Identifying Contact Patterns In Intermittently Connected Networks

Master of Technology
in
Information Technology

By

Yogesh Piolet T
[Roll No. 08IT6018]

Under the supervision of

Dr. Arobinda Gupta
School of Information Technology
&
Department of Computer Science

Indian Institute of Technology, Kharagpur

Nov 2009

ABSTRACT

Delay Tolerant Networks (DTNs) exhibit properties that are significantly different from that of conventional networks such as intermittent connectivity, long/variable delay, high error rates and asymmetric data rates. Routing becomes a problem in DTNs because the conventional routing protocols fail with these properties of DTNs. The DTNs follow the store-carry-forward strategy for routing. The mobile nodes store and carry the message with them until the intermediate or destination node comes into contact with it.

Several routing protocols have been proposed for DTNs. One such routing protocol is the *Probabilistic strict ROuting using Contact Sequencing.* This approach takes advantage of the contact patterns that may occur when several mobile nodes would come into contact with each other regularly. An example for such a contact pattern is a university campus scenario where all the students of a particular course would meet at the classroom four times per week and at the laboratory once per week. This motivated us into finding a solution for the following problem: *Given a history of contacts between a set of nodes, determine if the contacts are repetitive or not. If yes, determine the period of repetition.*

To achieve this objective, we approach the problem in a two-step manner. We need to slice the non-partitioned contact history into meaningful partitions such that majority of the contacts repeat across the partitions. To do so, a metric which would determine the similarity between contact patterns becomes necessary. In the first step, we come up with two metrics which determine the similarity between contact patterns on a scale of 0 to 1. The robustness of the metrics has been thoroughly examined. In the second step, an algorithm has been proposed to solve the slicing problem. Experimentation for testing the correctness and feasibility of the algorithm is in progress. Given a contact history, several alternative methods could exist to slice the given contact history. However, some of these methods have a significant benefit over the others in terms of time complexity and space, which are both critical to the DTN. Once slices have been generated, it becomes easier to predict whether the contact history is repetitive or not. If the contact history is indeed repetitive, the period of repetition will be reported.

Table of Contents

1. Introduction

One of the fundamental assumptions with conventional networks is that there always exists a contemporaneous end-to-end path between source and destination. This assumption also takes into consideration the link failures occurring due to network topology failures, mobility of nodes etc. Even mobile ad-hoc networks (MANETs) assume that the network is mostly connected. A characteristic feature of this assumption is that a message would be dropped if the next hop is not in contact. Another assumption with conventional networks is that most networks have a very low end-to-end delay. Conventional networks also assume that intermediate nodes store the message only for processing.

But, there are networks which defy these fundamental assumptions. There exist networks where the contemporaneous end-to-end path may never exist. In such networks, the nodes come into contact with each other intermittently. Hence, such networks are called *Intermittently Connected Networks*. In such networks, the network is mostly a collection of partitioned regional networks. Although all the links of the end-to-end path may not be connected contemporaneously, portions of the end-to-end path may be formed at different time intervals. Communication in such networks, though challenging, is not impossible. Communication in such *challenged* networks can take place by buffering the message until the node comes into contact and forwarding the message to the node when it comes into contact immediately. However, such a challenged network would violate the assumption of transient storage of messages. Such networks would require each node to hold the message until the next node comes into contact with it. Since such networks take advantage of the contact between nodes at unscheduled times to exchange messages, such networks are also called *opportunistic networks*. Also, a penalty with such opportunistic networks is the large delay in delivering the messages which is mainly due to the large amount of time spent in waiting for the contact between the intermediate nodes to occur. Hence, the applications that operate on such networks must be capable of tolerating the large/variable delays due to the disruptions in the end-to-end path between source and destination node. Hence, such networks are popularly known as *Delay Tolerant Networks* or *Disruption Tolerant Networks* (DTN) [1]. DTNs have higher error rates due to

link failures and require correction or even retransmission. Correction of a message implies more processing time, while retransmission results in higher network traffic. DTNs also suffer from asymmetries in bidirectional data rates. Summarizing, DTNs exhibit the following properties: intermittent connectivity, long/variable delay, high error rates and asymmetric data rates [2].

1.1 Motivation

Routing becomes a problem in DTN because the conventional routing protocols fail with the inherent characteristics of DTN. The DTNs follow the store-carry-forward strategy for routing. According to this strategy, on receiving a message, a node could transmit the message immediately if it is in contact with the next node, otherwise it will store the message. The node will carry the message with itself until the next intermediate node comes into contact. The node will immediately forward the message when the next intermediate node comes into contact.

Several routing protocols have been proposed for DTNs already [5,6,9,11,12]. One such preliminary protocol is the *Epidemic* protocol [12]. In this approach, the message is transferred by a node to any intermediate node that comes into contact with it, provided the intermediate node does not have the message already. Thus, it tries to increase the data delivery probability. But this protocol creates too many replicas of the same message in the network, resulting in increase of network congestion. To overcome this extreme replication problem, there are some routing protocols which transmit the message only to a selected subset of nodes from the set of all nodes that come into contact. *Spray and Wait* [11] and *PROPHET* [9] are examples of such protocols. There exist many other routing protocols for DTNs following variations of these approaches [5,6,12].

Probability strict ROuting using Contact Sequencing (PROCS) [6] is a routing protocol that uses repetitive behavioral patterns in the contact history between the nodes to decide which node must be chosen as the next intermediate node. Consider the following example:

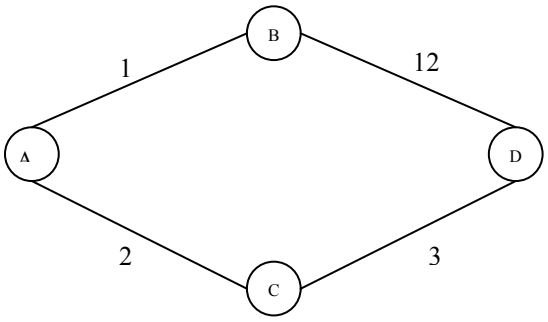

Figure 1.1: A contact graph

Let A, B, C, and D be four mobile nodes. A meets B at 1 pm and it meets C at 2 pm. B meets D at 12 pm. C meets D at 3 pm. Suppose A wants to transmit a message to D at any time before 1 pm, the obvious path to choose would be A-C-D. Even though A meets B first, it holds back the message until C comes into contact because C would probably deliver it to D by 3 pm while B would have done it only by 12 pm. Since the sequence in which the nodes meet is a basis for the routing decision, this protocol is called *Probabilistic strict ROuting using Contact Sequencing*. This achieves a high delivery ratio and low replication of messages. A graph depicting such information about contacts between nodes is called contact graph (as shown in Fig. 1.1)

Another example of a protocol which uses repeating behavioral patterns in the contact history between the nodes is *Probabilistic ROuting Protocol using History of Encounters and Transitivity* (PROPHET) [9]. Each node maintains a delivery predictability to every other node. A delivery predictability to a node *i* indicates the probability with which the current node would deliver the message to node *i*. When two nodes meet, they exchange their predictability to update their own delivery predictability and all the other transitive predictability to other nodes. A message will be passed to another intermediate node only if the latter has higher delivery predictability to the destination node. This method replicates message in a limited manner, unlike Epidemic routing.

The routing protocols such as PROCS and PROPHET take advantage of the knowledge regarding when two nodes would meet each other. But, where do they get this information from? These routing protocols assume the existence of knowledge oracles [6], each of which is capable of

answering the questions that are posed to it. The knowledge oracle which has the relevant information to determine when two nodes would meet each other is called contact oracle.

One of the challenges lying ahead is to eliminate the assumption of the existence of such a hypothetical contact oracle and, to design and implement a system which has equivalent answering powers. Given the contact history between a set of nodes for a given period of time and if the contacts between nodes are repetitive after a period, it should be possible for the proposed system to predict when two specific nodes are going to meet each other. Once we have such a system in place, we need to deduce the contact graph and other related information from the contact history and supply it to the corresponding routing protocol.

1.2 Problem Statement

The objective of this thesis is to predict when a specific pair of nodes is going to meet in the future based on the contact history between the nodes. This is equivalent to the problem of determining if given a contact pattern history between nodes, whether the contact patterns between nodes are repetitive with a certain period or not. The problem statement can be formally stated as follows:

Given a history of contacts between a set of nodes, determine if the contacts are repetitive or not. If yes, determine the period of repetition.

To achieve this objective, we approach the problem in a two-step manner. We need to slice the non-partitioned contact history into meaningful partitions such that majority of the contacts repeat across the partitions. To do so, a metric which would determine the similarity between contact patterns becomes necessary. In the first step, we come up with two metrics which determine the similarity between contact patterns on a scale of 0 to 1. The robustness of the metrics has been thoroughly examined.

In the second step, an algorithm has been proposed to solve the slicing problem. Experimentation for testing the correctness and feasibility of the algorithm is in progress. Efficient slicing strategies need to be developed. Given a contact history, several methods could exist to slice the given contact

history. However, some of these methods have a significant benefit over the others in terms of time complexity and space, which are both critical to the DTN. Once slices have been generated, it becomes easier to predict whether the contact history is repetitive or not. If the contact history is indeed repetitive, the period of repetition will be reported.

1.3 Organization

This report is organized into the following sections. In section 2, we give literature survey of the area. In section 3, we propose two metrics to determine the similarity between two contact patterns. In section 4, we propose a brute force based slicing algorithm. In section 5, a few conclusions are drawn from the work done until now. Finally, in section 6, we present the future directions of our work.

2. Related Work

The Delay Tolerant Network (DTN) exhibits properties such as intermittent connectivity, long/variable delay, high error rates and asymmetric data rates, which do not agree with the fundamental assumptions of networking. Fall et. al. [2] proposed an architecture for the DTN that allowed both DTN and other networks which followed the assumptions to operate together successfully. The architecture was flexible enough to allow each network stack to use the protocols that best suited its needs. But, to achieve interoperability with other networks, an overlay network protocol was added between the network stack and the applications. The overlay network protocol would work with all types of networks because it includes as little mandatory elements as possible. Fall et. al. identified three fundamental principles of the DTN architecture: Firstly, query/response or conversational form of communication was unsuitable for DTN due to large delay. Hence, all the metadata required to satisfy a request would be *bundled* together into a single message. Secondly, DTN architecture relies on tiered functionality of the underlying protocols and thirdly, DTN protocols transmit as little information as possible because bandwidth may not be cheap.

A few of the existing DTNs are mentioned below. One of the most popular DTNs is the *Interplanetary Internet* [2] where communication between the source and destination (planets, namely Earth and Mars) is carried out by the intermediate nodes (namely orbiting satellites) which come into contact with each other intermittently. Another example of a DTN is an experimental DTN project called *DakNet* [10] which facilitates asynchronous digital communication in remote villages of central India and Cambodia. The facility is provided by buses/motorbikes or even bicycles which are mounted with mobile access points (MAP). The kiosks in the villages and the MAPs on the vehicles exchange information when they come into contact with each other. Such an exchange of information occurs each time the vehicles come into contact with the kiosks in the other villages or towns. Thus, the asynchronous communication is achieved between the source and destination (villages or towns) through the intermediate nodes (vehicles). Another potentially commercial application is the project *FleetNet* [3] deployed in Europe. *FleetNet* aims at developing a communication platform for inter-vehicle communications. Such a platform could offer multiple ranges of services. It could offer emergency notification such as informing the driver about the accident on his/her route. It could dissipate necessary information such as the traffic jam status on his current route. It could also offer other communication/information service such as inter-car chat or advertise fuel prices of the next service station. Another compelling application of the DTN is the *ZebraNet* [7] project in central Kenya. *ZebraNet* tries to answer a biological research problem about long-range migration, inter-species interactions and nocturnal behaviour of the Zebra. The node in this DTN is the tracking collar carried by the Zebra which contains a GPS, Flash memory, wireless transceivers and a small CPU. The logged data is then percolated towards a base station (destination) through the interconnected nodes (tracking collars in each Zebra).

Vehdat et. al. [12] proposed a simple routing protocol for DTN called *Epidemic* protocol. In this approach, the message is transferred by a node to any intermediate node that comes into contact with it, provided the intermediate node does not have the message already. Thus, it tries to increase the data delivery probability. But, this protocol creates too many replicas of the same message in the network, resulting in increase of network congestion. Another variant of epidemic is the *Spray and Wait* protocol proposed by Spyropoulos et. al. [11]. In this approach, the source node will spray a

limited number of copies of the message into the network and waits for one of these nodes to meet the destination.

However, there are some routing protocols which transmit the message only to a selected subset of nodes from the set of all nodes that come into contact. Sushant Jain et. al. [5] proposed another approach with a low delivery probability and less network congestion. In the *First Contact* approach, the message is transferred to only the first node that comes into contact.

Lindgren et. al. [9] proposed the *Probabilistic ROuting Protocol using History of Encounters and Transitivity* (PROPHET) routing protocol. It uses repeating behavioral patterns in the contact history between the nodes. Each node maintains a delivery predictability to every other node. A delivery predictability to a node *i* indicates the probability with which the current node would deliver the message to node *i*. When two nodes meet, they exchange their predictability to update their own delivery predictability and all the other transitive predictability to other nodes. A message will only be passed to another intermediate node only if the latter has higher delivery predictability to the destination node. This method replicates message in a limited manner, unlike Epidemic routing.

It has been observed that patterns emerge from the movement of mobile nodes in many real life scenarios. Considering human beings as the mobile nodes, Gonzalez et. al. [4] have proved that the human beings follow simple reproducible patterns in their movement. Also, a spatial probability distribution was specified for such patterns in movement. Kim et. al. [8] have explored the mobility characteristics in traces of mobile users and discovered that speed and pause time of the mobile users, follow a log normal distribution. They also observed that the direction of the movement was affected by the direction of roads and walkways. Focusing on the movement of people in popular areas, they have also developed a mobility model which gives movement traces that resemble the real user traces.

3. Similarity Measures

A challenging question on the basis of which many delay tolerant network (DTN) routing protocols function is to determine who (amongst all nodes in contact) should be chosen to pass on the message. A few such strategic DTN routing protocols are *PROPHET* [9], *PROCS* [6] and others. However, these protocols have assumed the existence of many knowledge oracles [5], each of which is capable of answering the questions posed to them. The knowledge oracle that knows the information necessary to answer the above question is referred to as contact oracle. Contact oracles can answer any question regarding contacts between any two nodes at any specific instant of time. Contact oracle has the information about when two nodes are going to meet each other in the future.

One of the challenges lying ahead is to eliminate the assumption of the existence of such a hypothetical contact oracle and, to design and implement a system which has equivalent answering powers. Given the contact history between a set of nodes for a given period of time, it should be possible for the proposed system to predict when two specific nodes are going to meet each other. In order to perform this task, it becomes necessary to recognize the repeating behavioural patterns existent in the contact history and slice the given contact history into a set of periodically repeating sequences referred to as contact patterns. The points where contact history is sliced to obtain the contact patterns are called slice points. The recognition of patterns in an exhaustive contact history and subsequent identification of slice points is a challenge in itself. The identification of slice points would become easier if a metric existed which could measure the similarity between any two slices of the contact history. Thus, the first step is to come up with such a similarity measure.

3.1 Formal Specification of Slice Structure

The contact history is a 3-tuple list <*src, des, t*> indicating a mobile node *src* meets another mobile node *des* at time *t*. Two slices *X* and *Y* that are generated from the contact history and are to be compared can be represented in the following form –

X					Y			
src	**des**	**time**	**frequency**		**src**	**des**	**time**	**frequency**
A	B	X_1	F_1		A	B	Y_1	G_1
B	C	X_2	F_2		B	C	Y_2	G_2
.	.	.	.		.	.	.	.
.	.	.	.		.	.	.	.
F	H	X_m	F_γ		G	H	Y_n	$G_ß$

where, $\{A, B, C…\}$ is the set of mobile nodes that are involved. Time $t_{min} \leq X_i \leq t_{max}$ and $t_{min} \leq Y_i \leq t_{max}$ are the times at which the corresponding nodes meet each other in slice X and Y respectively. t_{min} and t_{max} are the minimum and maximum time of both the slices. Frequency F_i or G_i are the number of times the corresponding node pairs $<src, des>$ occur in the slice X or Y respectively. m and n are the number of contacts in slice X and Y respectively. γ and ß are the number of unique node pairs in contact in slice X and Y respectively.

3.2 Formulation of Similarity Measures

Given two slices from a contact history, the similarity measure **η** Є [0, 1] is a metric that is used to describe how closely the slices match with each other. The closeness must encompass vital characteristics such as the difference in time(s) at which specific node pairs meet and the frequency with which specific node pairs meet. The similarity measure may also give insight on the causality in the contact among node pairs implicitly or explicitly.

The similarity measure **η** must satisfy the following properties –

1. For any slice X, **η(X,X) = 1** and **η(X,Φ) = 0**
2. Commutative property: For any two slice X,Y, **η(X,Y) = η(Y,X)**
3. Learning (Preservation of similarity) property: A non-zero **η** for any two slices X and Y can never be brought back to zero by any number of insertions and deletions of contacts in any of the slices, unless all contacts with equal node pair in both slices are deleted.

We propose two such similarity measures namely Euclidean distance based similarity measure and set similarity measure. Then, we evaluated the suitability of these two similarity measures under different scenarios.

3.2.1 Euclidean Distance Based Similarity Measure (η_1)

The Euclidean distance based similarity measure takes two factors into account. Firstly, it considers the frequency with which the contact between the same pair of nodes is occurring. Secondly, how close are the contacts between corresponding node pairs in both the slices occurring temporally? The frequency factor of contacts between the same pair of nodes is measured using set similarity and is controlled by the weight *q*. The temporal closeness factor of the contacts between corresponding node pairs occurring in both the slices is measured using Euclidean distance between the times of contact occurrence in the corresponding node pairs and is controlled by the factor *p*.

The Euclidean distance based similarity measure is formulated as follows –

$$\eta_1 = p * \frac{\sum_{i,j} \min(F_i, G_j)}{\sum F + \sum G - \sum_{i,j} \min(F_i, G_j)} * \left(1 - \frac{\sqrt{\sum_{i,j}^{n} (X_i - Y_j)^2}}{\sqrt{n * (t_{max} - t_{min})^2}} \right) + q * \frac{\sum_{i,j} \min(F_i, G_j)}{\sum F + \sum G - \sum_{i,j} \min(F_i, G_j)} \quad \ldots(3.1)$$

where, *n* is the number of node pairs that occur in both the slices *X* and *Y*. $(X_i\text{-}Y_j)$ is the time difference in the contact of the corresponding node pair such that the corresponding node pair occur in both the slices *X* and *Y*.

3.2.2 Set Similarity Measure (η_2)

The similarity measure η_2 is calculated on a 3-tuple equality basis <*src, des, time*> unlike in η_1, where frequency is calculated over equality of 2-tuples <*src, des*>. Hence, frequency tuple becomes redundant in the calculation of η_2. The similarity measure η_2 imposes a restriction upon the node

pairs occurring in both the slices to meet each other at the same time to show an improvement in similarity. On the other hand, η_1 allows room for temporal variations in the contact of same node pairs in both slices. Considering slice *X* and *Y* as multi-sets, the set similarity measure is formulated as follows -

$$\eta_2 = \frac{X \bigcap Y}{|X| + |Y| - X \bigcap Y} \qquad \text{.....(3.2)}$$

3.2.3 Evaluation of Similarity Measures

The slices were generated in the following manner: The number of meetings, the number of unique node pairs and the maximum time were taken as input for the slice generator. The slice generator was also parameterized on the number of unique node pairs to which we are biased and the degree of bias to each unique node pair. Once a slice *X* was generated by the fore-mentioned slice generator, the slice *Y* can be generated by manipulating slice *X*, treating *X* as a seed. Each contact in slice *X* can be manipulated by the following operations: retain contact (the contact in slice *X* is placed into slice *Y* without any change), dissimilar contact (The contact in slice *X* is placed into slice *Y* by retaining the same node pair and inducing a temporal dissimilarity/variation) and drop contact (The contact in slice *X* is not placed into slice *Y*). After the completion of slice *X* manipulation by parameterized percentages of each of the three operations, a parameterized percentage of *new contacts* are added into slice *Y*. The new contacts are generated randomly and may or may not be present in slice *X*. The experiments were performed for 50 contacts. The number of nodes involved was 10, resulting in a total of 45 possible unique node pairs. Amongst these 45 possible unique node pairs possible, only 10 were chosen randomly. The time for which the slices were generated, fell in the range [0, 23]. However, no bias has yet been imposed on any of the unique node pairs in a slice.

3.3 Results & Inferences

3.3.1 Observation of rising and falling rate of similarity measures

In this experiment, the slice *X* was generated and slice *Y* was kept empty initially. After the addition of each contact into slice *Y* by retaining contacts from slice *X*, the similarity measures were calculated to observe the rising rate. When the slice *X* became exactly same as that of slice *Y*, an equal number of new contacts were added into *Y* to observe the falling rate. The graph obtained is shown in Figure 3.1. From the results, we infer that both η_1 and η_2 increase uniformly to the addition of same contacts into slice *Y* and decrease non-uniformly to the addition of new contacts. It must be noted that η_1 and η_2 would never come back to 0, no matter how many contacts are added. Thus, the similarity measure has the property of learning (preserving) the similarity once achieved in a chronological time frame.

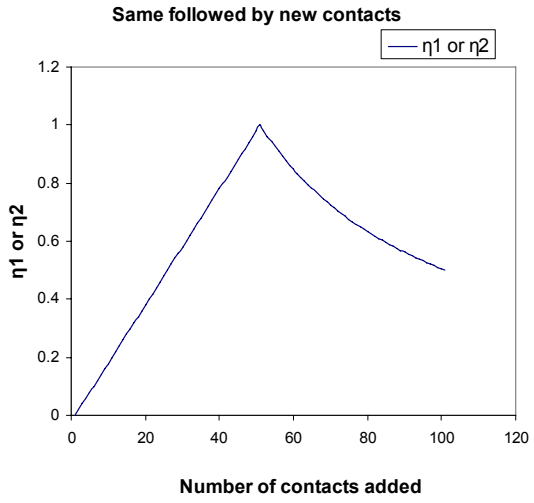

Figure 3.1: Same contacts followed by new contacts

3.3.2 Observation of effect of temporal variations on similarity measures

In this experiment, the slice *X* was generated and slice *Y* was kept empty initially. After the addition of each contact into slice *Y* by retaining node pairs and inducing temporal dissimilarity of varying degrees (say 0, 4, 8 and so on) from slice *X*, the similarity measures were calculated to observe the

rising rate. The results are shown in Figure 3.2 (a) and 3.2 (b) respectively. From the results, we infer that η_1 employs a continuous nature [0, 1] in determining the similarity of each contact added. It is tolerant to slight variations in time at which the nodes meet each other in the two slices. As per η_1, contacts are said to be varying between totally equal and totally unequal. On the other hand, η_2 employs a discrete nature [0/1] in determining the similarity of each contact added. It is absolutely intolerant to even slight temporal variations. As per η_2, contacts are said to be either equal or unequal, nothing in between. Two contacts are said to be equal only if they involve the same node pair and meet at the same time, unequal otherwise.

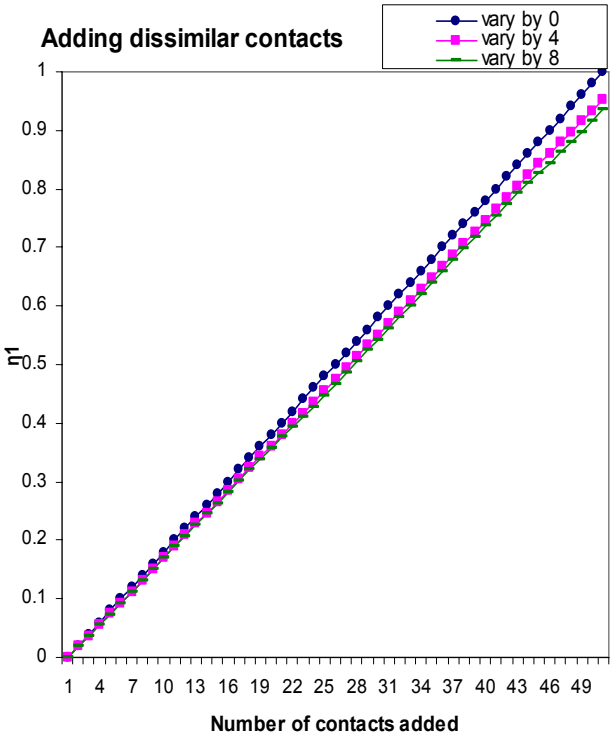

Fig 3.2 (a)

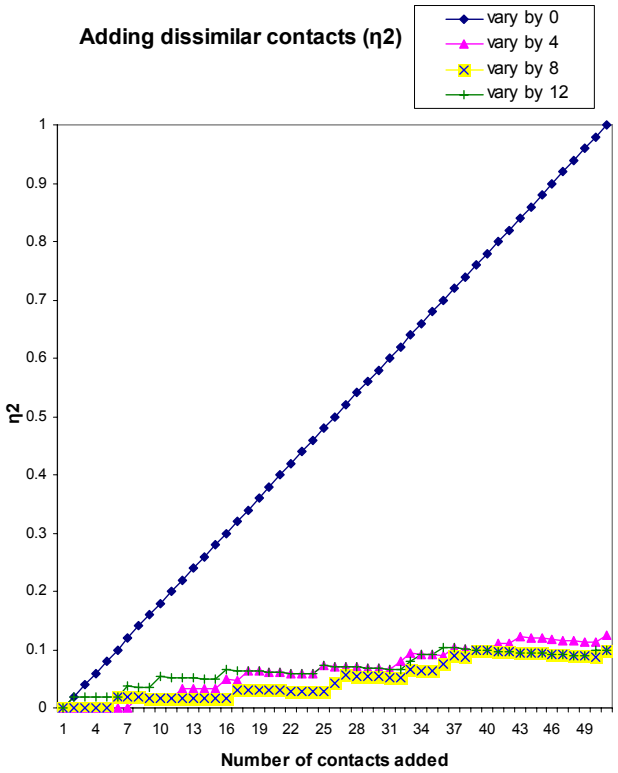

Fig 3.2(b)

Fig 3.2: Adding dissimilar contacts (a) Effect on η_1 (b) Effect on η_2

3.3.3 Observation of falling rate of similarity measures

In this experiment, the slice *X* was generated and slice *Y* was kept same as slice *X* initially. After the addition of each contact into slice *Y* by any operation (same, new or temporal variance), the similarity measures were calculated to observe the falling rate. The results are shown in Figure 3.3. From the results, it must be noted that though $\boldsymbol{\eta_1}$ and $\boldsymbol{\eta_2}$ are not falling uniformly; however, they do follow a pattern. For every i^{th} contact that is added into an initially equal slice *Y* with cardinality $|Y|$, the similarity measure falls to $\frac{|Y|}{i+|Y|}$. From the results, we infer that both $\boldsymbol{\eta_1}$ and $\boldsymbol{\eta_2}$ decrease harmonically when any contact is added to any one of already equal slices.

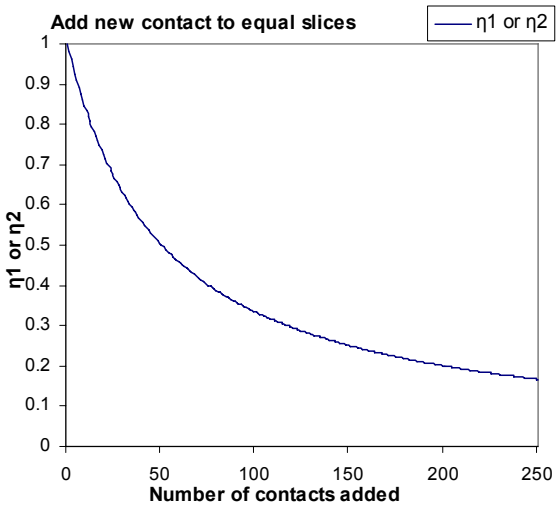

Figure 3.3: Addition of new contacts to equal slices

4. Slicing the Pattern

4.1 Formal Specification of the Slicing Problem

Slicing a contact history is the process of transforming a non-partitioned contact history into meaningful partitions (slices) such that majority of the contacts repeat across the partitions. Since a contact history can be partitioned at any time duration between 0 to $\frac{t}{2}$, it becomes necessary to

define a slice measure which would determine which time duration yielded the best slicing. Consider a pattern P that is sliced into n slices. We compare each pair of the n slices using the similarity measure $\boldsymbol{\eta_1}$. A threshold based slice measure is defined as follows –

$$\Phi(D) = \frac{x}{{}^{n}C_{2}} \quad \text{...} \quad (4.1)$$

ALGORITHM: Constant Time Duration Slicing
Input: A contact history of the form <*src, des, time*> which is time sorted.
Output: A single splice point (period after which most/all contacts repeat)

```
slice_point_found_flag = FALSE
For DURATION = 1 to (t+1)
        Slice the contact history for every DURATION units.
        For each slice
                For each contact entry
                        time ← time % (DURATION + 1).
                End for
        End for
        For each slice i
                For every slice j
                        comparison_matrix[i, j] ← η1 (slice i,slice j)
                End for
        End for
        x ← Number of entries in comparison_matrix above pre-specified threshold ß
        slice_measure[DURATION] = x / ⌈(t+1)/DURATION⌉C2

        if (slice_measure[DURATION] = 1)
                slice_point  = DURATION
                slice_point_found_flag = TRUE
                break
        End if
End for
if (slice_point_found_flag = FALSE)
        slice_point = smallest DURATION i with largest slice_measure value above α
End if
```

where $slice\_measure[DURATION] = \dfrac{x}{{}^{\left\lceil \frac{(t+1)}{DURATION} \right\rceil}C_2}$

Figure 4.1 Constant Time Duration Slicing Algorithm

where, n is the number of slices for duration D, x is the number of similarity measure values out of ${}^{n}C_2$ values that are above a pre-specified threshold ß. Suppose 80 out of 100 similarity measure values are above the pre-specified threshold ß for a duration D, then $\Phi(D) = 0.8$. The slice point must satisfy two conditions: Firstly, $\Phi(D)$ for a slice point D must be greater than α, where α is the pre-specified threshold percent of similarity measure values which must be above the pre-specified threshold ß. Secondly, if $\Phi(D_1)$ and $\Phi(D_2)$ are same for points D_1 and D_2, choose the minimum of D_1 and D_2. The slicing problem can be formally defined as follows:– Given a 3-tuple pre-processed contact history of the form <*src, des, time*> from time *0* to time *t*, the aim of the slicing problem is to find the smallest duration *SP* (slice point) $1 \leq SP \leq t+1$ using which the equal slicing of the contact history based on time results in a maximum slice measure $\Phi(SP)$ such that $\Phi(i) < \Phi(SP)$ for $1 \leq i \leq t+1$ and $i \neq SP$. Let α be the pre-specified threshold percent of similarity measure values which must be above the pre-specified threshold ß. A brute force based slicing strategy is shown in Fig. 4.1.

4.1 Experimental Setup

In order to evaluate the feasibility and correctness of the proposed algorithm, we are performing some experiments with the following setup: Using a pattern that is generated by the procedure mentioned in section 3.2.3 as a base, we generated a perfectly repetitive contact history by merely repeating the same base pattern for a fixed number of cycles. The proposed algorithm has been successful in identifying such a pattern as repetitive and has also successfully identified the slice point. In order to test the proposed algorithm on contact history which is not perfectly repetitive, we are using the following method: Rather than just repeating the base pattern exactly as it is, we modify the base pattern for each cycle by different operations resulting in three scenarios. The nodes which are already present in a slice are referred to as regularly meeting nodes. In the first scenario, we add a fixed percentage of node pairs that do not exist in the slice. This simulates the scene where nodes other than the regularly meeting nodes also meet in a specific slice. In the second scenario, we add a fixed percentage of same node pairs meeting at times other than those present in the slice. This simulates the scene where the regularly meeting nodes also meet other

nodes or they meet the same node again at a different time in addition to their existing meeting. In the third scenario, we only change the times at which the regularly meeting nodes are coming in contact with each other. This simulates the scene where the regularly meeting nodes meet at a time other than their usual time. Further, it is also possible that all these changes may not happen in all slices. To inculcate this, the fore-mentioned changes are made only on a fixed percentage of slices. We are currently in the process of determining appropriate values for parameters such as α and ß and are also testing the proposed algorithm on such non-perfectly repetitive contact history.

5. Conclusion

Euclidean distance based similarity measure gains an upper hand over set similarity measure because it is highly tolerant towards slight temporal variation by which the mobile nodes meet each other. However, both the metrics perform equally well in terms of robustness. Both metrics exhibit the property of preserving the similarity that once occurred for eternity. A simple brute force based splicing technique is proposed. Experimentation for testing the correctness and feasibility of the algorithm is in progress.

6. Future Work

- Efficient slicing strategies need to be developed. Given a contact history, several methods could exist to slice the given contact history. However, some of these methods have a significant benefit over the others in terms of time complexity and space, which are both critical to the DTN.
- Once slices have been generated, prediction about when specific pairs of nodes would meet each other in the future must be performed.
- Also, the current approach works in a centralized manner, in the sense that each node is assumed to know the contact history of every other node. Hence, a distributed approach would be a much more practical solution for DTNs.

References

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

[2] K. Fall, *"A delay-tolerant network architecture for challenged internets,"* in SIGCOMM '03: Proceedings of the 2003 conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, New York, NY, USA: ACM, pp. 27-34,2003.

[3] W. Franz, R. Eberhardt, T. Luckenbach, *"FleetNet - Internet on the Road",* Conference Proceedings ITS 2001, 8th World Congress on Intelligent Transportation Systems, Sydney, Australia, Oct. 2001.

[4] M. C. Gonzalez, C. A. Hidalgo, and A.-L. Barabási, *"Understanding individual human mobility patterns,"* Nature, vol. 453, no. 7196, pp. 779-782, June 2008.

[5] S. Jain, K. Fall, and R. Patra, *"Routing in a delay tolerant network,"* in SIGCOMM'04: Proceedings of the 2004 conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, New York, NY, USA: ACM, pp.145-158, 2004.

[6] R. Jathar, *"Probabilistic Routing Protocol using Contact Sequencing In Delay Tolerant Networks",* M.Tech thesis, Dept. of Computer Science & Engineering, Indian Institute of Technology – Kharagpur, May 2009.

[7] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. S. Peh, and D. Rubenstein, *"Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with Zebranet*," SIGOPS Oper. Syst. Rev., 36(5):96-107, 2002.

[8] M. Kim, D. Kotz, and S. Kim, *"Extracting a mobility model from real user traces,"* in INFOCOM 2006. 25th IEEE International Conference on Computer Communications Proceedings, pp.1-13, 2006.

[9] A. Lindgren, A. Doria, and O. Schelén, *"Probabilistic routing in intermittently connected networks,"* SIGMOBILE Mob. Comput. Commun. Rev., vol. 7, no. 3, pp. 19-20, 2003.

[10] A. S. Pentland, R. Fletcher, and A. Hasson, *"Daknet: Rethinking connectivity in developing nations,"* Computer, vol. 37, no. 1, pp. 78-83, 2004.

[11] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, *"Spray and wait: an efficient routing scheme for intermittently connected mobile networks,"* in WDTN '05: Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking, New York, NY,USA: ACM, pp. 252-259, 2005.

[12] A. Vahdat and D. Becker, *"Epidemic routing for partially connected ad hoc networks,"* Technical Report, Duke University, April 2000.