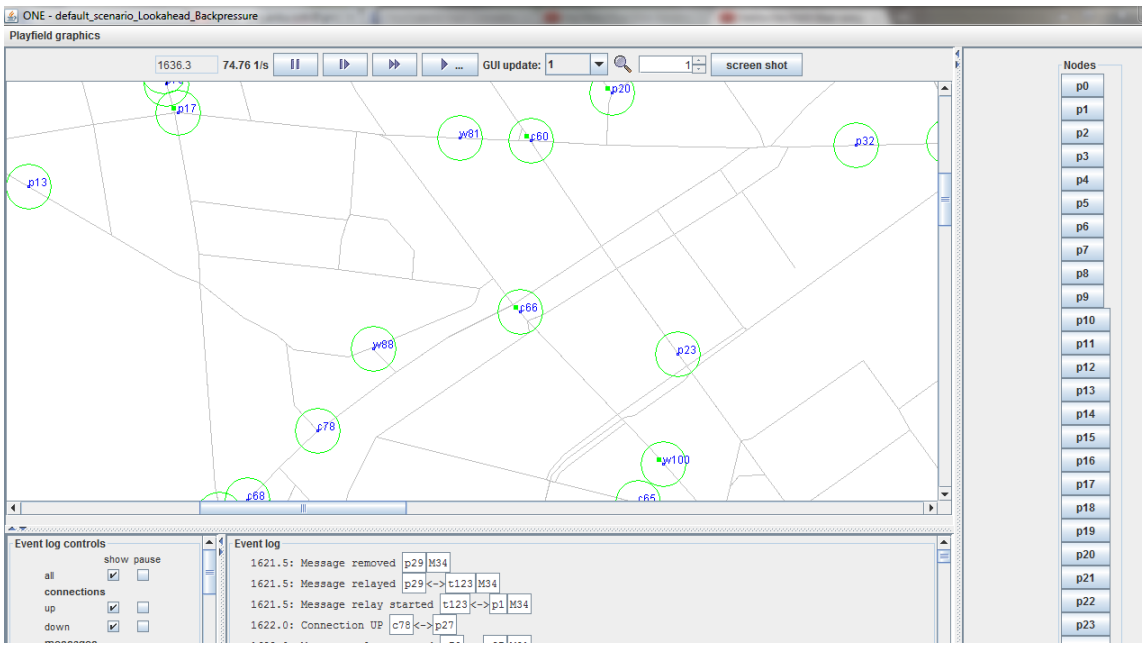

Figure 4.1: Screenshot of the GUI for ONE simulator

In the figure 4.1, the nodes are represented using names like p13, p17, c66, w88 (seen in the figure). The circle around them represents their transmission range. The square block above the name of each node name shows the queue for the node.

The GUI is suitable to get an overall picture of what is happening during the simulation. However, more rigorous information can be seen in the reports collected. It includes a message statistics report that gathers statistics of the overall performance like message delivery ratio, buffer time etc.

The simulator is configured using text based configuration files that contains the simulation, event generation and reporting parameters. This file also has the defining parameters for the nodes like the storage capability, transmit range, bit rates as well as the routing model to use.

4.2 Our Simulation Scenario

The protocols were first tested on a small group of static nodes. Once the correct functioning of the protocols was established, the protocols were simulated on a more realistic scenario with mobile nodes with different moving speeds.

4.2.1 Parameters Explained

For a comparison of the protocols that we have proposed, we compare the protocols on various network parameters. This section gives a brief description about the various parameters that we study and explain.

- Delivery Probability: The delivery probability is a measure of the fraction of the created packets that are delivered to the destination. This is the ratio of the total number of packets that are delivered to their destinations to the total number of packets that are created.

- Overhead Ratio: The overhead ratio is calculated using the following equation:

$$Overheadratio = \frac{(Numberofrelayedmessages - Numberofdeliveredmessages)}{Numberofdeliveredmessages}$$

  Here, the term *relayed messages* refers to the messages that have been forwarded by the source to an intermediate node to be forwarded towards the destination. This number is a measure for the number of packets or copies of packets that have been inducted into the network. The number of *delivered messages* refers to the total number of created packets that are successfully delivered to the destination.

- Latency: The latency measured here is the time that elapses between the creation of a message and its delivery at its destination. We consider the average of the latency of the packets over the entire simulation time. This is the time as calculated for the delivered packets only.

- Buffer time: This is the average amount of time that packets spend in the buffer of the nodes. This is not a measure of the time spent in the buffer by the delivered packets, but it is an average of the time spent by *all* the packets, delivered and dropped, in the intermediate nodes' buffers.

4.2.2 The Static Nodes Scenario

In this simulation, we used a group of 20 nodes that are placed within a map of 100 units * 100 units. Each node has a Bluetooth transmitter. The transmitters are omni-directional with a range of 45 units. This makes the network fully connected, that is, there is a definite path between any two nodes in the network.

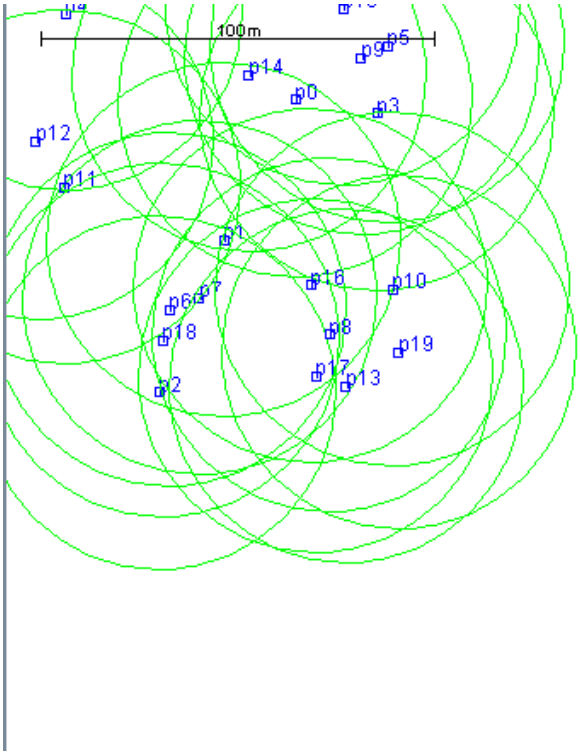

Figure 4.2: ONE Simulator for the Static Nodes Scenario

The figure 4.2 shows the 20 nodes in a 100 * 100 map space. The nodes are labeled p0 through p19.

The table 4.1 shows the value of the delivery ratio, overhead, latency, hopcount and the buffer time for the static nodes scenario.

As seen from the numbers in table 4.1 the delivery probability for the protocols listed is very high and is very close to the 100% mark. This is because the nodes are static and

Parameter	Lookahead	Random Lookahead	Double Lookahead	Backpressure Lookahead
Messages Created	822	822	822	822
Delivered	810	821	815	817
Delivery Ratio	98.54	99.88	99.15	99.39
Overhead	76.0474	12.3337	48.4687	32.5564
Latency (avg)	179.1318	20.0171	74.5521	33.8
Hopcount (avg)	65.0210	13.2887	40.98	41.69
Buffer time (avg)	2.6923	1.5062	1.7643	1.8529

Table 4.1: Comparison of the protocols when the nodes are static

given a source and destination, there surely exists a path between the two nodes. Hence, the packet will be delivered to the destination. The numbers do not read 100% because at the end of the simulation, there will be some packets that are in the buffers of the nodes. If the simulation is extended for some more time, these packets are bound to be delivered to the destination.

We see that the overhead ratio for the Lookahead protocol is rather higher than the other protocols. This behavior can be attributed to the fact that sometimes, given the design of the lookahead protocol, a packet can be stuck going back and forth between two nodes for a very long time. This increases the number of relayed messages in the network considerably. But the number of delivered messages remains the same despite the high number of relayed messages. This increases the overhead in the network. In other words, it increases the amount of network resources that are used to deliver one packet to its destination. In the random lookahead protocol, a change is made to ensure that a packet is not forwarded to the node that it came from. This change ensures that a packet is not stuck going back and forth between two nodes for a long time. The double lookahead protocol does not have this check in place, but adds a further check to see if the destination of a packet is two hops away from the current node. If it is, then, the packet is forwarded towards the

destination hence slightly reducing the number of hops. The backpressure lookahead takes into account many other parameters, hence the overhead is lower.

The latency experienced by the packets is the time that elapses between the time the packet is created and the time the packet is delivered to the destination. This parameter is high for the lookahead protocol for the reason that the packet can sometimes go back and forth between two nodes. This increases the amount of time that a packet spends between its creation and the time it is delivered at its destination. The numbers for this are lesser for the random lookahead protocol because it ensures that the packet is not forwarded to its source again. This reduces the latency of the packets. The double lookahead protocol shows an increased reading for this parameter because the packet tends to stay in the buffer longer. When it is at a two hop distance from its destination, the packet is forwarded towards the destination regardless of the queue information. Sometimes, this can increase the amount of time that the packet spends in the buffer of the penultimate node, hence an increase in the latency.

The hop count for the lookahead protocol is high again due to the packets going back and forth between two nodes. This increases the average hop count as the packet can be stuck between two nodes for a considerable amount of time. The hop count for random lookahead is reduced drastically owing to the check that it performs that ensures the packet does not go back to the source. The double lookahead protocol has a higher hop count due to the packet going back and forth between two nodes. It has a lower hop count compared to the lookahead protocol because it checks if the destination is a two hop neighbor of the current node. This reduces the hop count to a certain extent.

The buffer time of the lookahead protocol is the highest among all the protocols again owing to the packets sometimes going back and forth between two nodes. The random

lookahead protocol has the lowest buffer time among all the protocols as it eliminates the scenario where the packet goes back and forth between two nodes. Since this check is not performed by the double lookahead protocol, the buffer time is higher. It is lower than the buffer time for the lookahead protocol because it has the check to see if the destination of the packet is a two hop neighbor of the current node and forwards the packet in the direction of the destination. This reduces the buffer time to an extent, but can sometimes end up in the packet being in the buffer of the penultimate node for longer hence increasing the buffer time.

4.2.3 The Moving Nodes Scenario

In the moving nodes scenario, we have tried to simulate the protocols in a scenario as close as possible to a real world scenario. We have 6 groups of nodes each with 20 nodes. All the nodes have a Bluetooth interface. Each node has a buffer size of 5 MB. The movement is based on the Shortest Path Map Based Movement. The Time To Live (TTL) for each packet is 300 time units. The queuing model used is priority queuing, where the packets are sorted in descending order of their priority. The map used is 4500 units * 3400 units.

The first and third groups have pedestrians walking with speeds between 0.5 to 1.5 km/h. The second group has cars moving at 10 - 50 km/h. The fourth, fifth and sixth groups move on a tram with a speed of 7 - 10 km/h. They have only 2 hosts and they have a high speed interface along with the Bluetooth interface. They have a message buffer of 50 MB. They use a Route Based Movement. All three groups use maps defined in different map files for different routes.

The results obtained for the protocols is discussed in subsequent sections.

Time	Epidemic Router	Lookahead	Double Lookahead	Random Lookahead	Backpressure Lookahead
20000	45.19	46.81	44.59	46.67	47.26
40000	32.45	55.06	54.99	54.4	54.99
60000	32.15	58.2	57.16	57.9	56.72
80000	31.93	60.61	60.09	61.27	60.09
100000	31.27	62.66	62.01	63.73	62.01
500000	32.48	66.59	66.41	66.66	66.43
1000000	32.63	66.87	66.87	67.15	66.87
2000000	32.44	67.15	66.99	67.16	66.88

Table 4.2: Delivery probability of various protocols recorded with increasing time

4.3 Some Simulation Results

Using the scenario described in the previous section for moving nodes, we simulated the protocols that we propose and compared the numbers to those obtained for the Epidemic Routing protocol. We compare values of the delivery probability, overhead ratio, hop count, latency and buffer time.

4.3.1 Delivery Probability

This is the ratio of the total number of packets created to the total number of packets that are delivered to the destination. This is a direct measure of how reliably packets are routed in the network by the routing protocol.

Table 4.2 gives the values of the delivery probability for the various protocols with increasing time. Figure 4.3 is a plot of the values of delivery probability against time. We can see that the delivery probability of the Epidemic Routing protocol remains constant at about 32 - 33 % irrespective of the length of time for which the protocol is run. However, the delivery probability of the lookahead, random lookahead, double lookahead and the

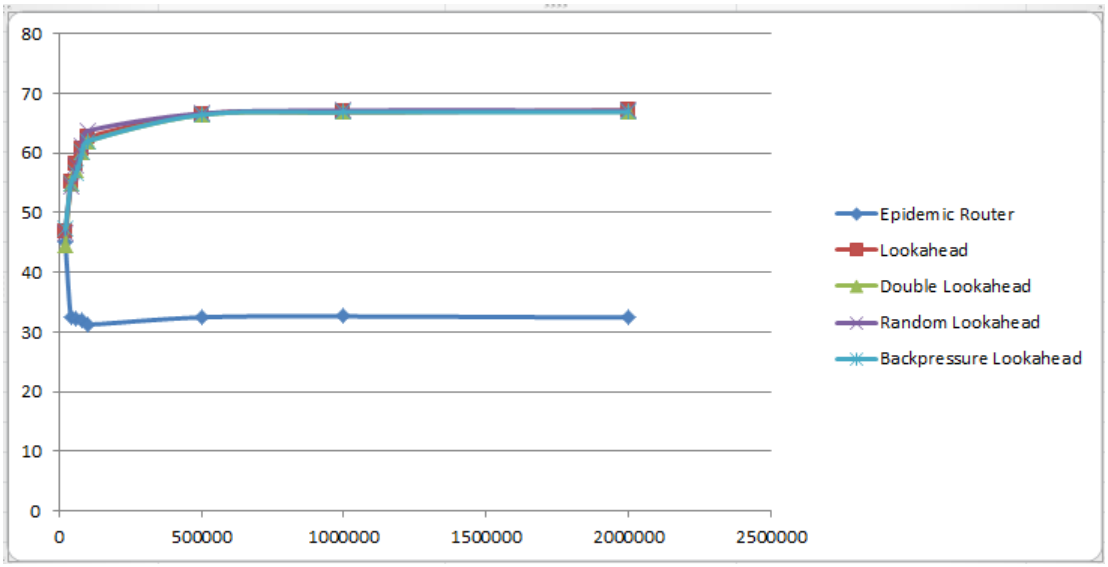

Figure 4.3: Plot of Delivery Probability against Time for the various protocols

backpressure lookahead protocol show similar behavior with increasing time. When the protocol runs for a significant amount of time, the delivery probability is at its maximum value of about 66 - 67 % which is double the value of the Epidemic Router protocol.

In all of the protocols that we propose, the packets are not replicated and forwarded in the network. There is only one copy of the packet in the network at any given point of time before it is delivered to the destination node or dropped from the buffer in an intermediate node. In the simulation scenario, the nodes have a good amount of buffer space. But in the Epidemic Routing protocol, since the packets are duplicated and forwarded to every node that does not have a copy of the packet in its buffer, the buffer queues of the nodes fill up fast. It is a possibility that more messages are dropped due to buffer overflow apart from the messages that are dropped because the TTL expired. This is one of the reasons that the Epidemic Router protocol has lower values of delivery probability.

In the lookahead protocol and its variants, we see that the delivery probability is rather low when the protocol is run for a lower amount of time. The values increase with increasing

Time	Epidemic Router	Lookahead	Double Lookahead	Random Lookahead	Backpressure Lookahead
20000		4670.392	4030.7	3900	3970.8243
40000	2708.3841	5725.0337	5802.7445	5864.4164	5802.7445
60000	2839.0663	6081.5981	6084.7097	5993.8021	6010.7445
80000	2584.587	6192.9743	6210.9221	6306.7574	6210.9921
100000	2885.6737	6308.243	6263.1094	6396.0269	6263.1094
500000	2954.8594	6423.4821	6444.6433	6410.9667	6442.0914
1000000	2944.0333	6426.7551	6443.3828	6434.9043	6443.3828
2000000	2973.4035	6480.3959	6484.6708	6477.0087	6477.5674

Table 4.3: Latency for various protocols recorded with increasing time

time to about 65% and then remain constant when the time is increased. This is because, when the protocol is run for a lesser amount of time, there are some packets in the buffers of the nodes whose TTL has not expired and can be delivered to the destination if the simulation is kept running for more time. When the time is increased significantly, the percentage of the number of packets that are in the buffers which can be delivered if the simulation runs longer is lesser than the number of packets that have been delivered. The number of these packets does not affect the delivery ratio significantly.

4.3.2 Latency

Latency experienced by a packet in the network is the amount of time that elapses between its creation and its delivery to its destination. In most protocols, it is desired that the value of latency is low. However, since we are dealing with DTNs where latency is acceptable, we do not strive to have very low values of latency. Since DTNs have a high latency, it is not very suitable for applications like video and audio streaming.

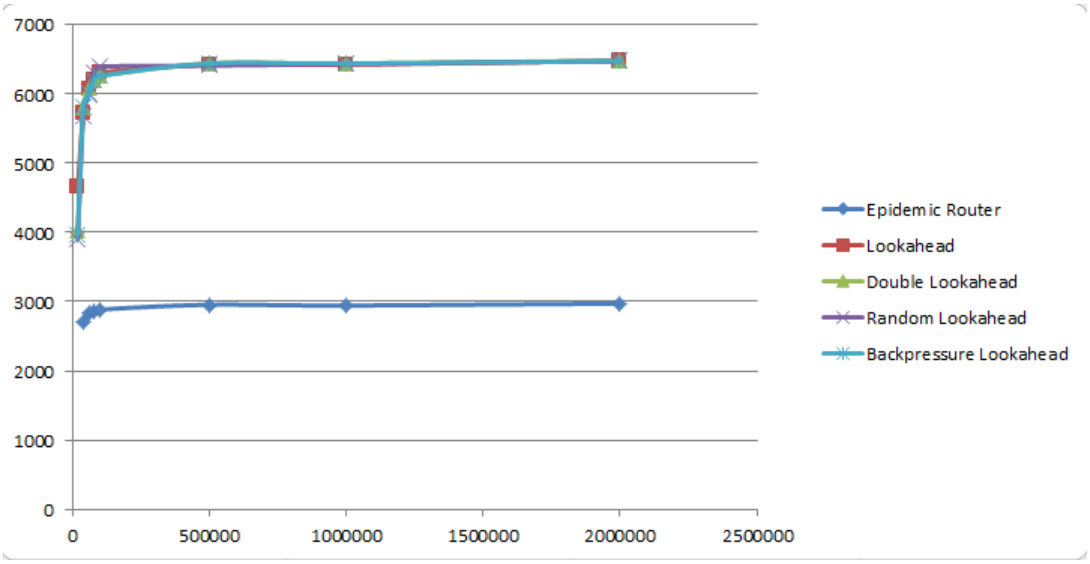

Figure 4.4: Plot of Latency against Time for the protocols

Table 4.3 lists the values of latency as recorded for the protocols in the simulation with increasing values of time. Figure 4.4 is a graphical representation of the trends of the latency values with increasing time.

The average latency experienced by a packet is almost constant in the Epidemic Roouting protocol with increasing values of time. However, the trend is different with the Lookahead protocols and its variants. For lower values of time, the values of latency show a steep increase with increase in time. But when the time is increased beyond 100,000 time units, the values remain fairly constant when the time is increased to very high values.

The reason for the increasing values of latency when the protocol is simulated for lesser time is that, when the simulation stops, then there are a significant number of packets in the buffers of the nodes that can be delivered if the protocol runs for a longer time. The number of such packets is comparable to the number of packets that have been delivered, hence the average value of latency is affected. However, when the protocol runs for a significantly large amount of time, then, the number of packets that are in the buffers of

the nodes is very small compared to the number of packets that have been delivered. The values that we obtain for latency here is affected mostly by the packets that have been delivered to their destinations.

In comparison, the trend we see in the latency values in 4.4 is similar to the trend we see for the delivery probability in 4.3.

The average value of latency for the Epidemic Router protocol is in the range of 2800 - 3000. But the values of latency for the Lookahead protocol and its variants is much higher than the latency seen in the Epidemic Router protocol. In the lookahead protocol and its variants, the packets are not necessarily forwarded in the direction of the destination. In other words, the path that a packet takes to reach its destination may not be the one with the least number of hops, although the packet would have reached the destination in a lesser number of hops. This is not the case in Epidemic Router protocol where the packets are replicated if a neighbor does not have a copy of any of the messages in the sender's queue. Due to the longer path that the packets might take, the latency also increases.

4.3.3 Buffer Time

Buffer time refers to the time that packets spend in the buffers of intermediate nodes. A high buffer time always means that the latency experienced by the packets is high. In other words, if packets are in the buffers of intermediate nodes for a longer time, then they would require more time to reach their desitination.

Table 4.4 lists the average values of buffer time for the various protocols and figure 4.5 shows a plot of how the latency of the different protocols varies with increasing time.

Time	Epidemic Router	Lookahead	Double Lookahead	Random Lookahead	Backpressure Lookahead
20000		105.7134	106.2283	182.85	212.6437
40000	516.627	429.1003	412.9409	659.1105	412.9409
60000	515.1632	749.1855	702.4818	1054.9858	702.4818
80000	516.3134	1021.234	965.3483	1520.0961	965.3483
100000	519.522	1279.0701	1195.7015	1763.6069	1195.7015
500000	513.1976	4022.6026	3723.9386	4540.73	3482.2338
1000000	510.4896	5295.0263	4698.6026	5323.9904	4268.6026
2000000	509.8511	6058.1824	5349.4784	6101.1297	5376.1082

Table 4.4: Buffer tme for various protocols recorded with increasing time

We see that the amount of time spent by the packets in buffers in the Epidemic Routing protocol does not show too much variation with time. However, with the Lookahead protocol and its variants, there is a marked increase in the latency values when the time increases. One of the reasons for the high latency in the lookahead protocols and its variants is that the path traversed by the packets from the source to the destination in the Lookahead protocols and its variants can definitely be longer than the path traversed by the packets when the Epidemic Router protocol is used. The Lookahead protocol does not forward the packet towards the destination, it forwards the packet to the neighbor which will forward the packet quickest. Since the packets traverse a longer path, the amount of time they are buffered in intermediate nodes increases.

Another noteworthy point from the graph in figure 4.5 is that, when the time for which the simulation runs is lower, then, the increase in the buffer time is very steep. But when the time is increased to higher values (multiples of 500,000), then the increase in the buffer time is not very steep. This is again because of the significant number of packets that remain in the buffers of the nodes that can be delivered if the simulation had run for longer. This gives a rather steep increase in the buffer times for lower values of time. But when the simulation is run for a longer time, the number of packets that remain in the nodes of the

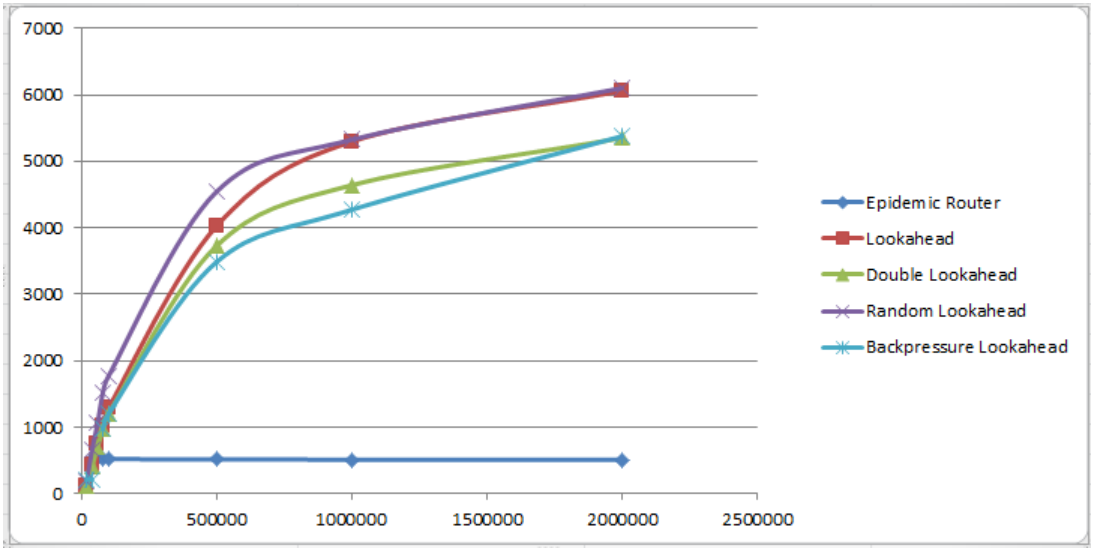

Figure 4.5: Plot of Buffer time against Time for the protocols

buffers of the nodes is not comparable to the number of packets that have been delivered. Hence, the increase in the buffer time is not very steep for higher values of time.

4.3.4 Overhead Ratio

Overhead ratio is defined as

$$Overheadratio = \frac{(Numberofrelayedmessages - Numberofdeliveredmessages)}{Numberofdeliveredmessages}$$

This is a measure of the number of packets that have been introduced into the network to deliver a packet from the source to its destination. The overhead ratio also shows the amount of the network resources needed to deliver a packet to its destination.

Table 4.5 gives the values of the overhead experienced by the network when the routing protocols used are different. Figure 4.6 is a graphical visualization of the variation of the overhead ratio with time.

Time	Epidemic Router	Lookahead	Double Lookahead	Random Lookahead	Backpressure Lookahead
20000	195.035	61.231	69.01	36.92	32.3762
40000	196.7654	26.1423	28.1667	17.163	28.1667
60000	203.5911	16.5567	18.1841	11.2679	19.65
80000	207.7106	12.0354	13.0547	7.5501	13.0547
100000	212.2791	9.382	10.282	6.2818	10.282
500000	213.3763	2.1219	2.417	1.7049	2.6873
1000000	213.9234	1.2704	1.6575	1.2408	1.6575
2000000	215.7183	0.9338	1.2592	0.915	1.249

Table 4.5: Overhead Ratio for various protocols recorded with increasing time

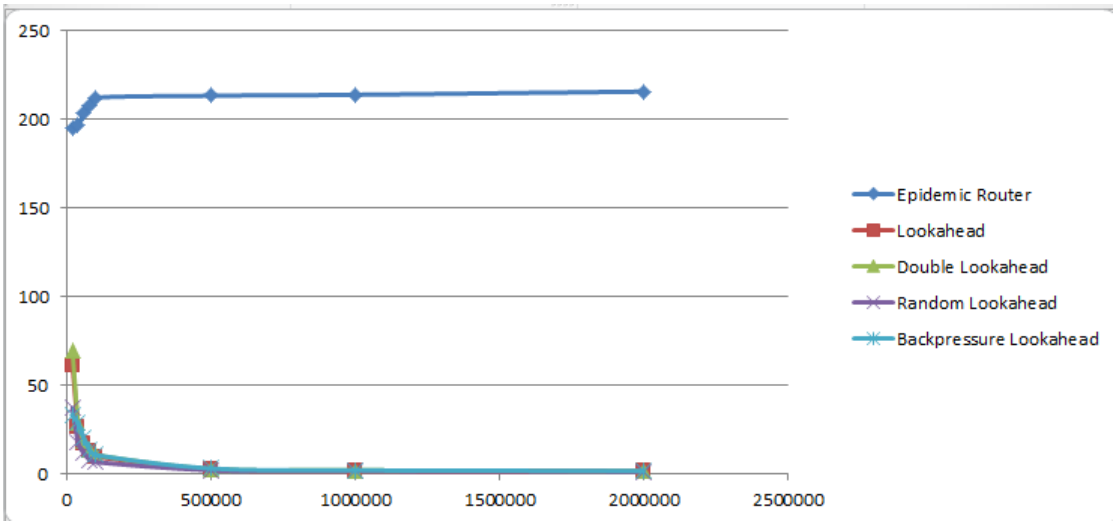

Figure 4.6: Plot of Overhead Ratio against Time for the protocols

By the definition of the overhead ratio, the value of overhead ratio should decrease when the delivery probability increases. This is also the case when we compare the graphs in figures 4.6 and 4.3.

The overhead that is caused by the Epidemic Router is in the 200 - 250 range where as the overhead caused by the Lookahead protocol and its variants is rather low, with the maximum being around 60. This follows from the way the protocols have been designed. Epidemic Router relays a lot more number of packets into the network when compared

to the Lookahead protocols and its variants. A node replicates a packet every time it encounters another node which does not have a copy of the message. This drastically increases the number of times a packet is relayed to intermediate nodes. With the lookahead protocol and its variants, the packet is forwarded only to a specific selected neighbor. The number of times a packet is relayed is the number of hops it takes to reach the destination. Hence, the Lookahead protocol and its variants have lower overhead.

With the protocols, another behavior we have seen consistently for all the protocols is that, the overhead ratio falls steeply when the time of simulation is increased. But for very high simulation times, the overhead ratio remains fairly constant. The reason for the sharp drop is the number of packets that are in the buffers of the nodes that can have been relayed but can be delivered given time, is significant compared to the number of delivered. But when the simulation runs for a very large amount of time, the number of packets that have not been delivered but have been relayed to an intermediate node is not comparable to the number that have been delivered or dropped. Thus, the overhead ratio is fairly constant for larger time.

4.4 Summary

In this chapter, we have detailed the simulations that we have performed to compare the Lookahead protocol and its variants to the Epidemic Routing Protocol. We have found that the Lookahead protocols and its variants have a better delivery probability and a lower overhead ratio in comparison to the Epidemic Routing protocol. However, the Epidemic Rouing protocol gives better latency and lower buffer time for the packets in the network.

Our protocols were proposed so that the amount of bandwidth or network resources used be low. The values that were obtained for the overhead ratio gives good proof that the Lookahead protocols and its variants do not consume a large amount of network bandwidth.

Chapter 5

Conclusions and Future Work

In this thesis, we have proposed a new protocol, the Look-ahead protocol. In addition to this, we have also proposed three variants of the look-ahead protocol: the Random Look-ahead protocol, the Double Look-ahead protocol and the Backpressure Look-ahead protocol. We have simulated these protocols on the ONE simulator.

5.1 Conclusions

From the results that we obtained from simulating the protocols using the ONE simulator, the following are a few conclusions that arise from the numbers:

- The Look-ahead protocol and its variants show a significantly higher delivery probability than the Epidemic protocol. Among the variants of the look-ahead protocol, the random look-ahead protocol shows a higher delivery probability.

- The overhead that the look-ahead protocol and its variants introduce in the network is also significantly lower than that of the Epidemic protocol. This can be attributed to the difference in the number of copies of a message that are in the network. In the case of the look-ahead protocol and its variants, there is only one copy of the message in the network, there are no duplicates. However, the Epidemic protocol replicates a message every time there is a node that does not have a copy of the packet. This introduces high amount of overhead on the network. This is also taxing on the network as the nodes have limites buffer space and messages can get dropped due to non-availability of buffer space.

- The buffer time for the look-ahead protocol and its variants is significantly higher than that of Epidemic routing protocol. The path traversed by the packets in the look-ahead protocol and its variants is significantly longer than the hops taken by the packets in Epidemic routing protocol. This is the reason that the amount of time spent by the packets in the buffers of the intermediate nodes is higher for the look-ahead protocol and its variants.

- The latency experienced by the packets is also high when the look-ahead protocol or one of its variants is employed. Since the packets tend to take longer paths to the destination, the time that elapses between the creation of the message and the delivery of the message at the destination is high. Higher latency is acceptable in DTNs.

5.2 Future Work

There are many aspects of the protocol simulations and the results that remain unexplored. We enlist a few of those here:

- The protocols have been simulated on the ONE simulator. They have not been deployed on a real network. It is of foremost importance that the protocol be tested out on a real network. The following assumptions that we make in the simulator may deviate the results from the values we see here.
  - We assume that there is no contention in the network for resources. This is not always the case. When there are nodes in close proximity, there will be a contention for the network bandwidth. There can also be interference in the transmission that is not accounted for in the simulator's environment.
  - The assumption that there will be no interference from nodes in close proximity may not hold good in a real network. The nodes that are physically close to each other may see some transmission interference.
  - The messages in the ONE simulator do not have any payload. The buffer sizes in the nodes are now enough to accommodate a large number of messages. However, when the packets are sufficiently large, then the buffer overflow may cause a bottlenck in the network that can lead to a lower delivery ratio.
- With the current design of the protocol, the amount of time that the packets spend buffered in intermediate nodes is rather high. The reason behind this is the lengthy paths that are traversed by the packets. Modifications can be made to the design of the protocol to ensure lesser buffer time for the packets.

- The look-ahead protocol and its variants assume that the nodes have infinite energy. They do not account for the energy loss per transaction to the neighboring nodes. We have assumed that the nodes use a Bluetooth interface for communication, which in reality is intensive on battery usage. Similarly, energy is expended every time a calculation is performed to decide which neighbor a packet should be forwarded to. The protocol design should take into consideration the energy spent for each operation.

- It is also desirable to have a cost metric attached to the header of the packet which can measure the cost of a packet from a source to a destination. This metric should take into account the resource cost for the packet to go from the source to the destination, where cost should include the energy spent, the network resources used and other factors like these.

Bibliography

[1] `http://www.netlab.tkk.fi/tutkimus/dtn/theone/`.

[2] C Bettstetter. Mobility modeling in wireless networks: categorization, smooth movement, and border effects. *SIGMOBILE Mob. Comput. Commun. Rev.*, 5(3):55–66, July 2001.

[3] E. Bulut. *OPPORTUNISTIC ROUTING ALGORITHMS IN DELAY TOLERANT NETWORKS*. PhD thesis, Rensselaer Polytechnic Institute, 2011.

[4] J. Burgess, B. Gallagher, and D. Jensen. Maxprop: Routing for vehicle-based disruption-tolerant networks. In *Proceedings of IEEE Infocom, April 2006*, 2006.

[5] Douglas S. J. De Couto, Daniel Aguayo, John Bicket, and Robert Morris. a high-throughput path metric for multi-hop wireless routing. *Wireless Networks*, 11:419–434, 2005. 10.1007/s11276-005-1766-z.

[6] A. Dvir and A V. Vasilakos. Backpressure-based routing protocol for dtns. In *Proceedings of the ACM SIGCOMM 2010 conference*, SIGCOMM '10, pages 405–406, New York, NY, USA, 2010. ACM.

[7] K. Fall. A delay-tolerant network architecture for challenged internets. In *Proceedings of the SIGCOMM '03 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 27–34, 2003.

[8] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis. Collection tree protocol. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, SenSys '09, pages 1–14, New York, NY, USA, 2009. ACM.

[9] B. Jabbari, Y. Zhou, and F. Hillier. Random walk modeling of mobility in wireless networks. In *48th IEEE Vehicular Technology Conference, 1998. VTC 98.*, pages 639 – 643 Vol.1, 1998.

[10] P. Jacquet, P. Muhlethaler, T. Clausen, A. Laouiti, A. Qayyum, and L. Viennot. Optimized link state routing protocol for ad hoc networks. In *Proceedings of IEEE INMIC 2001*, pages 62–68, 2001.

[11] D.B Johnson and D.A Maltz. Dynamic source routing in ad hoc wireless networks. In *Mobile Computing*, pages 153–181, 1996.

[12] A. Keränen, J. Ott, and T. Kärkkäinen. The one simulator for dtn protocol evaluation. In *Proceedings of the 2nd International Conference on Simulation Tools and Techniques*, Simutools '09, pages 55:1–55:10, ICST, Brussels, Belgium, Belgium, 2009. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[13] Q. Li, S. Zhu, and G. Cao. Routing in socially selfish delay tolerant networks. In *Proceedings of IEEE INFOCOM 2010*, pages 1–9, 2010.

[14] A. Lindgren, A. Doria, and O. Scheln. Probabilistic routing in intermittently connected networks. In *ACM SIGMOBILE Mobile Computing and Communications Review*, pages 19–20, 2003.

[15] S. Moeller, A. Sridharan, B. Krishnamachari, and O. Gnawali. Routing without routes: the backpressure collection protocol. In *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, IPSN '10, pages 279–290, New York, NY, USA, 2010. ACM.

[16] C.E. Perkins and E.M. Royer. Ad-hoc on-demand distance vector routing. In *Proceedings of Second IEEE Workshop on Mobile Computing Systems and Applications*, pages 90–100, 1999.

[17] J.M. Pujol, A.L. Toledo, and P. Rodrigfuez. Fair routing in delay tolerant networks. In *Proceedings IEEE INFOCOM 2009*, pages 837–845, 2009.

[18] J. Ryu, L. Ying, and S. Shakkottai. Back-pressure routing for intermittently connected networks. In *Proceedings of IEEE INFOCOM 2010*, 2010.

[19] T. Spyropoulos, K. Psounis, and C Raghavendra. Spray and wait: an efficient routing scheme for intermittently connected mobile networks. In *Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, pages 252–259, 2005.

[20] T. Spyropoulos, K. Psounis, and C. S. Raghavendra. Spray and focus: Efficient mobility-assisted routing for heterogeneous and correlated mobility. In *Fifth Annual IEEE International Conference on Pervasive Computing and Communications Workshops, 2007*, pages 79 – 85, 2007.

[21] A. Vahdat and D. Becker. Epidemic routing for partially-connected ad hoc networks. Technical report, Duke Tech Report CS-2000-06, 2000.

[22] A. Vasilakos, Y. Zhang, and T. Spyropolous. *Delay Tolerant Networks: Protocols and Applications*. 2009.

[23] L. Ying, S. Shakkottai, A. Reddy, and S. Liu. On combining shortest-path and back-pressure routing over multihop wireless networks. *IEEE/ACM Trans. Netw.*, 19(3):841–854, June 2011.

[24] M. Zuniga and B. Krishnamachari. Analyzing the transitional region in low power wireless links. In *IEEE SECON 2004*, pages 517–526, 2004.

Vita

Priyanka Rotti was born in Hubli, India in1987 to Gururaj Rotti and Vidya Rotti. She obtained her Bachelor of Engineering in Information Science and Engineering from Visvesvaraya Technological University, Belgaum, India in 2008. She worked as a Technical Support Engineer at Microsoft India Global Technical Support Center at Bangalore, India. In Fall 2010, she came to the US in pursuit of a Masters degree. She completed her Masters in System Science at Louisiana State University, Baton Rouge in December 2012.