An Effective Infrastructure for Pocket Switch Network

SUBMISSION DATE: 14.12.17

SUBMITTED BY:

Mohammad Kaif Ul Majed (14101066)

Anika Mariam (14101044)

Afia Noshin (14101076)

Md. Jamiul Alam Khan (14101073)

Department of Computer Science and Engineering

Supervisor:

Amitabha Chakrabarty, Ph.D

Assistant Professor

Department of Computer Science and Engineering

Declaration

We, hereby declare that this thesis is based on results we have found ourselves. Materials of work from researchers conducted by others are mentioned in references.

Signature of Supervisor

Amitabha Chakrabarty, Ph.D
Assistant Professor
Department of Computer Science and Engineering
BRAC University

Signature of Authors

Mohammad Kaif Ul Majed (14101066)

Anika Mariam (14101044)

Afia Noshin (14101076)

Md. Jamiul Alam Khan (14101073)

ABSTRACT

In everyday life we are using Internet with fixed routing configurations such as Broadband, mobile data etc. Pocket Switch Networks (PSN) uses phones as the medium to connect with other people without a fixed cellular network or Internet. Since no fixed route to destination is available, problems arise when a node wants to send any message to a destination node. Packets are forwarded by flooding. Flooding causes increase in network congestion, resource consumption, delay and drop of packets. This approach can be applied to smaller networks but in large scale inefficient. To deliver messages in a larger network with high throughput and low latency we propose an infrastructure for PSN called ZoneCluster. First, we used probabilistic approach to eliminate flooding by forming clusters and having inter-cluster and intra-cluster routing. Secondly, we have saved the path of delivered messages from source to destination so that they can use the same path for faster communication. Lastly, we emphasize its application for larger networks such as an AD-Hoc WAN by comparing its performance with other protocols.

Keyword: pocket switch network, zone routing, clustering, ad hoc network.

Acknowledgement

We would like to thank Almighty Allah for the opportunity and the capability that we could achieve a vast amount of knowledge and by His blessings we would achieve our goal.

We also like to thank our supervisor Dr. Amitabha Chakrabarty, for his immense support and guidance we could finish our research properly. He was always there for us and we learned a lot from him that helped us to reach our goal smoothly.

We thank our parents for always believing in us and support us throughout to reach our goal for this research.

Finally, we thank our faculty members from whom we gather a lot of knowledge and that helped us to think deeply to accomplish our research.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

Introduction

Pocket switched networks (PSN) [1] is a type of delay tolerant network (DTN) where the data is transferred from one node to another by use of radio technology like Wi-Fi or Bluetooth. The messages need to be forwarded in a network where the nodes are intermittently connected. At any moment of time, two nodes in communication with one another may not have a connected path between them. As a result, the intermediate nodes need to apply a store and forward approach. This can cause the messages to be delivered with huge latency.

Furthermore, as the nodes are mobile devices they have a storage limitation known as buffer. If a node were to store a message for a long time, it will occupy buffer space and as a result, it may be unable to accept new messages from other nodes. Therefore, it needs to have a mechanism for freeing up the buffer space when it is unable to forward a message for a long time. Messages have a (Time-to-live) TTL field that causes them to expire or the node itself can implement a type of congestion management system.

To forward messages continuously between two nodes, the nodes themselves must store route information for communication with one another. Otherwise, the nodes will have to search the network every time for one another when sending a message leading to increased overhead in communication. Saving a route also poses problems. Due to the mobility of hosts leading to a rapid change in topology, any saved route can soon become broken.

Another issue with such networks is the unwillingness of the hosts to forward messages for other hosts. Since forwarding of data packets consumes resources of the host, such as power and memory some hosts can be selfish and carry their own messages. Such a network requires every host to participate in the carrying and forwarding of messages so the presence of uncooperative nodes can hinder the performance.

Finally, since hosts carry messages, there is a security concern raised. Malicious hosts can carry messages for others and try to alter them or view them. Therefore, measures to encrypt the data to prevent this are required.

Hence, we have tried to present an infrastructure for routing in such an intermittent network scenario after providing the existing literature.

1.1 Motivation

Technological advancement has led to those achievements that were considered impossible even a few decades ago. DTN is one of those fields which is centered on intermittently connected networks. Today a large number of people uses a smart portable networking device to communicate either using mobile data or Internet. However, what if this communication could be done without using mobile data or internet connection? Surely communication could be cheaper but reliant on the number of users willing to participate in such a network. Keeping this notion in mind many routing protocols have been established to forward this idea. This is because communication with a fixed route that we generally get through the Internet is unavailable here. Facing it is a great challenge. PSN (Pocket Switch Network) works on this kind of network where it is aimed for a good communication between mobile devices without any kind of disruption. It comes of great advantage for communication where fixed infrastructure is not available, like after natural disasters, in rural areas, military use on battlefields, vehicular networks etc.

In our thesis, we propose a routing algorithm that can work on larger ad hoc networks and have a higher delivery rate, eliminate flooding as well as relatively low latency.

1.2 Thesis Contribution

This thesis contributes a new infrastructure for large ad hoc networks spanning hundreds of kilometers with hundreds of thousands of hosts. There are so many routing protocols that are available but most of them do not work efficiently on large networks. We have combined several algorithms together to determine the forwarding policy of the nodes. We have then compared our algorithm's performance with others to measure its suitability in such a network.

1.2.1 Problem Statement

One of the major problems of pocket switched networks is the constant changing of path of connection between nodes. So routing is not always fixed for data exchange. Moreover, every node has to store the routing information in its routing table otherwise delay occurs when two nodes continuously want to exchange messages. Furthermore, because of lack of nodes, the network may become partitioned leaving messages unable to reach their destination.

1.2.2 Solution

One of the ways to solve the problem of trying to establish a path between two hosts for continuous exchange of messages can be to extract the hop sequence of the message on the receiving host. Then the receiver can save the route in its routing table. The receiving host can then use this route to send subsequent messages to the sending host. However, as in this type of network the nodes are mobile, a stored path can soon be invalidated. Nodes then need to react to these changes by finding a new path for the packet instead of dropping it entirely. As mentioned in some works like [2], an attempt is made to salvage the packet. The forwarding node checks its own routing table to see if it has another route to the destination. Also it sends a gratuitous reply to the source of the message telling it that the route is no longer available and the host can then drop this route from its routing table to prevent using this route further for new packets.

1.2.3 Methodology

In our algorithm, we have kept the problems as mentioned above in mind and tried to find a solution. First, we have taken the clustering approach as mentioned in [3] to classify the nodes in our network into three distinct roles; the clusterhead, gateway and ordinary. Secondly, we have used Dynamic

Source Routing [2] to discover routes from the source to the destination for which it does not have any path. The route requesting is handled by the clusterheads and gateways thus eliminating the flooding done in the network. Furthermore, we provide our own mechanism for route maintenance that can correct breakages in routes instead of dropping the route or packet entirely. In this way, whenever a route fails, the need to reinitiate route request from the source to the destination is no longer required, it is done instead by the node that detected the breakage. Doing so results in more reactivity of the network to topology changes and effectively reduces the latency and the number of route request messages being propagated in the network.

1.3 Goal

In the literature, we will see that all the algorithms have been used in small networks with movement models like the SASSY dataset [4]. In our algorithm we are thinking on a larger scale and want to implement an AD-HOC WAN network with real-life scenario movement of people. Such a network can involve thousands of users and thousands of messages generated at any time. Therefore, we need to ensure high delivery rate and low latency to use such an algorithm instead of the traditional Internet or mobile networks.

1.4 Thesis Overview

This thesis paper is divided into five chapters. The outline of each chapter is given below.

Chapter 1 contains an introduction to PSN and its features. The challenges of developing an algorithm for such networks are discussed. We propose how to solve these problems and what we hope to achieve with our algorithm.

Chapter 2 presents the relevant works we have studied to come up with our own algorithm.

Chapter 3 describes the steps of our implementation and our proposed algorithm.

Chapter 4 discusses results and comparison of our algorithm with others.

Chapter 5 is the conclusion and future work. Here we summarize our idea and provide the scope for improvement in the future.

CHAPTER 2

Literature Review

In traditional way, for message delivery there is always a fixed router and has a fixed route as well. The source and the destination are not movable. If one wants to send a message it is quite easy to delivery it after examining the destination and send it to the routers. We already know the TCP/IP protocols. On that they use internet to deliver messages but they have some limitations. They only work on the fixed network. TCP/IP cannot transfer messages when there is a loss of connectivity. But in Ad hoc networks messages can be delivered without any fixed path.

In Delay Tolerant Network (DTN) is a type of ad hoc network in which message delivery is possible without any fixed connectivity. It [5] can transfer messages where is insufficient end to end connections between source and destinations. They have intermitted connections between then so they follow ‘store and forward’ approach to move data [6].

There is another type of ad hoc network which is called Mobile ad hoc network (MANET) [7]. It works with the radio technology such as mobile devices and mobility of users to establish connectivity and transfer data [8].

They form a temporary network without any fixed infrastructure as they are very flexible [9].

The subclass of Delay tolerant network (DTN) is Pocket Switch Network (PSN). This works with mobile phone that we keep in our pockets. It can work on intermitted communication that has no stable infrastructure. It follows the mobility module of human society. We have read various algorithms that are based on PSN. We listed some of them below.

There are various algorithms to decide how a message should be forwarded from one node to the next. For example, in [10] [11] [12] [13]. The main object of such algorithms is to:

1. Limit the number of times a message is copied by nodes while forwarding.

2. Limit the number of hops traversed by a message and as such reduce the latency.

3. Increase the delivery probability.

4. Drop few messages.

We mentioned here a number of algorithms that implemented Pocket Switch Network (PSN). We provide a brief description of all the algorithms that we read and compare them with ours.

2.1 Cluster based routing protocol

A group of nodes is defined as a cluster [3]. Every cluster has a cluster head, gateway and some cluster members. A cluster head is in charge of routing within the cluster and to other clusters. It is selected within a cluster by their ID [14]. ID generates MAC address or IP address. At the time of election those nodes which have the lowest ID is defined as the Cluster head. Cluster head has all the information within its cluster. They exchange hello messages to know about their neighbors. More than one cluster head cannot be present in one cluster. If one cluster head receives hello message from another cluster head they will wait for a certain amount of time. If they are still in contact, the one with the lowest ID remains a cluster head and the other becomes a cluster member. They follow the two hops away approach. A cluster head knows the other cluster head that is two hops away by hello messages.

In this protocol only cluster head and gateway can send messages. Cluster members can only transfer messages to the cluster head. Gateway node has a bi-directional link to the other clusters. Gateway becomes the connection between two clusters. When a node tries to send a packet at first it sends a route request to the destination. It saves the path and the destination

sends a route reply packet to the source. Initially route request packet broadcast within the network. Only cluster head broadcast the route request packet. The route request packet is not send from where it comes. The route reply packet copies the cluster addresses. This information helps destination to reach the source. In this protocol route shortening is possible by searching the furthest node of the path. If it finds that, it will shorten the path.

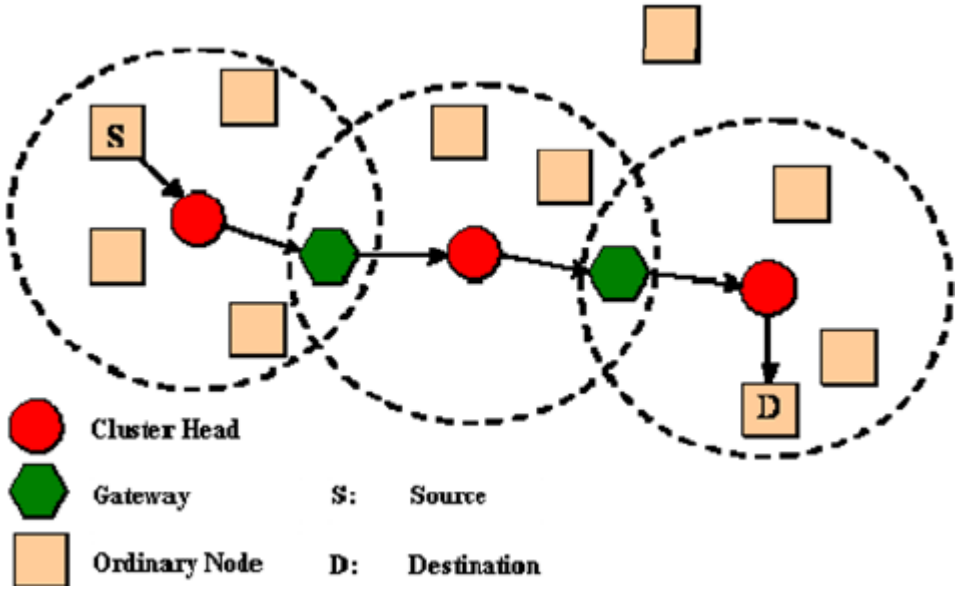

Figure 2.1 Message transfer by clustering protocol

2.2 Zone routing protocol

Zone Routing Protocol (ZRP) is kind of a mixture of proactive and reactive routing protocols [15]. Proactive routing is similar to the wired IP networks. It saves the up-to-date information of all nodes which are

consistent. On the other hand, reactive routing the routes are not set. When a source wants to send a message to the destination, it discovers the route and to the destination [16].

ZRP does not work with hierarchical protocols it works with the flat protocols. It is easier to get a good overview and it also reduces the congestion of the network. In ZRP each node has a separate zone and the zones overlapped with each other. The protocol defines zones as hop count. They have two types of nodes like peripheral nodes and interior nodes. Peripheral nodes are the border nodes which stays at the border of the zone and interior nodes stays inside the zone not on the border.

For Example, Node I is source and node V is destination. The destination is not present in node I's zone. So I send the packet to all the border nodes like T, R, Q, F, S and D which follows 2-hop approach. After connection establishment source forward data packets.

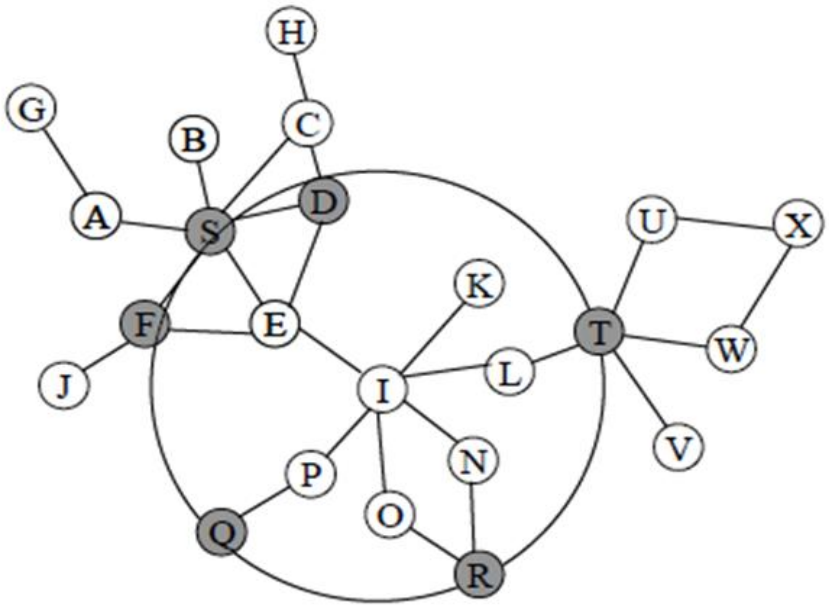

Figure 2.2 Zone of Node I

In ZRP when a node wants to send a packet to the destination at first it checks whether or not the destination is in the local zone or not. If it is then it uses proactive routing to send the packet. There can be another scenario that destination is not in the local zone there they use reactive routing. It has two phases such as route request phase and route reply phase. For sending packets outside its own zone at first source sends a route request packet to the peripheral nodes and if the nodes find the destination it sends a route reply packet to the source. The route reply packet includes the path to the destination. The route request packet will broadcast the packet unless it finds

the destination. In that case, destination can get same packets several times. Those packets will be treated as redundant packets. The route reply packets are saving the path. Source knows the whole path to the destination and the next-hop address is saved in the path nodes. In this type of networks paths can be broken so route maintenance is necessary. If one node is missing they search for the next hop and choose the shorter path to the destination.

2.3 Epidemic

Epidemic routing protocol forwards messages whenever it comes in contact with other node and that node does not carry the same message [17]. All hosts preserve a buffer of their own messages as well as the messages of other hosts. Every message contains a unique identifier. Each host has a cache where they store the information of previous host that they encountered in recent times for avoiding redundant connections. Every host also contains a summary vector where they accumulate information about messages they already have. So when two hosts meet they exchange summary vector and request for new messages that individuals do not contain. The receiver can decide whether or not it will take the message or not. This algorithm tries to maximize delivery of messages. It follows flooding approach so it uses higher buffer size and transmission power [11].

```
Procedure Epidemic:

ForEach node s, do
        ForEach message  k, do
                ForEach encountered node d, do
                        sendMessage (k, d)
                EndForEncounter

        EndForMessage
EndForNode
EndProcedure
```

Figure 2.3 Pseudocode of Epidemic routing protocol

2.4 First Contact Routing Protocol

First contact routing protocol [18] is a single copy routing protocol which forwards messages to only the node it encounters first. The source forwards the copy to the first node it meets and delete the message instantly. Again the recipient node transfers to the next node it meets and then remove it. This goes on until the destination is met. While forwarding it does not consider the next best node who have higher probability of delivering the copy. If in case the recipient node fails to transfer the message, then the message is lost. Hence the delivery ratio is low.

```
Procedure FirstContact:

ForEach Node s, do
        ForEach message  k, do
                For encountered node i,do
                        sendMessage(k,d)

                EndFor
        EndForEach
EndForNode
EndProcedure
```

Figure 2.4 Pseudocode of First Contact Routing

2.5 Direct Delivery Routing Protocol

This routing protocol removes the flooding concept and follows a very simple approach [13]. The source stores the message until it finds a destination. So it directly delivers the message to the destination. In this approach the source needs to have a direct connectivity to the destination without concerning the network topology.

```
Procedure DirectDelivery:

ForEach Node s,do
        getAllMessages()
        ForEach message  k, do
                ForEach encountered node d, do
                        If k.destination = d
                                then sendMessage(k,d)

                         EndIf
                EndForEncounter
        EndforMessage
EndForNode
EndProcedure
```

Figure 2.5 Pseudocode of Direct Delivery Routing Protocol

2.6 ChitChat

ChitChat [19] is one of those message delivery methods where routing decisions are made based on the higher probability that the message receiver has of relaying it to the next node closer to the destination or that have high chance of forwarding it to the destination. This method approaches to make message delivery possible in such situations where there is no fixed connection between dynamic nodes. These situations refer to the network of PSN (Pocket Switch Network) as we described before. Since there is no stable infrastructure between the carrier nodes in our pockets frequent disconnection is a common scenario. Moreover, to reach the destination network congestion

might occur due to lesser contact frequency, contact duration and higher momentum of information in a single message. So to avoid these problems this method has brought forth an approach to optimal solution where routing is done only through that connection that has higher feasibility of reaching destination without creating network congestion and delay. This has focused on sparse network where nodes have distance between them and not congested. Here, each carrier nodes have defined profile where they have social interest with unique keyword for it. This is the medium of determining if it has higher chance of forwarding it towards the destination.

Whenever two nodes within range connect with each other they exchange two kinds of information before forwarding the message. They are direct social interest and transient social relationships. Direct Social Interest refers to the social interest of the respective node that the other node has met. Every node has few social interests which have unique key ID to it for identification. These interests are mainly hobbies or likings. Transient social relationships are the social interests of other nodes that the encountered node has interacted with. Basically the main idea is to increase the horizon of meeting more nodes that can lead to the destination. It focuses on taking calculative routing decision even though the distance between the source to

destination might be longer. Here messages are forwarded by keeping the copy of message to itself when until it reaches the destination or TTL is expired or buffer is congested with messages that has higher probability of being delivered.

As mentioned before about the TSR (Transient Social Relationships), it is represented for user *u* in timestamp *t* as *TSRu (SID1, w (SID1, t))* where *SID1* is the social interest and *w(SID1, t)* is the weight of social interest at time *t.* Initially the value of weight is kept at 0.5 later with more encounters it is increased. The equation (1) of transient social relationship [19] is given below.

$$Wu(SIDi, ts) = \begin{cases} \dfrac{wu(\mathrm{SID}i, td, i)}{\beta \cdot (ts - td, i)} \\ \quad If\ SIDi \notin SPu \\ \dfrac{(wu(\mathrm{SID}i, td, i) - 0.5)}{\beta \cdot (ts - td, i)} + 0.5 \\ \quad if\ SIDi \in SPu \end{cases} \tag{1}$$

Whenever two users are in a communication range, using this decay function the ChitChat calculates the weight of user's social interest in their respective TSR first. Then they are exchanged.

All experiments of this method was done by One simulator [13]. And it was tested against several algorithms like Epidemic [11], Sane [20] and Sedum [21]. It has shown some results regarding having more hop counts than Sane [20] and Sedum [21] but has indicated that lower hop count does not ensure efficiency but portrays capability to deliver message to neighboring nodes while failing to deliver most of the created messages. In addition to that ChitChat [19] has shorter buffer as it makes intelligent decisions in forwarding Messages other than flooding and filling up the buffer unnecessarily without creating network congestion. The following pseudocode [19] represents the Real-time transient social relationship modeling.

```
Procedure CONNECT(u, vtc )
For each SIDi ∈ SPu do
        If td,i == tc then
                wu(SIDi, tc) = wu(SIDi, td,i)
        ElseIf SIDi ∈ SPu then
                wu(SIDi, tc) = wu(SIDi, td,i) − 0.5/β · (tc − td,i) + 0.5
        Else
                wu(SIDi, tc) = wu(SIDi, td,i)/β · (tc − td,i)
        EndIf
 EndFor
// Exchange current TSRs with new neighbors.
 ForEach node v ∈ vtc do
        Send TSRu to v
        Receive TSRv from v
        ts,v = tc
 EndFor
ForEach SIDi ∈ all received and cached TSRs do
        Δ =v∈v  wv(SIDi, ts,v) · (tc − ts,v) /ψi,u,v
       wu(SIDi, tc + 1) = min{1, wu(SIDi, tc)+Δ}
 EndFor
EndProcedure
```

Figure 2.6 Pseudocode of ChitChat

2.7 Sedum

While many Algorithms are based on flooding and single copy routing there are some which focuses on minimum number of copies of messages which is forwarded by intelligent routing decision based on some definite

properties. Sedum [21] is one of those algorithms. It focuses on contact frequency and contact duration between two nodes. By contact frequency it is meant the nodes having closer contact in a fixed range but for shorter period of time. And by contact duration we understand that nodes having higher probability of meeting and being in contact for a longer period. They have approached for a protocol that exhibits both the features in it. This proposed approach has three main features which are Duration utility-based distributed routing, efficient multicopy routing and Effective buffer management.

The duration utility refers to the ratio of duration of contact between two nodes over time period T. The higher the value is more is the throughput of the message transmission. The equation of Duration Utility U between two nodes i and j considers the maximum value of Direct Utility, $\widehat{U}_{(i,j)}$ and Indirect Utility, $\widetilde{U}_{(i,j)}$ between two nodes. The equation (2) of the Duration Utility [21] is given below:

$$U_{(i,j)=max\left(\widehat{U}_{(i,j)},\ \underset{k\in N}{max}\left(\widetilde{U}_{(i,j)}\right)\right)} \quad (2)$$

Again nodes take into account their historical and current utility, according to the formula (3), update Duration Utility [21] periodically.

$$U_{(i,j)_{new}} = \gamma U_{(i,j)} + (1-\gamma) U_{(i,j)_{old}}, \gamma \in (0,1) \quad (3)$$

Here, γ represents weight constant ranging from 0 to 1, $U_{(i,j)_{new}}$ denotes the utility of the new time interval and $U_{(i,j)_{old}}$ denotes the utility of old time interval.

Multicopy routing focuses on minimal number of copy of messages from source to destination. Buffer management gives priority the message with higher utility to stay if it is congested and the longer staying messages in it. Moreover, it deletes the replica of delivered message very quickly. Higher throughput is possible considering both contact frequency and contact duration in Sedum has been proved by simulation than the throughput achieved by considering only contact frequency. There is table for each node where they keep record of the duration utilities with other nodes it contacted. The node forwards message to that node which has higher duration utility. The method focuses on node movement pattern based on social network. The following pseudocode [21] represents the duration utility.

```
If meet node n_j then
    Exchange utilities that has been updated since last
    Time meet with n_j
    If U_(i,j) exists in n_i's utility table then
        ForEach node n_k in updated utilities do
            If U_(i,j)*U_(j,k)> U_(i,k) then
                Update U_(i,k) using Formula (2)
    Record the contacting time with n_j
If currentTime= updateTime then
    updateTime+=T
    ForEach meeting node n_j in the last time period T do
        Calculate U_(i,j) using Formula (2)
            If U_(i,j)_old exists in its utility table then
                Update U_(i,j) using Formula (3)
```

Figure 2.7 Pseuducode of Duration Utility Calculation

2.8 PRoPHET Routing Protocol

In Delay Tolerant Network (DTN) nodes forward message to the next node without knowing the direction and so they use various methods of fining the best path towards destination. They might keep record of previous encounters of each node for packet forwarding or flooding where they forward message to every node it encounters. From [10] we know that this new

approach is the improved version of the general PRoPHET routing protocol which uses additional feature of considering the distance metric.

The previous version used two factors when forwarding message in a wireless network. They are: Delivery Predictability and Transitivity. Here Delivery Predictability is set for every node X for its destination Y. When it is calculated a node with higher DP (Delivery Predictability) is always considered to be the best route to forward the packet that is, it is used for forwarding decision. When two nodes encounter with each other they exchange their DP through hello message to update it and bundle information including the destination and the total size of it along with the updated DP. More nodes a node encounter the value of DP increases. Again the value may decrease if it does not encounter with any nodes for a long period.

Transitivity refers to the degree of association with other nodes of the encountered nodes to forward a packet. That is if a node A frequently meets node B and node B frequently meets C then node C is the best to forward message for node A. The equation (4) of the process [10] is mentioned below.

$$P_{(a,c)} = P_{(a,c)_{old}} + (1 - P_{(a,c)_{old}}) \times P_{(a,b)} \times P_{(b,c)} \times \beta \qquad (4)$$

Here $P_{(x,y)}$ represents the delivery predictability of x to y and β represents the scaling constant that weighs the impact of transitivity on delivery predictability.

The value of DP is never fixed and it fluctuates with every encounter it has. Moreover, since it does not consider the distance between the nodes so DP can be same for two more nodes even though distance might vary. In this situation, the node forwards message to both and thus causes resource consumption and delay. Therefore, it causes a great dilemma.

2.9 Other algorithms

Sane [20] is another routing algorithm that focuses on the similar interests of nodes in a network that deduces the mobility pattern of each for routing decisions. Each individual in a network has a compact set of interests along with their unique id within their interest space. Here, when a node suppose X wants to send a message to Y, interest profile is attached to the message. So before any carrier node receives it via forwarding, at first the interests are compared between the carrier node and destination and depending on the result, forwarding decisions are made. It has suggested an interest-casting where a message is portrayed as message relevance profile.

This message is designed to reach a number of users of similar interests or almost similar interests.

Maxprop routing protocol is designed for vehicle based delay tolerant network [22]. It increases delivery rate and decreases latency rate. They deliver the messages by probability that could reach the destination. It is one kind of probabilistic routing.

Another routing protocol named Spray and Wait was proposed in [12]. Here the source node sends a definite number of copies to other nodes in the networks and wait for the corresponding recipient nodes to pass it over the destination. Since there is no definite path between two nodes, after source sends to a number of nodes, these nodes may take directly to destination if there exists a link or wait until another link comes up towards the destination. Forwarding decisions are made based on connectivity information. It uses flooding but not as [11] or other routing protocols where copies of message is sent to all the nodes available. Here source sends L number of copies to other nodes. This is the Spray Phase. After copies of messages being sent, the receiving nodes see if they have any link towards the destination. If they don't find any link they wait for a link to come up. The recipients from the source only send to destination if available and if not then they wait.

Some algorithms such as [23] considers the battery life, as they load balance the forwarding of messages between nodes to ensure that one node does not transfer the majority of messages for a long time. Another factor found is the processing power of a node. The number of messages that a host can handle can put a bottleneck on the performance of the network. Many routing algorithms require hosts to keep information about the network which they use to decide how to route different messages to different hosts. As a result, it can add significant overhead to the network and require more computational power from the hosts.

CHAPTER 3

Implementation

3.1 Election Process

As we have mentioned before, we have used the clustering approach to define the behavior of nodes in our network. The first phase involves the selection of clusterheads.

```
Procedure Election:
// Election phase
ForEach node s, do
        s.nodeType=clusterhead
        s.nodeCount=0
        ForEach encountered node d, do
                nodeCount++;
        EndForEach
        ForEach encountered node d, do
                If s.nodeCount < d.nodeCount then
                        s.nodeType=ordinary
                        break
                EndIf
        EndForEncounter
        If s.nodeType=ordinary then
                s.clusterCount=0
                ForEach encountered node d, do
                        If d.nodeType=clusterhead then
                                clusterCount++
                        EndIf
                EndForEncounter

                If clusterCount=1 then s.nodeType=ordinary
                Else s.nodeType=gateway
        EndIf
        If s.nodeType=clusterhead then
                ForEach encountered node d, do
                        If d.nodeType=gateway then
                                gatewayCount++
                        EndIf
                EndForEncounter
        EndIf
        //End of Election
EndForNode
EndProcedure
```

Figure 3.1 Pseudocode of election process

3.1.1 Clusterhead Election

Clusterheads are like the default gateways in wired networks. They control the routing of packets to nodes outside as well as inside the cluster. At the initial phase, every node scans their transmission range to find the nodes present. It then saves the number of nodes it encountered. After every node has determined the number of neighbors it has, we can now start the clusterhead election. Nodes initially believe themselves to be clusterheads. Once they start comparing their node counts with other nodes in their transmission range, the nodes with less node count then their neighbors automatically become ordinary. The equation (1) below is used to save the number of nodes present in the range.

$$nodeCount_i = \sum_{j \in C(i,j)} 1 \qquad (1)$$

Here, C(i,j) denotes the connection between node i and its neighbor j.

In cases where two nodes have the same node count, the tie can be broken using lowest-ID approach [14]. Here each node is given a unique ID in the network, the ID is an alphanumeric one. The node with the lowest ID wins and the other node becomes ordinary. After the nodes have completed

this phase we are left with clusterheads and ordinary nodes. Now we enter the second phase which is the determination of gateways.

3.1.2 Gateway Election

Every ordinary node can now scan their transmission range to find others nodes that are either ordinary or clusterhead. Using the equation (2), It keeps a count of the number of clusterheads it encountered, if this count is greater than 1 then it becomes a gateway node or it remains ordinary. This concludes the election process.

$$clusterCount_{i_{ordinary}} = \sum_{j_{cH} \in C(i,j)} 1 \qquad (2)$$

3.1.3 Re-election

The entire clustering process will need to happen again under two conditions.

1. If an ordinary or gateway node does not have a clusterhead in transmission range. The equation (3) below is used in determining if there is no clusterhead in the range.

$$clusterCount_{i=(ordinary||gateway)} = 0$$
$$\sum_{J_{cH} \in C(i,j)} 1 = 0 \quad (3)$$

2. If two clusterheads come into transmission range of one another. The equation (4) is used to find if more than one clusterhead comes into contact.

$$C(i,j) \in (i = cH \ \in \& \ j = cH) \quad (4)$$

Here, C(i,j) denotes the connection between node i and its neighbor j.

Ordinary and gateway nodes only need to keep count of the number of clusterheads they are in contact with. If an ordinary node now has more than one clusterhead, it becomes a gateway and a gateway becomes an ordinary node if it has only one clusterhead in range. If neither of these nodes detect any clusterhead in range, they start the election process again and may end up becoming clusterheads themselves. The algorithm suffers from inconsistency owing to the fact that the network may become partitioned. This will create disjoint sets of clusters who have no connection to one another. So gateway nodes may be absent in such cases. The clusterhead is then unable to forward

the messages to any clusterhead and so, must wait till it finds a gateway due to the movement of nodes which can remove partitions created in the network. If the cluster happens to be isolated from the rest of the network. This completes the election phase of our algorithm.

3.2 Packet forwarding

Now that we have established the type of every node, we can begin the next part of our algorithm which is packet forwarding. Every node has a data structure where it stores the sequence of hops to traverse for a particular host. We keep in mind that unlike Destination-sequence Distance Vector (DSDV) routing protocol [8], we do not keep a route to every reachable destination but rather to destinations we have communicated with recently. Hence there need not be any routing table exchanges between nodes so they can run distance vector algorithms to find the shortest paths to their neighboring networks. Only routes that have been discovered using the Dynamic Source Routing (DSR) protocol [24] are saved by each host.

After creating a message, a host checks its route table to see whether it contains a path to the destination. Initially no node in the network has any routes saved. Therefore, whenever a node wishes to send a message to any

destination, it must send the message using clustering approach. The mechanism of the clustering approach will now be described.

3.2.1 Clustering process

Every ordinary node that creates a message will forward it to their clusterhead. The clusterhead then checks the nodes in its transmission range to find the destination node and if found, forwards to that node. In this way, nodes within the same cluster only need 2 hops to reach one another. If the destination node does not reside in the same cluster, the clusterhead must forward the message to one of its gateways. Instead of flooding the message to all the available gateways, each gateway maintains a count of the number of clusterheads that is in its transmission range. The clusterhead can use this count to compare all the gateways and select the one with the largest count. If a tie exists, the clusterhead can select any one to them to forward to. The following equation (5) is used to decide the gateway, to whom to be forwarded.

$$gateway = C(i,j) \in (i = cH \;\&\; j = gateway) \max(clusterCount_j) \qquad (5)$$

Here we have scope to apply some load balancing approaches to ensure that the gateways are properly utilized in situations where they are all eligible and good candidates to forward the message. The gateways then forward the message to an eligible clusterhead. Every clusterhead maintains a count of the number of nodes in its cluster and amongst them how many are gateways. The clusterhead with the most number of nodes and gateways has a higher probability of delivering the message to the destination. So the gateway forwards the message to this clusterhead. A tie may exist in that case the same approach can be applied as with the clusterhead forwarding to a gateway scenario. The following equation (6) is used to find the clusterhead with highest node and gateway count.

$$cH = C(i,j) \in$$
$$(i = gateway \;\&\; j = cH) \max(nodeCount_j + gateCount_j) \qquad (6)$$

In this manner a sequence of clusterheads and gateways are used to propagate the message throughout the network in search of the destination. This process can be avoided however if any node along the way already possesses in its

route table an entry for the destination. Then it can add the path to the message and it can be forwarded by the hosts accordingly. How nodes save routes to other hosts will be discussed shortly.

```
// Message Forwarding by clustering
Procedure Clustering :
    ForEach message k, do
        If nodeType=clusterhead then
            // Direct delivery of messages
            ForEach encountered node d, do
                If k.destination = d
                    then sendMessage(k,d)
                EndIf
            EndForEncounter
            ForEach encountered node d, do
                If d.nodeType=gateway
                    then gateways.add(d)

                EndIf
            EndForEncounter
            Node d = mostPopularIn(gateways)
            sendMessage(k,d)
        EndIf
        Else
            ForEach encountered node d, do
                If d.nodeType=clusterhead
                    then clusterheads.add(d)

                EndIf
            EndForEncounter
            Node d = mostPopularIn(gateways)
            sendMessage(k,d)
        EndElse
    EndForMessage
EndProcedure
```

Figure 3.2 Pseudocode of clustering process

3.2.2 Forwarding with saved route

After the destination has received a message from the source for the first time, it must send a response message back to the source. The destination also saves the list of hops that was traversed by the message originated from the source in its route table. Whenever this node wants to forward messages to the source it can use this route. The response message can also be piggybacked with data that the destination wants to send to the source. The route is attached with the message is now forwarded according to the sequence of hops specified. Once the message reaches the source, the source then saves the path travelled by the message in its route table. In this way, after sending only two messages, two nodes can know of a route to one another and can use this for further communication.

However, the likelihood of being able to use this route for a long period of time in an Ad-Hoc network is not realistic. We propose a way of correcting any breakages in the path using an algorithm we designed called “local correction.” The details of this is discussed later. After invoking local correction and successfully repairing the break in the path, the message will arrive at its destination carrying a different sequence of hops than the one saved in the host’s route table. The host can then update its route table entry

with the new route from the latest message. This ensures that nodes communicating consistently for a prolonged period of time can maintain a fresh route and send messages to one another with little delay and reduces the need to constantly find routes to one another.

Intermediate nodes, that forward a message that was a known path from its source and destination can also save this information in their own route tables. For example, if node A and F are in communication with one another and have a route established between them, every message they send contains the route they need to follow. If a message M, sent by A to F has the following sequence of nodes in its route entry, A-B-C-D-E-F, then the intermediate nodes, B, C and D can also save the route in their route tables. For B, an entry will be added for F and contain the sequence of hops B-C-D-E-F, similarly for C, C-D-E-F and likewise for D. These nodes previously had not communicated with F and hence did not know of a path to reach it. Therefore, if they needed to send any messages to F, they would have to use the clustering approach described before to find a path to F. With them saving a route by extracting it from a message that contained a route itself, they have potentially saved themselves the trouble of finding a route. These hosts are also capable

of answering the route problem of messages that do not contain a route and save them from going through the clustering process.

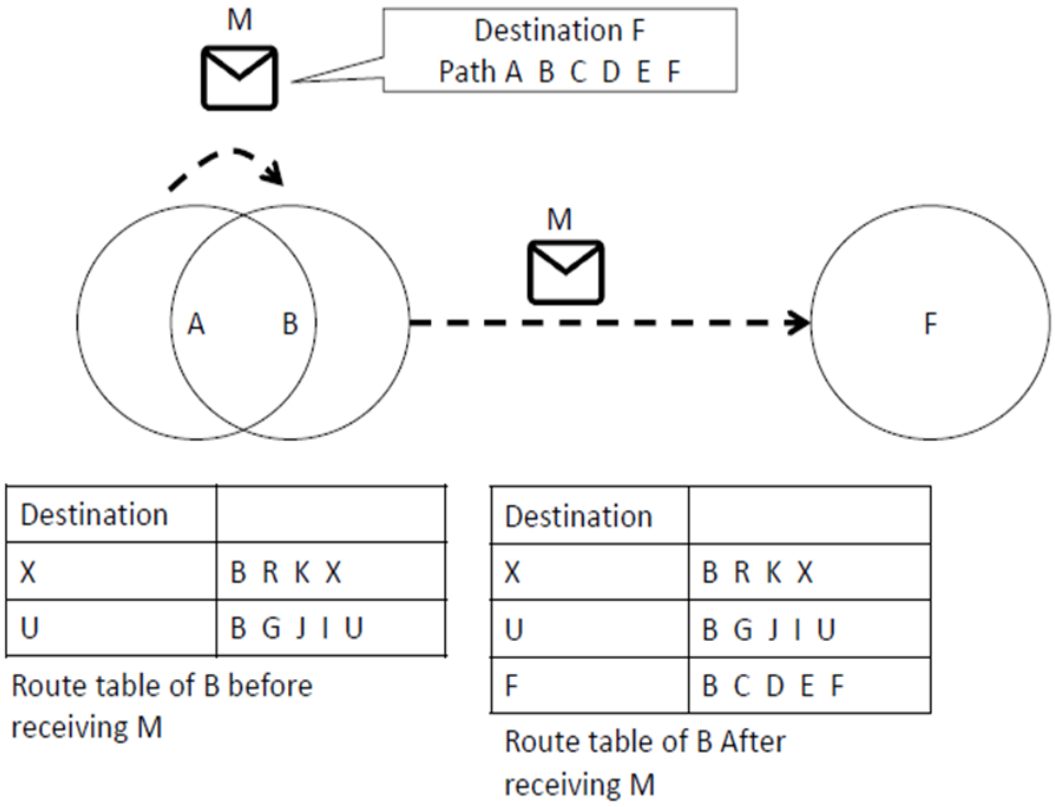

Figure 3.3 Path saving during message transfer

As a result of such methods of message transfer, after a time many hosts will contain routes to many other hosts. However, these routes in the table are passive until they are either used by the host to forward a message it created, or to answer the route problem of packets looking for their destination. As a

consequence of this, after a node has not communicated with another node for which it has an entry in its routing table, it is likely that the route has gone stale and is no longer valid. To update these old routes which are no longer connected due to mobility of nodes in the network, we introduce our local correction algorithm.

```
//Message Forwarding by path
Procedure ForwardByPath :
ForEach message k, do
      routes.add(k.destination,k.getHops())
      Node n = k.getNextHop()
      boolean found = false;
      ForEach encountered node d, do
            If n = d
                  then sendMessage(k,d)
                        found = true
                        break
            EndIf
      EndForEncounter
      If found = false then
            If routes.contain(k.destination) then
                  hops = routes.get(k.destination)
                  k.replaceHops(hops)
                  found = true
            EndIfRoute
            ElseIf routes.contains(k.intermediateHops) then
                  hops = routes.get(k.intermediateHop)
                  k.updateHops(hops)
            EndElseIf
            Else
                  k.removeNextHop()
                  LocalCorrection(k)
      EndIfFound
EndForMessage
EndProcedure
```

Figure 3.4 Pseudocode of message transfer through saved path

3.2.3 Local Correction

The main idea behind this operation is simple. Let us try to visualize it with an example. Say a node A uses a route for node F that it saved from a previous exchange of messages. Now upon using the path to forward the message, it sees that the next hop is missing. Immediately it removes entry for F in its route table. This is our only method of removing routes from a node's route table. If the message has traversed the path but detected a breakage later on, then we attempt to repair the route. We assume the break is only a local one, that means an intermediate node cannot find the next hop specified in the message, but that does not mean the rest of the path is broken as well. Suppose in our example, the path of the message M is A-B-C-D-E-F. Consider currently that C has the message and cannot find D in its transmission range. It then assumes that the rest of the path, E-F is still valid and is optimistic that forwarding the message to E can ensure its delivery to the destination F. E may or may not be in transmission range of C, it first needs to verify this. If found then it is transferred to E, if not then we need to find a route to E. A special packet called correction is created which finds the route to E by undergoing the clustering approach. Once it reaches E, it generates a local response that returns to C with a path to E. While this process of route

correction is taking place, the message sits in the buffer of the host. The host starts a timer of 30 seconds for the message for the time it remains waiting for a correction to come. If the timer expires and the host has not received a local response, then C clears the route of the message and message is forwarded by the clustering approach. Otherwise, if the local response reaches C successfully, it then adds this path to message in place of D (the node that went missing) and continues forwarding the packet with this new corrected route. Upon the delivery of the message to F, F will notice a change in the hops traversed by the message compared to the route in its table. F will then update the entry for A with this new path. Once F sends a message back to A, A also can update its old path with the new one.

```
//Local Correction of paths
Procedure LocalCorrection (Message k):
    If k.getHops().size < 2 then
        k.removeHops()
    EndIf
    Else
        Node n = k.getNextHop()
        correctingMessages.add(k)
        RouteRequest(n)
    EndElse
EndProcedure
```

Figure 3.5 Pseudocode of local correction process

In some cases, the local correction process fails because of changes in the topology during the local response phase leads to the route that the message is carrying being invalidated. The last host to receive the message

but unable to send to the next hop does not call local correction for the local response packet because this may cause a recursion. Imagine a local response packet which cannot find its next hop, it will call local correction process which will create a local correction packet that will find the appropriate next hop and create another local response packet to carry the route back. If this local response packet detects a break in the path as well it will call local correction again. If somehow after all this the route does become corrected and the original local response does find its way to the host that called the local correction in the first place, chances are the route will be consisting of too many hops, or the message it wants to correct has already been dropped from the buffer. or the already forwarded by the clustering approach.

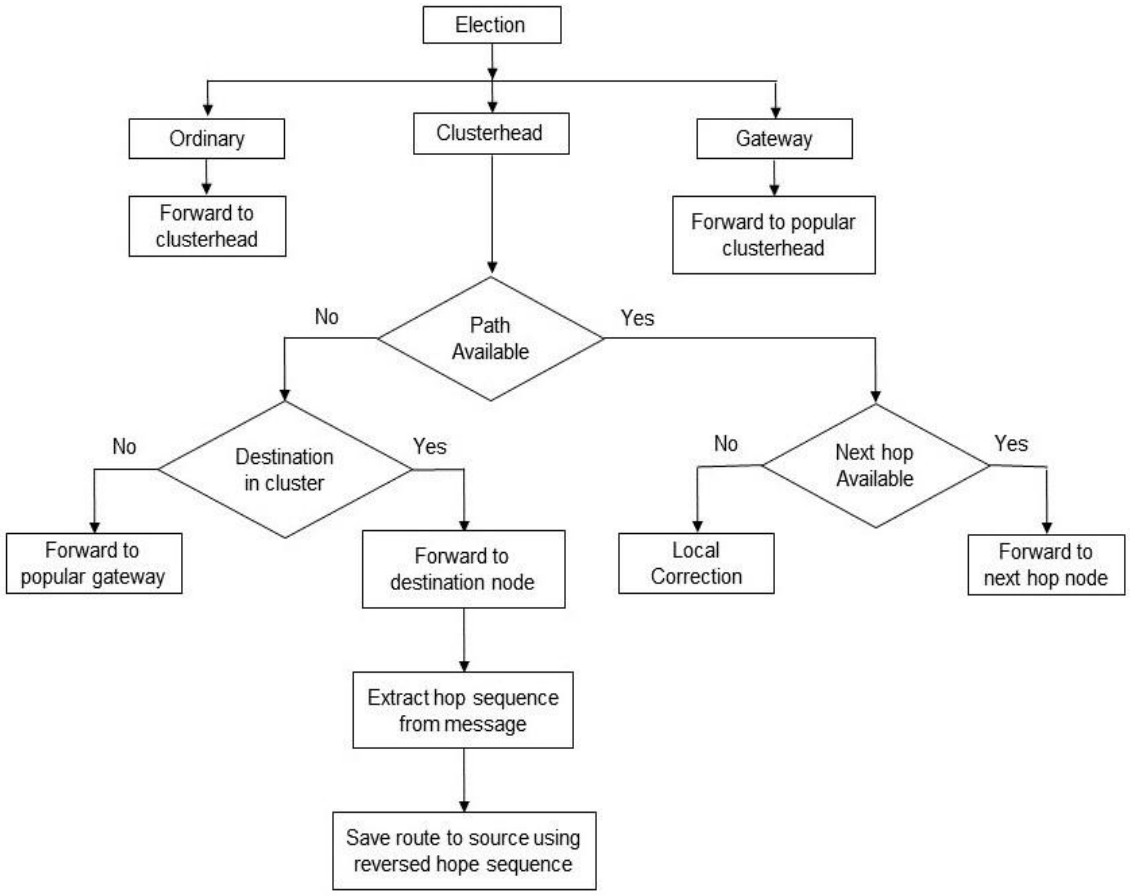

Figure 3.6 Flow Chart for Zone Cluster algorithm

The above flowchart summarizes the entire working process of our algorithm. Each node has a defined role and has different responsibilities in handling the forwarding of messages. Whenever any node receives a message with a saved path, it will add the route carried by the message in its route table. If the message cannot be forwarded to the next hop on the path, the node will call the local correction method to fix the broken path and forward the

message following new path. If no route is available for the message, the node will forward it using the clustering approach.

CHAPTER 4

Result analysis

In our simulations, we have used The Opportunistic Network Environment (ONE) simulator. This simulator was ideal for us as it is written in Java, a programming language that we are strong in. The simulator was made with the intention of evaluating different routing protocols specifically made for Delay Tolerant Networks (DTN). The protocols that we have used to compare the performance of our own protocol are Epidemic, First Contact, Direct Delivery, MaxProp [25] and Prophet.

The simulation was done on a map of the city of Helsinki, Finland. This was the default area given in the ONE simulator. Under such parameters we were unable to obtain simulations for MaxProp and Prophet. These protocols were simulated with 128 hosts keeping all other factors constant where they showed impressive delivery probability of ~1. However, they did not scale with the increase in the number of hosts and simulations could not complete with the numbers that we wanted. We are not aware of whether this is a limitation of the ONE simulator or the protocols themselves. One of the reasons could be that these protocols require hosts to maintain large amounts of data and require high computational power that was not feasible for such a

large scale networks. Consequently, we have excluded these two protocols from our discussion from henceforth.

We have considered the following parameters for all the protocols.

Table 4.1 Parameters used in simulation

1. Number of hosts	250,500,750,1000
2. Duration	1 day = 86400s
3. Message buffer	30 users with 50M and the rest with 20M.
4. Message TTL	5 hours
5. Message size	500kB - 1MB
6. Movement area	4500mx3400m
7. Movement model	ShortestPathMapBasedMovement and MapRouteMovement
8. Wi-Fi speed and range	30m with 250Kbps
9. Messages created	one new message every 25 to 35 seconds

In evaluating routing protocols for suitability in large ad-hoc networks such as an Ad-Hoc WAN, there are several key criterions that we need to observe.

4.1 Overhead Ratio

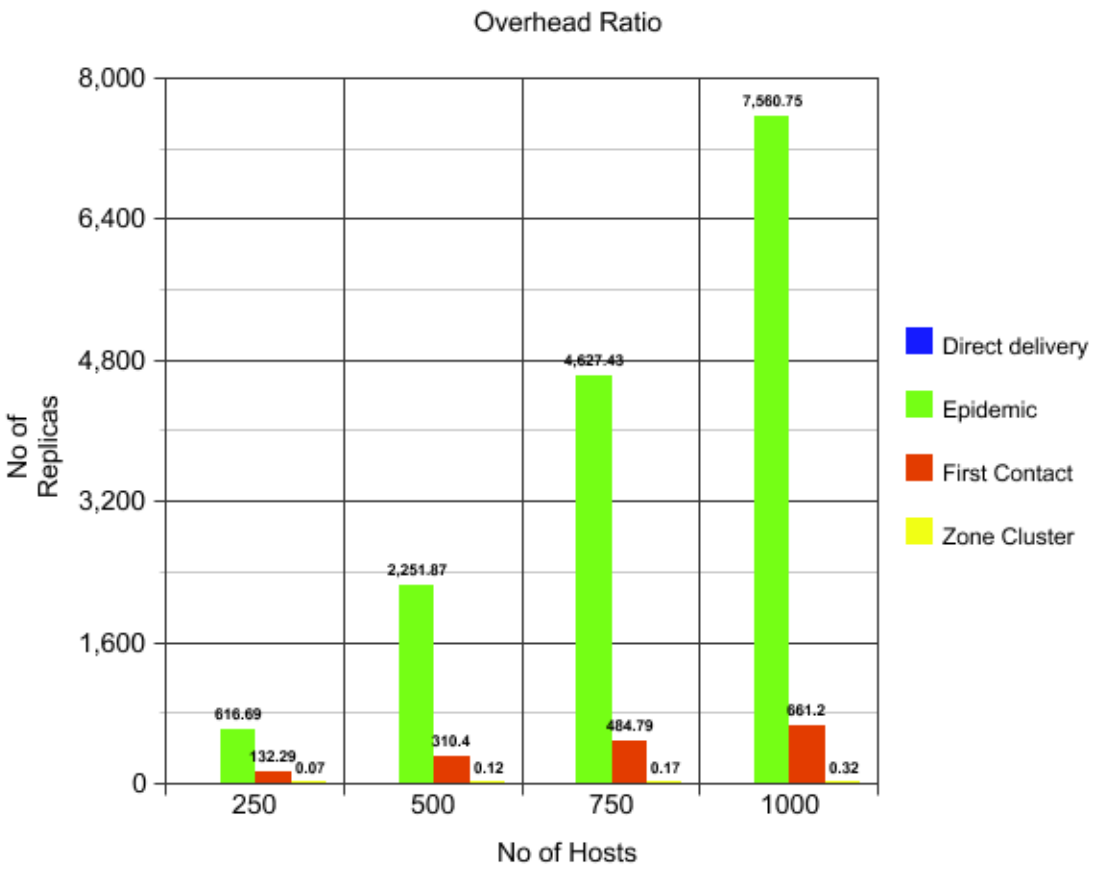

Figure 4.1 Message replicas vs no of hosts

The above graph compares the different protocols in terms of overhead ratio by varying the number of hosts in the network. The hosts are increased from 250 to 1000 in increments of 250.

The term overhead ratio is defined as the number of messages relayed to the number of messages delivered. Essentially, if the protocol is a multi-copy one, then it will transmit many copies of a message which results in a large number of the same message being relayed only for one copy of the message to be delivered. Single copy on the other hand reduces these large amounts of message relays extensively. It is desirable to have a low overhead ratio to minimize the hop count of the message, reduce latency and network congestion.

Epidemic creates unlimited copies of messages and really does not scale well at all with network size. The rate of increase in overheard increase in hosts is too large. A single message will have multiple copies which are relayed resulting in large overhead ratio. Such protocol would create too much congestion in the network.

First Contact does better in this regard as it is a single copy. However, it also increases slowly with network size. More hosts mean frequent contacts which means more relays occur.

Direct Delivery has no overhead as it only delivers to the destination. No message is relayed, only delivered.

Our ZoneCluster fairs very well because it uses metrics discussed previously which determine the best node to forward to. Doing so allows the messages to be relayed in a decisive way rather than being sent in a random way. Hence the number of message relays is significantly reduced. When the network size increases our overhead ratio sees negligible change.

4.2 Latency

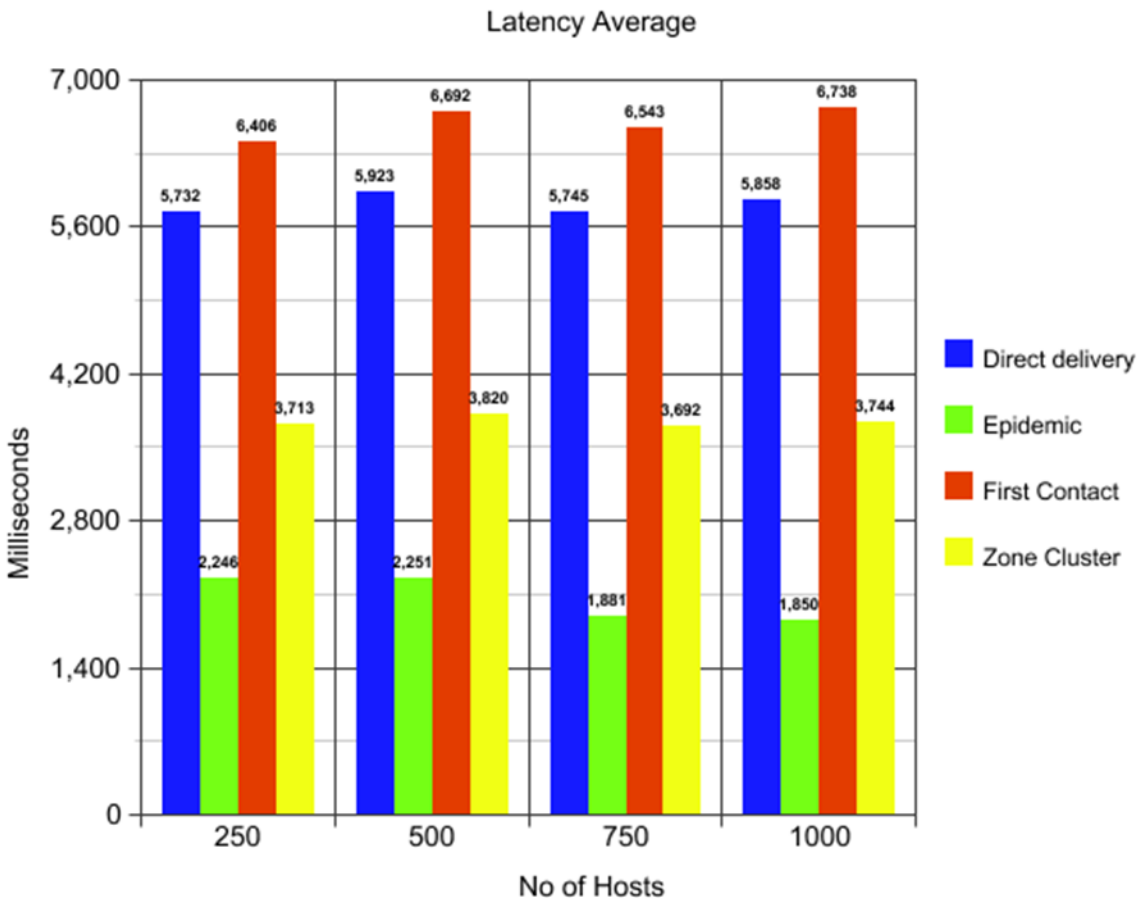

Figure 4.2 Latency time vs no of hosts

The above graph depicts how the latency varies for the different routing protocols for different number of hosts. Hosts were varied from 250 to 1000 in increments of 250.

Latency is the time between the creation of a message and its delivery. It is desirable to have a low latency to ensure fast exchange of messages that are required for certain applications.

Epidemic wins in this category as it floods the network and essentially every copy is on a race against time to reach the destination first. As a result, one copy is bound to get forwarded in the ideal sequence of hops and get delivered before all others in a short period of time. The latency decreases with an increase in network size so it is favorable for larger networks in this regard.

Direct Delivery and First Contact show a trend of increase in latency with size. In both cases, this is a matter of the mobility of the nodes. If the source and destination nodes are moving towards each other or are in range of one another than direct delivery performs well but this is more often not the case. Same goes for First Contact, the nodes forward messages to anyone they come in contact with, which may or may not move towards the

destination. Hence these protocols cannot provide consistency in real world scenarios.

Finally, our ZoneCluster fares better as it sends messages with routes but it slowed down by the invocation of the local correction method during times when the route breaks. While the path is being corrected, the message waits in the buffer which adds to the latency.

4.3 Buffer Time

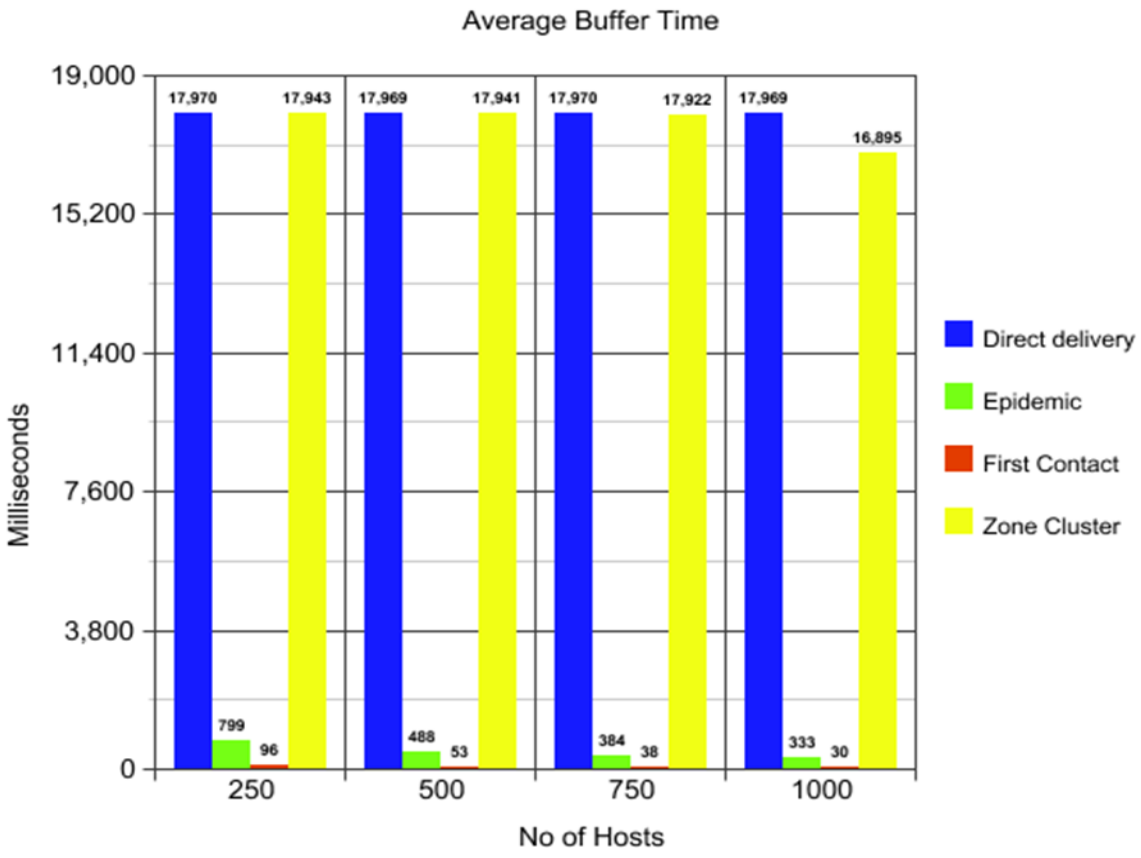

Figure 4.3 Buffer time vs no of hosts

The above graph shows the comparison of average buffer time between the different routing protocols. The number of hosts have been varied from 250 to 1000 in increments of 250 and the corresponding changes in average buffer time of each algorithm have been compared.

Buffer time is the time spent by messages in the buffer of hosts before they are deleted. Messages can be deleted from hosts after reaching their destination, after their TTL expires or the host's buffer is full to receive new messages.

Epidemic and First Contact show remarkably low average buffer times for all varying number of hosts. As the host number increases, their buffer time decreases which is desired in case of networks with large number of hosts. These protocols fair well because they do not have to make any routing decisions. As soon as a host receives a message it can forward to any node it encounters. As a result, messages spend very little time waiting in the buffers.

For Direct Delivery, as the number of hosts increases the buffer time remains unchanged. This means that the size of the network does not affect the buffer time. This is because a node must wait to come into contact with the destination before forwarding the message. If a node carrying a message

never comes into contact with the destination node of the message, then the message reaches its TTL and is dropped.

For ZoneCluster protocol, the buffer time shows a similar trend to that of Direct Delivery for hosts 250, 500 and 750, however for host number of 1000, the buffer time decreases. This decrease will continue as the number of hosts increases which is favorable for large networks. The clusterhead upon receiving a message not destined for any node in its cluster has to forward to a desired gateway. If the gateway is not present due to absence of nodes creating isolated clusters, then the message stays in the clusterhead till a gateway is available or if the destination comes into contact with the clusterhead. As the density of nodes increases, chances of isolated clusters decreases and hence gateways become more available.

4.4 Delivery Probability

The graph below shows the comparison between the different routing protocols with increasing number of hosts, from 250 to 1000 in increments of 250.

Delivery probability is the most crucial factor when evaluating any routing protocol as it is an indication of the reliability of a routing protocol. It

is measured as a ratio of the number of delivered messages to the total number of messages created. If messages do not reach their destination, the source may have some sort of retransmission mechanism for sending the same messages again and again till they were delivered. This will would be a costly operation on the network.

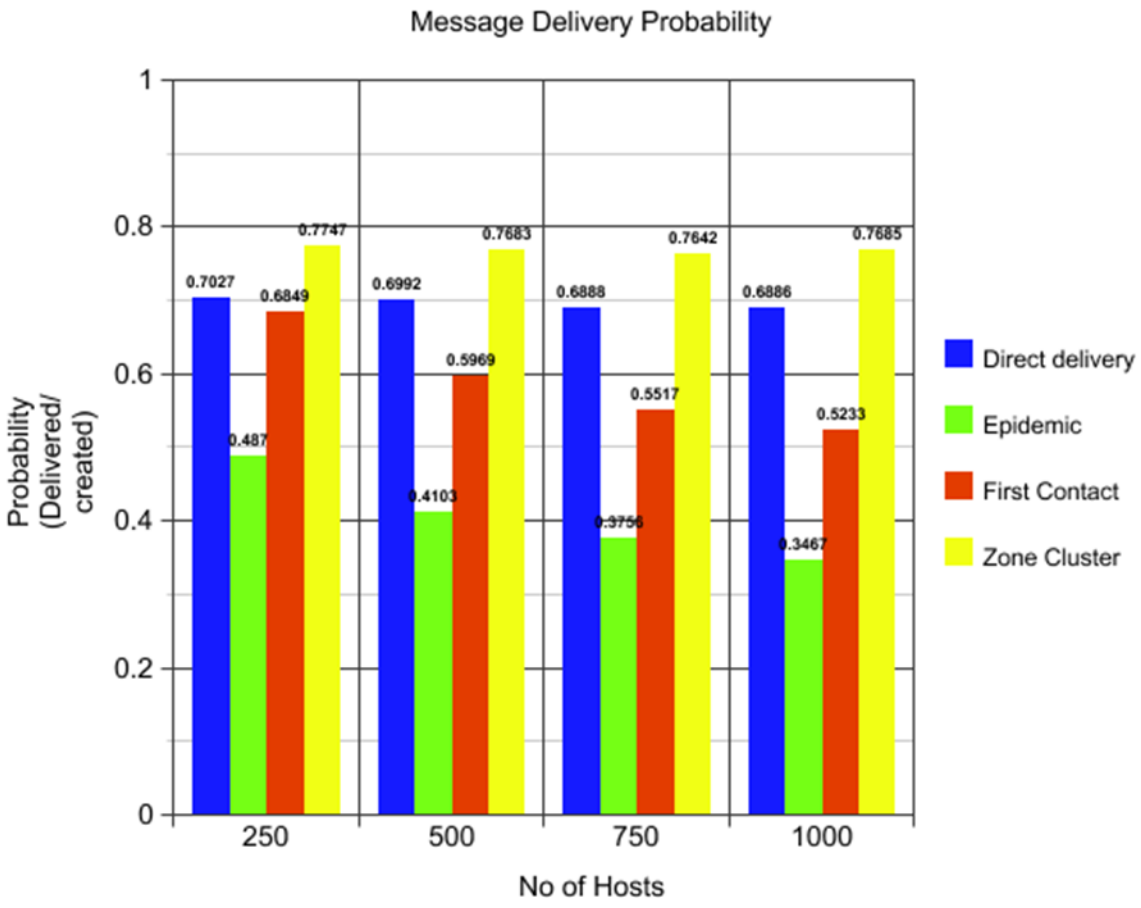

Figure 4.4 Delivery Probability vs no of hosts

ZoneCluster is the best in this category. Forwarding messages to clusterheads that control routing for a large number of nodes means that other nodes do not need to take any routing decisions. As we have a well-structured network where each node's role is defined, a hierarchy is imposed on the network that ensures every operation runs smoothly; from finding the routes, to maintaining and repairing them while forwarding messages. It shows that selecting the most popular clusterheads and gateways to forward messages ensures a high probability of messages reaching their destination.

Direct delivery is also good with about 10% less messages delivered than ZoneCluster across the different combination of hosts. Changing the size of network does not affect the delivery a great deal as it mainly depends on the relative movement of source and destination nodes.

First Contact suffers from a drop in delivery as nodes forward the messages randomly and more often. The increase in network size increases this random effect making it less likely to be delivered.

Finally, Epidemic suffers from the lowest delivery rate of them all. Flooding says that it will ensure the eventual delivery of the message but with

limitations of buffer space, many messages are dropped as simply there are too many messages in the network for the nodes to carry.

4.5 Summary of simulation

Table 4.2 Comparison between different protocols

	Direct Delivery	Epidemic	First Contact	Zone Cluster
Overhead Ratio	None	Very large	Medium	Very Small
Latency	High	Lowest	Highest	Medium
Buffer Time	High	Very low	Very low	High
Delivery Probability	Medium	Low	Low	High

The table is a summary of our findings from the simulations conducted.

Direct Delivery has shown to have moderately good delivery probability and no overhead ratio. However, with the latency and buffer time being high and the message only being forwarded with a hop count of 1, it is not suitable for long distance communication.

For Epidemic routing, the latency is the lowest out of all the protocols and buffer time is also very low. However, it is not scalable as with increase in network size as the overhead ratio increases a tremendous amount and the delivery probability suffers thus not being suitable for large networks such as a WAN.

In the case of First Contact, it achieves very little buffer time however in the other three categories it performs very poorly. Delivery probability is low coupled with the highest latency of all the protocols and a significant overhead ratio makes it a poor choice for our purpose.

CHAPTER 5

Conclusion and Future work

5.1 Conclusion

For an intermittent connection like Mobile Ad-hoc network in DTN (Delay Tolerant Network), sending message to respective destination without flooding is very hard. This is because the carrier nodes are always moving and hence having no fixed path. In such situation if flooding is used network congestion easily occurs as multiple copies of same message linger in every node buffer. Hence forwarding message in a calculative way is an ideal approach. This can decrease congestion, lesser consumption of resources and also can lower the latency. With this in mind many have approached for different categories of algorithm but most of them are for a smaller area of network which cannot be implemented in the practical world. Keeping everything in mind we have proposed an algorithm called Zone Cluster that can works better as the size of the network increases and while maintaining a good delivery ratio as well as low latency.

5.2 Future work

Zone Cluster is efficient when it comes to work in a larger area of network. It is a single copy routing hence it does not cause network congestion and calculates every forwarding decision. Moreover, it does not require any internet to establish a connection with other nodes or to establish a path from source to destination. Hence, it is cheaper and affordable for all. For future purpose we can focus on increasing its efficiency in increasing its delivery ratio and decreasing latency. We also have plans to develop an app that can be installed on the phones of users and used for messaging between them.

REFERENCES

[1] Chuang, M., Yongjian, Y., Du, Z., & Zhang, C. (2014). Overview of Routing Algorithm in Pocket Switched Networks. 2014 Ninth International Conference on Broadband and Wireless Computing, Communication and Applications. doi:10.1109/bwcca.2014.44

[2] Johnson, D., Hu, Y., & Maltz, D. (2007). The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4.

[3] Dang, H.; Wu, H. Clustering and cluster-based routing protocol for delay-tolerant mobile networks IEEE Trans. Wirel. Commun. 2010, 9, 1874–1881

[4] Greg Bigwood, Tristan Henderson, Devan Rehunathan, Martin Bateman, Saleem Bhatti, CRAWDAD dataset st_andrews/sassy (v. 2011-06-03), downloaded from https://crawdad.org/st_adrews/sassy/20110603, https://doi.org/10.15783/C7S59X,Jun 2011

[5] Li, Y., Hui, P., Jin, D., & Chen, S. (2015). Delay-tolerant network protocol testing and evaluation. *IEEE Communications Magazine*, *53*(1), 258-266. doi:10.1109/mcom.2015.7010543

[6] Chuah, M., Yang, P., Davison, B., & Cheng, L. (n.d.). Store-and-Forward Performance in a DTN. 2006 IEEE 63rd Vehicular Technology Conference. doi:10.1109/vetecs.2006.1682801

[7] Alslaim, M. N., Alaqel, H. A., & Zaghloul, S. S. (2014). A comparative study of MANET routing protocols. *The Third International Conference on e-Technologies and Networks for Development (ICeND2014)*. doi:10.1109/icend.2014.6991375

[8] Lee, F. (2011). Routing in Mobile Ad hoc Networks. *Mobile Ad-Hoc Networks: Protocol Design*. doi:10.5772/13155

[9] D. Sivakumar, B. Suseela, and R. Varadharajan, "A survey of routing algorithms for MANET," *IEEE-International Conference On Advances In Engineering, Science And Management* (ICAESM -2012), pp. 625–640, 2012

[10] Sok, P., Tan, S., & Kim, K. (2013). PRoPHET Routing Protocol Based on Neighbor Node Distance Using a Community Mobility Model in Delay Tolerant Networks. *2013 IEEE 10th International Conference on High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing*. doi:10.1109/hpcc.and.euc.2013.175

[11] Amin Vahdat and David Becker. *Epidemic routing for partially connected ad hoc networks*. Technical report, Duke University, 2000

[12] Spyropoulos, T., Psounis, K., &Raghavendra, C. S. (2005, August). Spray and wait: an efficient routing scheme for intermittently connected mobile networks. In *Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking* (pp. 252-259). ACM.

[13] Keränen, A.; Ott, J.; Kärkkäinen, T. The ONE Simulator for DTN Protocol Evaluation. In Proceedings of the 2nd International Conference on Simulation Tools and Techniques, Rome, Italy, 2–6 March 2009.

[14] J. Y. Yu and P. H. J. Chong, “3hBAC (3-hop between Adjacent Clusterheads): a Novel Non-overlapping Clustering Algorithm for Mobile Ad Hoc Networks,” in proceedings of IEEE Pacrim’03, vol. 1, pp. 318–21, Aug. 2003

[15] Mbarushimana, C., & Shahrabi, A. (2007). Comparative Study of Reactive and Proactive Routing Protocols Performance in Mobile Ad Hoc Networks. *21st International Conference on Advanced Information Networking and Applications Workshops* (AINAW07).

[16] Sinha, P., Krishnamurthy, S., & Dao, S. (n.d.). Scalable unidirectional routing with zone routing protocol (ZRP) extensions for mobile ad-hoc networks. 2000 IEEE Wireless Communications and Networking Conference. Conference Record (Cat. No.00TH8540). doi:10.1109/wcnc.2000.904825

[17] Kawabata, N., Yamasaki, Y., & Ohsaki, H. (2017). On Message Delivery Delay of Epidemic DTN Routing with Broadcasting ACKs. 2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC). doi:10.1109/compsac.2017.176

[18] Gamit, V., & Patel, M. H. Evaluation of DTN Routing Protocols.

[19] McGeehan, D. (2016). ChitChat: An Effective Message Delivery Method in Sparse Pocket-Switched Networks. *2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS)*.

[20] A.Mei, G. Morabito, P. Santi, and J. Stefa. Social-aware stateless routing in pocket switched networks. *IEEE Transactions on Parallel and Distributed Systems*, 26:252–261, Jan 2015

[21] Ze Li and Haiying Shen. Sedum: Exploiting social networks in utility based distributed routing for dtns. *IEEE Transactions on Computers*, 62(1):83–97, Jan 2013

[22] A.Ker¨Anen, "opportunistic network environment simulator. Special assignment report, helsinki university of technology," Department of Communications and Networking, May 2008. available at WW.netlab.tkk.fi_tutkimus_dtn_theone_pub_the_one.pdf

[23] Dahane, A., Loukil, A., Kechar, B., & Berrached, N. (2015). Energy Efficient and Safe Weighted Clustering Algorithm for Mobile Wireless Sensor Networks. Mobile Information Systems, 2015, 1-18. doi:10.1155/2015/475030

[24] Johnson, D., Hu, Y. C., & Maltz, D. (2007). The dynamic source routing protocol (DSR) for mobile ad hoc networks for IPv4 (No. RFC 4728).

[25] Burgess, J.; Gallagher, B.; Jensen, D.; Levine, B.N. MaxProp: Routing for Vehicle-Based Disruption-Tolerant Networks. In Proceedings of the 25th IEEE International Conference on Computer Communications INFOCOM, Barcelona, Spain, 23–29 April 2006.