Identifying Contact Patterns in Intermittently Connected Networks

Thesis submitted in partial fulfillment of the
requirements for award of the degree of

Master of Technology
in
Information Technology

By

Yogesh Piolet T
[Roll No. 08IT6018]

Under the supervision of

Dr. Arobinda Gupta

School of Information Technology
Indian Institute of Technology, Kharagpur

May 2010

DECLARATION

I, Sri. **Yogesh Piolet T**, Roll no. **08IT6018** registered as a student of M.Tech. in the **School of Information Technology**, Indian Institute of Technology, Kharagpur, India (hereinafter referred to as the 'Institute') do hereby submit my thesis, title:

Identifying Contact Patterns in Intermittently Connected Networks.

(hereinafter referred to as 'my thesis') in a printed as well as in an electronic version for holding in the library of record of the Institute.

I hereby declare that:

1. The electronic version of my thesis submitted herewith on CDROM is in **PDF** format. (mention whether PostScript or PDF).
2. My thesis is my original work of which the copyright vests in me and my thesis does not infringe or violate the rights of anyone else.
3. The contents of the electronic version of my thesis submitted herewith are the same as that submitted as final hard copy of my thesis after my viva-voce and adjudication of my thesis on **03-05-2010**.
4. I agree to abide by the terms and conditions of the Institute Policy on Intellectual Property (hereinafter Policy) currently in effect, as approved by the competent authority of the Institute.
5. I agree to allow the Institute to make available the abstract of my thesis in both hard copy (printed) and electronic form.
6. For the Institute's own, non-commercial, academic use I grant to the Institute the non-exclusive license to make limited copies of my thesis in whole or in part and to loan such copies at the Institute's discretion to academic persons and bodies approved of from time to time by the Institute for non-commercial academic use. All usage under this clause will be governed by the relevant fair use provisions in the Policy and by the Indian Copyright Act in force at the time of submission of the thesis.
7. Furthermore *(strike out whichever is not applicable)*
   (a) I agree / ~~do not agree~~ to allow the Institute to place such copies of the electronic version of my thesis on the private Intranet maintained by the Institute for its own academic community.
   (b) I agree / ~~do not agree~~ to allow the Institute to publish such copies of the electronic version of my thesis on a public access website of the Internet should it so desire.

8. That in keeping with the said Policy of the Institute I agree to assign to the Institute (or its Designee/s) according to the following categories all rights in inventions, discoveries or rights of patent and/or similar property rights derived from my thesis where my thesis has been completed (tick whichever relevant):

   (a) with use of Institute-supported resources as defined by the Policy and revisions thereof,

   (b) with support, in part or whole, from a sponsored project or program, vide clause 6(m) of the Policy.
   I further recognize that:

   (c) All rights in intellectual property described in my thesis where my work does not qualify under sub-clauses 8(a) and/or 8(b) remain with me.

9. The Institute will evaluate my thesis under clause 6(b1) of the Policy. If intellectual property described in my thesis qualifies under clause 6(b1) (ii) as Institute-owned intellectual property, the Institute will proceed for commercialization of the property under clause 6(b4) of the Policy. I agree to maintain confidentiality as per clause 6(b4) of the Policy.

10. If the Institute does not wish to file a patent based on my thesis, and it is my opinion that my thesis describes patentable intellectual property to which I wish to restrict access, I agree to notify the Institute to that effect. In such a case no part of my thesis may be disclosed by the Institute to any person(s) without my written authorization for one year after the date of submission of the thesis or the period necessary for sealing the patent, whichever is earlier.

Yogesh Piolet T
(Name of Student)

(Name of Supervisor)
1. **Dr. Arobinda Gupta**
2.

(Signature of the Student)

Department/School/Centre : **School of Information Technology**

Signature of the Head of the Department/Centre/School

Indian
Institute of
Technology,
Kharagpur

School of Information Technology

Certificate

This is to certify that this thesis entitled "**Identifying Contact Patterns In Intermittently Connected Networks**" submitted by Yogesh Piolet T, in partial fulfillment for the award of the degree of Master of Technology, is a record of bonafide research work carried out by him under my supervision during the period 2009-2010.

In my opinion this work fulfills the requirements for which it has been submitted. To best of my knowledge, this thesis has not been submitted to any other university or institution for any degree or diploma.

Dr. Arobinda Gupta
School of Information Technology &
Department of Computer Science & Engineering
Indian Institute of Technology, Kharagpur.
Dated:

Acknowledgements

As the final days of my Masters course in *IIT Kharagpur* approach closer, I would like to acknowledge all those people who have helped me achieve this milestone. It brings me immense pleasure to express my deepest sense of gratitude to my thesis advisor **Prof. Arobinda Gupta** for his expert guidance and support *throughout* the project work, despite his personal obligations. Our weekly meetings which were filled with constructive criticism are memories that I will cherish forever. His innovative ideas and invaluable suggestions have helped me shape this thesis to the current form. I look at him with great respect for his profound knowledge and relentless pursuit for perfection. I consider it as an honour to have felt a subtle taste of fine research under his experienced supervision.

I would like to thank all my **Teachers** at the *School of Information Technology* who have helped me improve myself intellectually and on a personal level. My **Lab colleagues** and **Friends** at *IIT Kharagpur* have been a constant source of unconditional support.

Nothing would have been possible without the support of my family who have been backing me up throughout my life. I wish to convey my sincere thanks to my parents; **Thulasidharan Pillai** and **Radhamma** and my younger brother **Rajesh Piolet** who have always supported me in pursuing my higher education selflessly.

Finally, I thank the **Almighty** for bringing me to this silent college town, **Kharagpur**.

Yogesh Piolet T

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 the 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 has been conducted extensively. Once slices have been generated, it becomes easier to predict whether the contact history is repetitive or not. A distributed approach, where each node independently determines the period of repetition is also considered. However, some of these approaches have a significant benefit over the others in terms of time complexity and space, which are both critical to the DTN.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Most network protocols are designed assuming that the underlying network satisfies certain properties such as the existence of a path between a source and a destination, relatively small end-to-end round trip delay, and relatively small node and link failures. In some cases such as mobile ad hoc networks (MANETs), link formation and failures are caused due to mobility of the nodes. However, protocols designed for MANETs also assume the formation of an end-to-end path between two nodes in order for them to communicate. In practice, the assumption of an end-to-end path between two nodes may not hold for certain types of networks. In such networks, nodes may come in contact with each other intermittently, causing parts of an end-to-end path to be formed at different times, and a complete end-to-end path may not exist at any time. Communication in such networks are done by relaying messages from one node to another. Each node, on receiving a message, either passes it on to the next node, or holds the message if no connection is available. The message is passed on to an appropriate node when it comes in contact at a later time (Note that a forwarding protocol for conventional networks, even MANETs, would drop the packet if the next hop is not in contact). Thus, the messages, which would otherwise be dropped, are delivered at the cost of an increased delay. Such networks are commonly called as *Delay Tolerant Networks (DTNs)*. They are also sometimes referred as *Challenged Networks* or *Opportunistic Networks*.

1.1 Motivation

Routing becomes a problem in a DTN because the conventional routing protocols fail with the inherent characteristics of a 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 node comes into contact. The node will immediately forward the message when the next node comes into contact. Several routing protocols have been proposed for DTNs [1, 2, 3][5, 6, 10, 11, 12].

There exists several real-world scenarios where contacts occur repeatedly over a period of time, resulting in contact patterns [1, 2, 3][4, 8]. For instance, people may move from their residences to workplaces or other activity places such as schools at around the same time in the morning. They also may come into contact with each other at the cafeteria during lunch. They return back home at around the same time in the evening. Also, the buses/trains move along designated routes repeatedly throughout the day. Due to these specific movement patterns, many contacts would occur repeatedly. A person who goes to office at around the same time everyday, comes in contact with similar people in the bus or train. On reaching the office, the person would come in contact with another group of people at around the same time everyday. Thus, the contacts will occur repeatedly over days, following a pattern.

This notion of contact patterns can be used to predict when the next contact is going to occur between specific node pairs. Such a prediction can be used effectively in routing for DTNs. In fact, a routing protocol called PROCS [1, 2, 3][6] uses the repetitive patterns in the contact history between nodes to decide which node must be chosen as the next node. But, PROCS assumes that the repetitive patterns are provided by a knowledge oracle. Hence, it is useful to design algorithms to find if a given contact history between a set of nodes has periodic repetitions or not.

1.2 Contributions of this Thesis

The contributions of this thesis are listed below:

1. Two similarity metrics, *Euclidean distance based similarity measure* and *set similarity based similarity measure*, have been proposed to measure the similarity between two sets of contact patterns on a scale of 0 to 1. These similarity measures have been

thoroughly evaluated for robustness using simulated contact histories. We concluded that the *Euclidean distance based similarity measure* is more robust and less susceptible to irregularities in the contact times.

2. An algorithm, *Constant Time Duration Slicing Algorithm*, has been proposed to slice a non-partitioned contact history into meaningful partitions such that most of the contacts repeat across the partitions. Such a slicing, if found, can give us a period of repetition for all the contacts. The algorithm has been evaluated with experimentation, both with simulated and real-world contact histories. The simulated contact history took into account different possible scenarios that could occur during contact between nodes.

1.3 Organization of this Thesis

The remaining chapters of this thesis are organized as follows:

Chapter 2 discusses the work done earlier for identifying contact patterns in DTNs.

Chapter 3 describes the similarity metrics that have been proposed to measure the similarity between a pair of contact patterns on a scale of 0 to 1. It provides the formal specification of the contact history, the mathematical formulation of the two similarity measures, and the inferences made with the experimentation of these measures.

Chapter 4 proposes an algorithm to partition the non-partitioned contact history into meaningful partitions such that most of the contacts repeat across the partitions. The chapter provides the formal specification of the slicing problem, the proposed algorithm along with the experimental results that were conducted to evaluate the correctness of the algorithm.

Chapter 5 concludes the thesis and outlines some possible extensions of the present work.

Chapter 2

Related Work

Delay Tolerant Networks (DTNs) exhibits properties such as intermittent connectivity, long/variable delay, high error rates and asymmetric data rates. Fall [4] [2] proposed the architecture for DTN that allowed both DTN and other conventional networks 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 identified the three major 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 a tiered functionality of the underlying protocols and thirdly, DTN protocols transmit as little information as possible because bandwidth may not be cheap.

There exists a few practical implementations of DTNs. One of the DTNs is the *Interplanetary Internet* [4] [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* [12] 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. Thus, the asynchronous communication is achieved between

the source and the destination (villages or towns) through the intermediate nodes (vehicles). Another potentially commercial application is the project *FleetNet* [4] [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 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 application of the DTN is the *ZebraNet* [4] [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 and Becker [4] [14] 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. Initially, they exchange an index of the messages called *summary vector*. Then, the node gives the other node a copy of all the messages that the other node does not have. Thus, this protocol creates too many replicas of the same message in the network, resulting in the increase of network congestion. The protocol has a high message delivery probability and a low average delivery time. However, it has poor buffer utilization and its performance falls rapidly with decreasing buffer size.

Several variants of the *Epidemic protocol* exist Fall:delay-tolerantnetworkarchitecture [13,11]. In Fall:delay-tolerantnetworkarchitecture [11], immunity based information is disseminated in the reverse direction once the message is delivered to its destination. The information of already delivered messages prevents any further exchange of those messages. Also, the existing copies of those messages would be dropped. This approach improves the network and buffer utilization. In the *Spray and Wait* protocol proposed by Spyropoulos et al. [4] [13], 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. 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. Jain et al. [4] [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.

Jathar et al. [4] [6] proposed the *Probabilistic strict ROuting using Contact Sequencing*

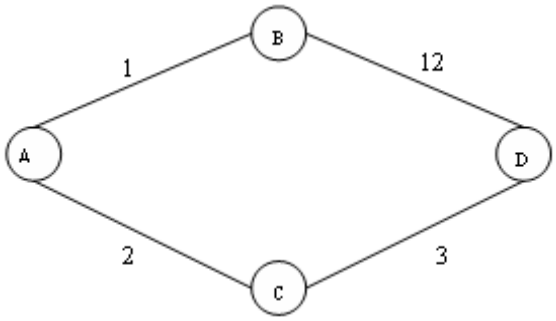

Figure 2.1: A Contact Graph

(PROCS). It 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:

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. 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. 2.1).

Lindgren et al. [11] 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 if the latter has higher delivery predictability to the destination node. This method replicates a message in a limited manner.

Leguay et al. [10] have proposed an approach where routing decisions are made using the concept of a *Euclidean* virtual space called *Mobyspace*. Each node forwards the message towards the nodes that have mobility patterns that are more similar to that of the destination.

It has been found that patterns have been observed from the movement of mobile nodes in many real life scenarios. Considering human beings as the mobile nodes, Gonzalez et al. [5]

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. [9] 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.

Chapter 3

Similarity Measures for Contact Patterns

This chapter describes two measures which help to find similarity between two contact patterns on a scale of 0 to 1. Firstly, the structure of a contact pattern is described. The possible operations by which a contact pattern can be made similar or dissimilar to another contact pattern are discussed. Also, some of the properties that any similarity measure for contact patterns must satisfy are specified. The approach by which the similarity measures namely, *Euclidean distance based similarity measure* and *set similarity measure* try to find the similarity between the contact patterns is described along with its mathematical formulation. We further describe the experimental setup that was used to evaluate the correctness and robustness of the proposed measures. The results and the inferences concluded from the simulations are described in detail in the last section of this chapter. It must be noted that the conclusions derived on the proposed similarity measures from this chapter are used to find periodicity in the contact patterns, as described in Chapter 4.

3.1 Formal Specification of Contact Pattern

The contact pattern is a 3-tuple list $\langle src, des, t \rangle$ indicating that a mobile node *src* meets another mobile node *des* at time *t*. Two contact patterns X and Y that are to be compared can be represented as shown in the 3.1.

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 contact

Table 3.1: Structure of contact patterns X and Y

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_β

patterns X and Y respectively. t_{min} and t_{max} are the minimum and maximum time of both the contact patterns. Frequency of meeting F_i and G_i are the number of times the corresponding node pairs ⟨src,des⟩ occur in the contact pattern X or Y respectively. m and n are the number of contacts in contact patterns X and Y respectively. γ and β are the number of unique node pairs in contact in contact patterns X and Y respectively.

3.2 Formulation of Similarity Measures

Given two contact patterns X and Y, three possible cases have been identified by which the contact patterns can be made similar or dissimilar. For every meeting $\langle \text{A,B}, t\rangle$ in X, the following are the three cases that are possible in Y correspondingly

1. $\langle \text{A,B}, t\rangle$ exists in Y too, indicating that A met B at time t in contact pattern Y. This case makes X and Y more similar to each other.

2. $\langle \text{A,B}, m\rangle$ exists in Y, such that m t. This indicates that A did not meet B at t, instead A met B at another time m. This case makes X and Y less similar to each other. However, the magnitude of similarity caused by this case is less than Case 1 and more than Case 3.

3. $\langle \text{A,B}, n\rangle$ does not exist in Y for any n, indicating that A did not meet B at all in contact pattern Y. This case makes X and Y more dissimilar to each other, and the magnitude of dissimilarity caused by this case is more than Case 2.

Given two contact patterns, the similarity measure η [0, 1] is a metric that is used to describe how closely the contact patterns match with each other. The closeness may take into account the factors 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 η must satisfy the following properties

1. For any contact pattern X, $\eta(X,X) = 1$ and $\eta(X,\phi) = 0$
2. *Commutative property*: For any two contact patterns X,Y, $\eta(X,Y) = \eta(Y,X)$
3. *Learning (Preservation of similarity) property*: A non-zero η for any two contact patterns X and Y can never be decreased to zero by any number of insertions and deletions of contacts in either of X and Y, unless all contacts with equal node pair in both contact patterns are deleted.

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

3.2.1 Euclidean Distance Based Similarity Measure

The *Euclidean distance based similarity measure* takes two factors into account. Firstly, it compares the frequency with which the contact between the same pair of nodes is occurring in both the contact patterns. This is done to differentiate between the node pairs that are meeting in both the contact patterns to the ones that are not. Also, if the node pairs are meeting in both the contact patterns, the difference in their number of meetings in both of them is considered. For instance, A meets B five times in the contact pattern X, while A meets B only one time in the contact pattern Y. Secondly, it takes into account as to how close are the contacts between corresponding node pairs in both the contact patterns occurring in terms of time (temporally). This is done to compare the difference in time by which the same pair of nodes meets in both the contact patterns. For instance, A meets B at 5 PM in contact pattern X, while A meets B at 10 PM in contact pattern Y. 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 contact patterns 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

-

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

In the Equation 3.1 , n is the number of node pairs that occur in both the contact patterns X and Y. (X_i-Y_j) is the time difference in the contact of the corresponding node pair such that the corresponding node pair occur in both the contact patterns X and Y.

3.2.2 Set Similarity Measure

The similarity measure η_2 is calculated on a 3-tuple equality basis $\langle$src,des,$time\rangle$ unlike in η_1, where frequency is calculated over equality of 2-tuples $\langle$src,des$\rangle$. Hence, frequency tuple becomes redundant in the calculation of η_2. The similarity measure η_2 imposes a strict restriction upon the node pairs occurring in both the contact patterns to meet each other at exactly the same time to show an increase in similarity. On the other hand, η_1 allows temporal variations in the contact of same node pairs in both the contact patterns. Considering contact patterns X and Y as multi-sets, the *set similarity measure* is formulated as shown in the Equation 3.2.

$$\eta_2 = \frac{X \cup Y}{X + Y - X \cap Y} \tag{3.2}$$

3.3 Evaluation of Similarity Measures

The two measures were used to measure similarity between different types of contact patterns. The contact patterns 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 contact pattern generator. The contact pattern 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 contact pattern X was generated by the fore-mentioned contact pattern generator, the contact pattern Y can be generated by manipulating contact pattern X, treating X as a base pattern. Each contact in contact pattern X can be manipulated by any of the following four operations. *Retain contact* is an operation where the contact in contact pattern X is placed into contact pattern Y without any change. For instance, if $\langle$A,B, 10$\rangle$ is present in X, A meets B at 10 in contact pattern Y also. *Dissimilar contact* is an operation by which the contact in contact pattern X is placed into contact pattern Y by retaining the same node pair and inducing a temporal dissimilarity/variation in meeting time. For instance, if $\langle$A,B, 10$\rangle$ is present in X, A meets B at 12 in contact pattern Y. *Drop contact* is an operation by which the contact in contact pattern X is not placed into contact pattern Y. For instance, if $\langle$A,B, 10$\rangle$ is present in X, A does not meet B at 10 in Y. After generating the contact pattern Y by manipulating the contact pattern X by parameterized percentages of each of the three operations, a parameterized percentage of *new contacts* are added into contact pattern Y. The new contacts are generated randomly and may or may not be present in contact pattern Y. 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 contact patterns were generated fell in the range [0, 23].

3.4 Results and Inferences

The similarity measures that were proposed were evaluated for correctness and robustness on the simulated contact patterns. In the following sub-sections, the procedure carried out in conducting each of the experiment is explained in detail. Later, an analysis is made on the results obtained from the experiments. The inferences concluded are mentioned.

3.4.1 Observation of the rising and falling rate of similarity measures

In this experiment, the contact pattern X was generated and contact pattern Y was kept empty initially. After the addition of each contact into contact pattern Y by retaining contacts from contact pattern X, the similarity measures were calculated to observe the rising rate. When the contact pattern X became exactly same as that of contact pattern

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 with the addition of same contacts into contact pattern Y and decrease non-uniformly to the addition of new contacts. It must be noted that η_1 and η_2 can never be reduced to 0 irrespective of the number of contacts added. Thus, the similarity measure has the property of learning (preserving) the similarity previously achieved in a chronological time frame.

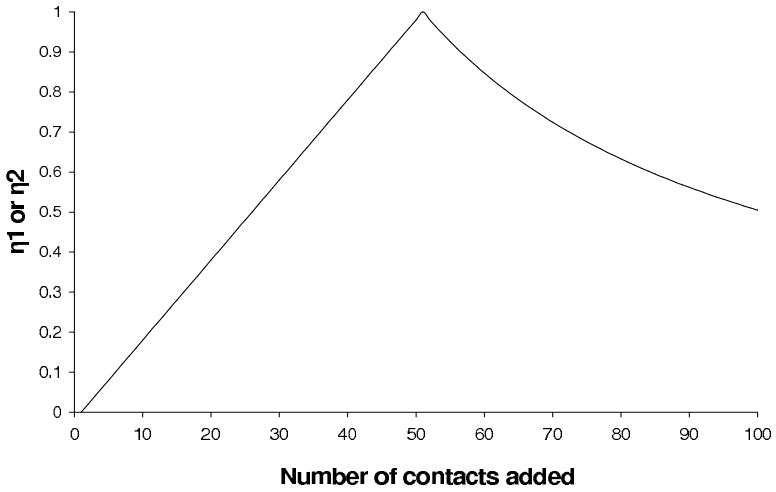

Figure 3.1: Variation in η_1 and η_2 for same contacts followed by new contacts

3.4.2 Observation of the effect of temporal variations on similarity measures

In this experiment, the contact pattern X was generated and contact pattern Y was kept empty initially. After the addition of each contact into contact pattern Y by retaining node pairs and inducing temporal dissimilarity of varying degrees (say 0, 4, 8 and so on) from contact pattern X, the similarity measures were calculated to observe the rising rate. The results are shown in Figure 3.2 and 3.3 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 contact patterns. 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.

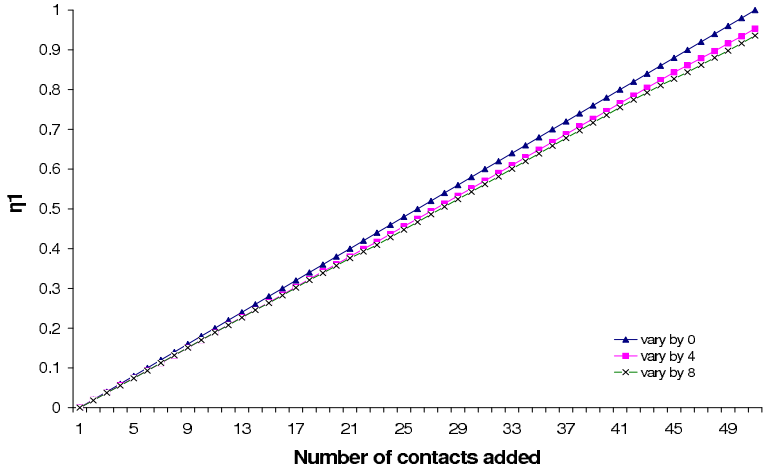

Figure 3.2: Adding dissimilar contacts - Effect on η_1

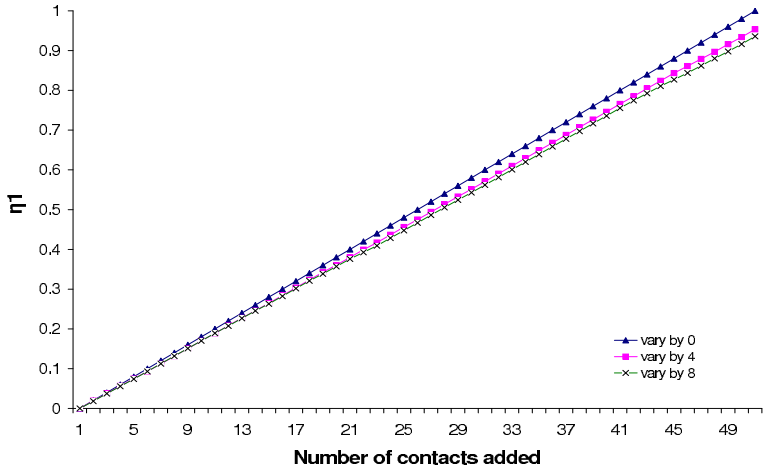

Figure 3.3: Adding dissimilar contacts - Effect on η_2

3.4.3 Observation of the falling rate of similarity measures

In this experiment, the contact pattern X was generated and contact pattern Y was kept same as contact pattern X initially. After the addition of each contact into contact pattern Y by any operation (same, new or temporal variation), the similarity measures were calculated to observe the falling rate. The results are shown in Figure 3.4. From the results, it can be noted that though η_1 and η_2 are not falling uniformly, they do follow a pattern. For every ith contact that is added into an initially equal contact pattern Y with cardinality —Y—,

the similarity measure falls to . From the results, we infer that both η_1 and η_2 decrease harmonically when any contact is added to either of the already equal contact patterns.

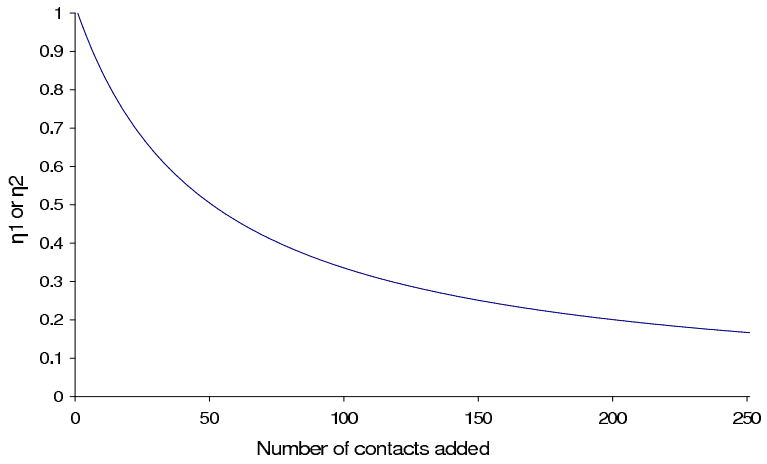

Figure 3.4: Variation in η_1 and η_2 for addition of new contacts to equal contact patterns

Chapter 4

An Approach to Finding Periodicity

In the previous chapter, we discussed the robustness of the proposed similarity measures. We concluded that *Euclidean distance based similarity measure* is more tolerant towards irregularity in contact patterns than set similarity measure. In this chapter, we used the *Euclidean distance based similarity measure* to find periodic repetition of contacts in the contact pattern. The problem addressed in this chapter is to determine the period of repetition in the contact pattern of a given set of nodes or the lack of it. In this chapter, we formally define the slicing problem. We propose an algorithm, *Constant Time Duration Slicing Algorithm* to solve the slicing problem. We describe the experimental setup that was used to evaluate the correctness of the proposed algorithm. The results and the inferences concluded from the simulations are described in detail in the last section of this chapter.

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. Let t be the maximum time of all the contacts that occurred in the contact pattern. Since a contact pattern can be partitioned at any time duration between 0 to $\lceil \frac{t}{2} \rceil$, it becomes necessary to define a slice measure which would determine which time duration yielded the best slicing. Consider a contact pattern P that is sliced into n slices. We compare each pair of the n slices using the *Euclidean distance based similarity measure* $\eta 1$. A threshold based

slice measure is defined as follows -

$$\phi(D) = \frac{x}{{}^{n}C_2}$$

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 β. For example, if 80 out of 100 similarity measure values are above the pre-specified threshold β 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 β. 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.

Definition: *Given a 3-tuple pre-processed contact history of the form ¡src, des, time¿ from time 0 to time t, the aim of the pattern slicing problem is to find the smallest duration SP (Slice Point), 1 SP 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 i t+1 and i SP. On equal slicing, α is the pre-specified threshold percent of similarity measure values of the slice pairs which must be above the pre-specified threshold β.*

4.2 *Constant Time Duration Slicing Algorithm*

We propose a brute force based solution to solve the slicing problem. In this approach, the contact pattern is sliced into several equal-duration partitions (slices). A pair-wise comparison of the contacts that occurred in these partitions is made to determine the similarity amongst the partitions using the *Euclidean distance based similarity measure*. Thus, if there are n partitions, the number of comparison values would be ${}^{n}C_2$. Once we determine the comparison values of all the slice-pairs, we separate the slice pairs into matching pairs and non-matching pairs. The matching slice-pairs are the ones whose similarity measure is equal to or greater than a given threshold β. The non-matching pairs have a similarity measure below β. The above steps are performed for each duration from 1 to $\lceil \frac{t}{2} \rceil$. The algorithm declares that the contact history is non-repetitive if the number of matching slice-pairs is below a pre-specified threshold α percentage of the total number of slice-pairs. Otherwise, it

returns the slice point as the smallest time duration on which slicing resulted in the number of matching slice-pairs above or equal to the pre-specified threshold α percentage of the total number of slice-pairs, thus declaring the contact pattern to be repetitive.

Algorithm 4.2.1 shows the pseudo-code of the *Constant Time Duration Slicing Algorithm*. Let t be the maximum time in the contact pattern which is sorted in the order of time. $DURATION$ is the variable used to represent the time period over which the slicing is performed in each iteration of the algorithm. The slicing of the contact pattern is performed repeatedly from $DURATION = 1$ to $\lceil \frac{t}{2} \rceil$. However, if the contact pattern is found to be perfectly repeating for any of the $DURATION$ values, the algorithm exits immediately. In line 2, the contact pattern is sliced using the corresponding $DURATION$ value of that iteration. In lines 3–7, the starting times in each of the slices is made to offset from 0. It must be noted that prior to slicing, the times in the contact pattern was monotonically varying from 0 to t. This results in n number of equally partitioned slices and the times in each of these slices is offset from 0. In lines 8–12, the slices are compared with each other using the *Euclidean distance based similarity measure*. Thus, the $comparison_matrix$ is created with one entry for each slice-pair. The entries in the $comparison_matrix$ must be in the range [0,1]. Let X be the number of entries above the pre-specified threshold β. In line 14, we compute the threshold measure for the $DURATION$ value of the corresponding iteration. In lines 15–19, we check if the contact pattern is perfectly repeating. If it is, the algorithm exits with $DURATION$ as the slice point. In lines 21–23, the slice point for the non-perfectly repeating contact pattern is found by choosing the smallest $DURATION$ which has a threshold slice measure above or equal to α.

4.3 Experimental Results

In order to evaluate the feasibility and correctness of the proposed algorithm, we conducted extensive experiments with the following setup: Using a pattern that is generated by the procedure mentioned in section **??** as a base, we generated a perfectly repetitive contact pattern by 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 a contact pattern which is not perfectly repetitive, we used 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 node pairs which are already meeting

Algorithm 4.2.1 *Constant Time Duration Slicing Algorithm*: $CTDSlicing()$

INPUT: A contact pattern of the form ⟨src,des,time⟩ which is time sorted.
var $DURATION$
var i, j // counters for increments
var X // Number of slice comparisons above β
var list $slice\_measure$ // An array of $\phi(D)$ values for each $DURATION$
var $slice\_point$ // slice point of the contact history
var $slice\_point\_found\_flag$ // To indicate whether the slice point has been found yet

$slice\_point\_found\_flag \leftarrow FALSE$

for $DURATION = 1$ to $\lceil \frac{t}{2} \rceil$ **do**
  Slice the contact pattern for every $DURATION$ units.
  for each slice **do**
    for each contact entry **do**
      $time \leftarrow time\%(DURATION + 1)$
    end for
  end for

  for each slice i **do**
    for every slice j **do**
      $comparison\_matrix$[i,j] $\leftarrow \eta_1(slice$ i, $slice$ j$)$
    end for
  end for

  $X \leftarrow$ Number of entries in $comparison\_matrix$ above pre-specified threshold β
  $slice\_measure$[DURATION] $\leftarrow \frac{X}{\lceil \frac{t+1}{DURATION} \rceil C_2}$
  if $slice\_measure[DURATION] = 1$ **then**
    $slice\_point \leftarrow DURATION$
    $slice\_point\_found\_flag \leftarrow TRUE$
    break
  end if
end for

if $slice\_point\_found\_flag = FALSE$ **then**
  $slice\_point \leftarrow$ smallest DURATION i with $slice\_measure$ value $\geq \alpha$
end if

in a slice are referred to as *regularly meeting node pairs*. All the nodes involved in the base pattern are called *participating 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 node pairs other than the regularly meeting node pairs also meet in a specific slice. This is referred to as new unique node pair scenario. For instance, the participating node A meets another non-participating node C. Another instance, A and B, both of which are participating nodes but are not a regularly meeting node pair meet each other for the first time. In the second scenario, we add a fixed percentage of the same regularly meeting node pairs meeting at times other than those present in the slice. This simulates the scene where the regularly meeting nodes meet the same node again at a different time in addition to their existing meeting. This is referred to as new meeting scenario. For instance, regularly meeting node pair, A and B, meet again at 5 PM. 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. This is referred to as time dissimilarity scenario. For instance, regularly meeting node pair, A and B do not meet at their regular time 5 PM, instead they meet at 10 PM. We have tested the proposed algorithm for correctness and feasibility on such non-perfectly repetitive contact pattern.

4.3.1 The New Unique Node pair Scenario

In the following set of experiments on the new unique node pair scenario, the base pattern was generated by the procedure mentioned in section **??**. The base pattern simulated 400 meetings between 10 nodes, of which only 25 node pairs met each other over the time period from 0 to 50. This base pattern was then repeated for 20 cycles by inducing the changes in each cycle so as to simulate the new unique node pair scenario. Thus, the contact pattern was generated and the slice point was set to 51. To simulate the new unique node pair scenario, we increased the number of node pairs from 25 in steps of 10% of the total number of possible non-regularly meeting node pairs. The changes were induced in all the 20 cycles. Note that, the constant time duration algorithm can declare the contact pattern generated to be repetitive by specifying the slice point to be anything other than 1020 (the time the last meeting occurred in the entire contact pattern, thus making the entire contact pattern as one slice). However, the proposed algorithm is assumed to have successfully found the contact pattern to be repetitive only if the slice point is found to be in the close range of 51. The failure in finding a slice point and its subsequent declaration as a non-repetitive contact

pattern by the algorithm is depicted in the results by the slice point 1020.

4.3.1.1 Observation of the effect of variation caused by β with fixed α

In order to observe the variation caused by β on the contact pattern where new node pairs were added continually, an experiment was conducted. The α was fixed to 0.8, implying that only if 80% of the slice comparisons were above the specified β value would the contact pattern be considered to be repetitive. In order to observe the effect of the variation caused by β on this scenario, the β value is varied from 0.8 to 1.0 in steps of 0.05. The slice point is then calculated for each combination of β and the percentage of change induced by the new unique node pair scenario. The graph obtained is as shown in Figure 4.2.

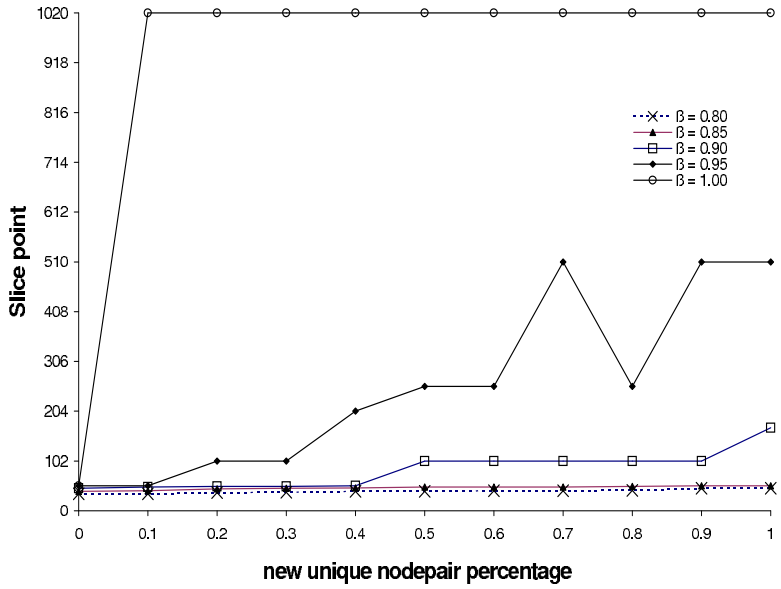

Figure 4.1: Effect of β on slice point in new unique node pair scenario

From the results, we observe that the *Constant Time Duration Slicing Algorithm* declared the contact pattern to be repetitive for 100% of the changes and found that the slice point was in a close range to that of 51 for the β values 0.80 and 0.85. However, as the β values increased to 0.90 and above, the slice point found by the algorithm was in a close range of 51 only for upto 40% change in 0.90, 10% change in 0.95 and 0% change in 1.00. From the results, we infer that the maximum percentage of change that can still result in identification of the slice point is largely dependent on the value of β. As β value increases, the *Constant Time Duration Slicing Algorithm* declares the contact pattern to be non-repetitive even for small changes induced by this scenario. When β value becomes unity, the proposed

algorithm declared the contact pattern to be non-repetitive even for 10% changes induced by this scenario.

4.3.1.2 Observation of the effect of variation caused by α with fixed β

In order to observe the variation caused by α on the contact pattern where new node pairs were added continually, an experiment was conducted. The β was fixed to 0.975, implying that only if 97.5% of the slice comparisons were above the specified α value would the contact pattern be considered to be repetitive. In order to observe the effect of the variation caused by α on this scenario, the α value is varied from 0.6 to 1.0 in steps of 0.1. The slice point is then calculated for each combination of α and the percentage of change induced by the new unique node pair scenario. The graph obtained is as shown in Figure **??**. From the results, we observe that the *Constant Time Duration Slicing Algorithm* declared the contact pattern to be repetitive for 10% of the changes and found that the slice point was in a close range to that of 51 for the α values 0.60 and 0.7. As the α values are increased to 0.80 and above, the slice point was found to be not close to 51 even for small changes. From the results, we infer that as the α value increases, the *Constant Time Duration Slicing Algorithm* declares the contact pattern to be non-repetitive even for small changes induced by this scenario.

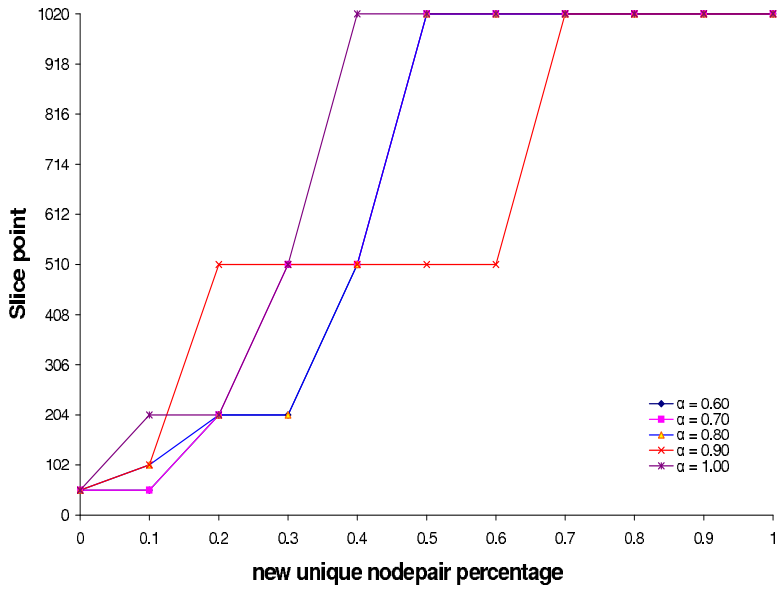

Figure 4.2: Effect of α on slice point in new unique node pair scenario

4.3.1.3 Observation of the effect of variation caused by α and β

In order to observe the effect of variation caused by α and β on the new unique node pair scenario, a new parameter is defined. Tolerance level is defined as the maximum percentage of changes induced by the new unique node pair scenario that still resulted in detecting a slice point in the range 51+10% or 51-10% of the slice point. In this experiment, the base pattern was generated by the procedure mentioned in section **??**. The base pattern simulated 400 meetings between 25 nodes, of which only 125 node pairs met each other over the time period from 0 to 50. This base pattern was then repeated for 20 cycles by inducing the changes in each cycle so as to simulate the new unique node pair scenario. Thus, the contact pattern was generated and the slice point was set to 51. To simulate the new unique node pair scenario, we increased the number of node pairs from 125 in steps of 10% of the total number of possible non-regularly meeting node pairs. The changes were induced in all the 20 cycles. In order to observe the effect of the variation caused by α and β on the tolerance level, the α value is varied from 0.75 to 1.0 in steps of 0.05 and the β value is varied from 0.80 to 1.0 in steps of 0.05. The slice point is then calculated for each combination of α and β for each percentage of change induced by the new unique node pair scenario from 0.0 to 1.0 in steps of 0.1 until the slice point obtained goes out of range of the tolerance level. The result obtained is as shown in Figure 4.3.1.3. From the results, we observe that the tolerance level of the constant time duration algorithm decreases as the β value increases. The tolerance level is also independent of α. From the results, we infer that the constant time duration algorithm can tolerate a minimum of 10% of the changes caused by the new unique node pair scenario, by still being able to detect the slice point in the range close to that of the actual slice point.

Table 4.1: Effect of α and β on slice point in new unique nodepair scenario

α \ β	**0.80**	**0.85**	**0.90**	**0.95**	**1.00**
0.75	0.3	0.2	0.2	0.1	0.1
0.80	0.3	0.2	0.1	0.1	0.1
0.85	0.3	0.2	0.2	0.1	0.1
0.90	0.3	0.2	0.1	0.1	0.1
0.95	0.3	0.2	0.1	0.1	0.1
1.00	0.3	0.2	0.1	0.1	0.1

4.4 The New Meeting Scenario

In the following set of experiments on the new meeting scenario, the base pattern was generated by the procedure mentioned in Section **??**. The base pattern simulated 400 meetings between 10 nodes, of which only 25 node pairs met each other over the time period from 0 to 50. This base pattern was then repeated for 20 cycles by inducing the changes in each cycle so as to simulate the new meeting scenario. Thus, the contact pattern was generated and the slice point was set to 51. To simulate the new meeting scenario, we increased the number of meetings among the regularly meeting node pairs from 400 in steps of 10% of the total number of meetings in the base pattern. The changes were induced in all the 20 cycles. As in the earlier scenario, the failure in finding a slice point and its declaration as a non-repetitive contact pattern is depicted in the results by the slice point 1020.

4.4.0.4 Observation of the effect of variation caused by α and β

In order to observe the variation caused by β on the contact pattern where new meetings among regularly meeting node pairs were added continually, an experiment was conducted. The α was fixed to 0.8. In order to observe the effect of the variation caused by β on this scenario, the β value is varied from 0.8 to 1.0 in steps of 0.05. The slice point is then calculated for each combination of β and the percentage of change induced by the new meeting scenario. The graph obtained is as shown in Figure 4.3. From the results, we observe that the *Constant Time Duration Slicing Algorithm* declared the contact pattern to be repetitive for 100% of the changes and found that the slice point was in a close range to that of 51 for the β values 0.80 and 0.85. As the β values increased to 0.90 and above, the slice point was found to be in a close range of 51 only for upto 10% change for β value 0.90. For β values 0.95 and 1.0, the proposed algorithm declared that the pattern was non-repetitive even for 10% change caused by this scenario. From the results, we infer that as the β value increases, the *Constant Time Duration Slicing Algorithm* finds it harder to declare the contact pattern to be repetitive even for small changes induced by this scenario. When β value becomes unity, the proposed algorithm declared the contact pattern to be non-repetitive even for 10% changes induced by this scenario.

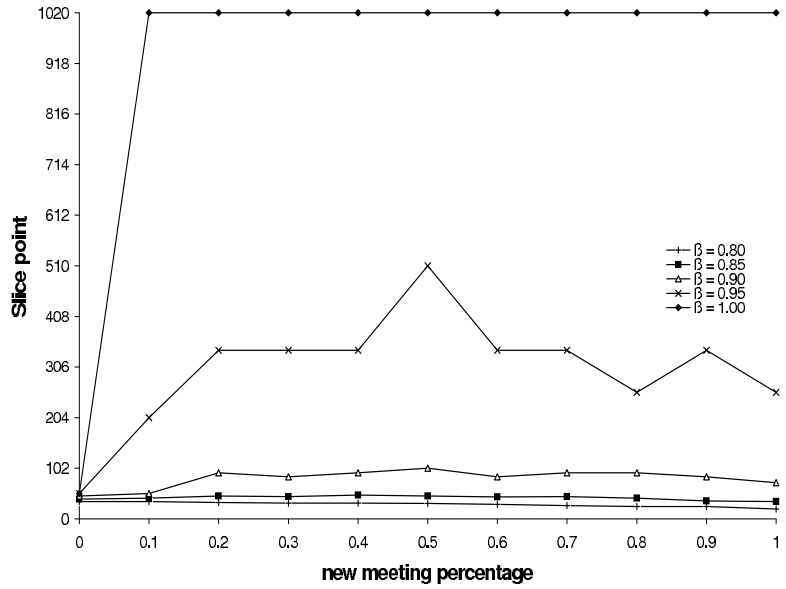

Figure 4.3: Effect of β on slice point in new meeting scenario

4.4.0.5 Observation of the effect of variation caused by α with fixed β

In order to observe the variation caused by α on the contact pattern where new meetings among regularly meeting node pairs were added continually, an experiment was conducted. The β was fixed to 0.975. In order to observe the effect of the variation caused by α on this scenario, the α value is varied from 0.6 to 1.0 in steps of 0.1. The slice point is then calculated for each combination of α and the percentage of change induced by the new unique node pair scenario. The graph obtained is shown in Figure 4.4. From the results, we observe that the *Constant Time Duration Slicing Algorithm* declared the contact pattern to be non-repetitive for even 10% of the changes for α values from 0.6 to 1.0. Higher the α value, faster the algorithm loses its tolerance to changes induced by the new meeting scenario. From the results, we infer that as the α value increases, the *Constant Time Duration Slicing Algorithm* finds it harder to declare the contact pattern to be repetitive even for small changes induced by this scenario.

4.4.0.6 Observation of the effect of variation caused by α and β

In order to observe the variation caused by α and β on the new meeting scenario, a new parameter is defined. Tolerance level is defined as the maximum percentage of changes induced by the new meeting scenario that still resulted in detecting a slice point in the range 51+10% or 51-10% of the slice point. In this experiment, the base pattern was generated by

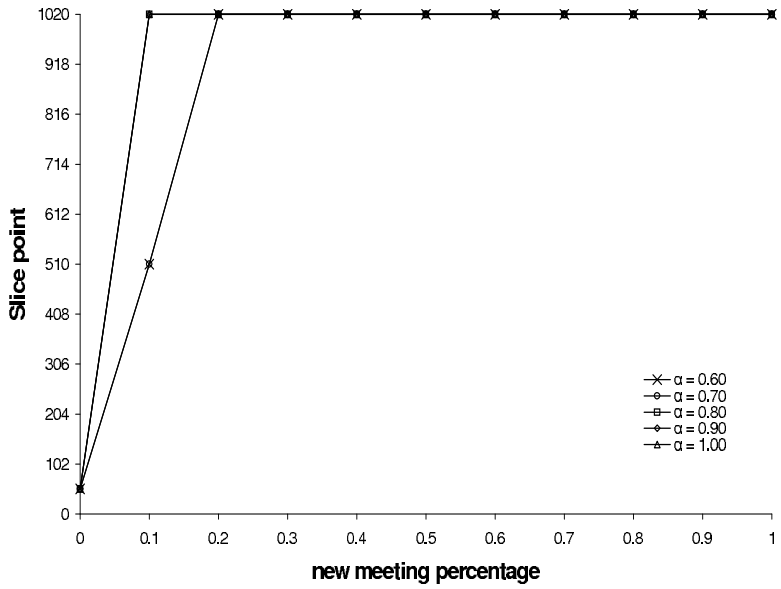

Figure 4.4: Effect of α on slice point in new meeting scenario

the procedure mentioned in Section **??**. The base pattern simulated 400 meetings between 25 nodes, of which only 125 node pairs met each other over the time period from 0 to 50. This base pattern was then repeated for 20 cycles by inducing the changes in each cycle so as to simulate the new meeting scenario. Thus, the contact pattern was generated and the slice point was set to 51. To simulate the new meeting scenario, we increased the number of meetings among the regularly meeting node pairs from 400 in steps of 10% of the total number of meetings in the base pattern. The changes were induced in all the 20 cycles. In order to observe the effect of the variation caused by α and β on the tolerance level, the α value is varied from 0.75 to 1.0 in steps of 0.05 and the β value is varied from 0.80 to 1.0 in steps of 0.05 . The slice point is then calculated for each combination of α and β for each percentage of change induced by the new meeting scenario from 0.0 to 1.0 in steps of 0.1 until the slice point obtained goes out of range of the tolerance level. The result obtained is as shown in Figure 4.4.0.6. From the results, we observe that the tolerance level of the constant time duration algorithm decreases as the β value increases. The tolerance level is also independent of α. From the results, we infer that the constant time duration algorithm can tolerate a minimum of 10% of the changes caused by the new meeting scenario.

Table 4.2: Effect of α and β on slice point in new meeting scenario

α \ β	**0.80**	**0.85**	**0.90**	**0.95**	**1.00**
0.75	0.2	0.1	0.1	0.1	0.1
0.80	0.2	0.1	0.1	0.1	0.1
0.85	0.2	0.1	0.1	0.1	0.1
0.90	0.2	0.1	0.1	0.1	0.1
0.95	0.2	0.1	0.1	0.1	0.1
1.00	0.2	0.1	0.1	0.1	0.1

4.5 The Time Dissimilarity Scenario

In the following set of experiments on the time dissimilarity scenario, the base pattern was generated by the procedure mentioned in Section **??**. The base pattern simulated 400 meetings between 10 nodes, of which only 25 node pairs met each other over the time period from 0 to 50. This base pattern was then repeated for 20 cycles by inducing the changes in each cycle so as to simulate the time dissimilarity scenario. Thus, the contact pattern was generated and the slice point was set to 51. To simulate the time dissimilarity scenario, we changed the meeting times of the contact occurring between participating nodes in each slice. This dissimilarity in time was induced from 0% to 100% of the meetings in the contact pattern in steps of 10% of the total number of meetings in contact pattern. The changes were induced in all the 20 cycles. As in the earlier scenarios, the failure in finding a slice point and its subsequent declaration as a non-repetitive contact pattern is depicted in the results by the slice point 1020.

4.5.0.7 Observation of the effect of variation caused by α and β

In order to observe the variation caused by β on the contact pattern where the meeting times between the participating node pairs were changed continually, an experiment was conducted. The α was fixed to 0.8. In order to observe the effect of the variation caused by β on this scenario, the β value is varied from 0.8 to 1.0 in steps of 0.05. The slice point is then calculated for each combination of β and the percentage of change induced by the time dissimilarity scenario. The graph obtained is as shown in Figure 4.5. From the results, we observe that the *Constant Time Duration Slicing Algorithm* declared the contact pattern to be repetitive for 100% of the changes and found that the slice point was in a close range to that of 51 for the β values 0.80, 0.85, 0.90 and 0.95. For β value 1.0, the proposed algorithm

declared that the pattern was non-repetitive even for 10% change caused by this scenario. From the results, we infer that when β value becomes unity, the proposed algorithm declared the contact pattern to be non-repetitive even for 10% changes induced by this scenario.

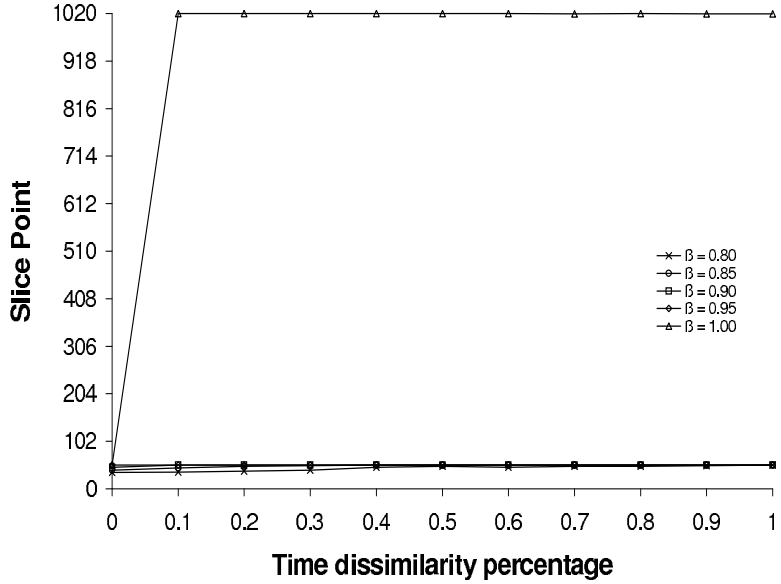

Figure 4.5: Effect of β on slice point in time dissimilarity scenario

4.5.0.8 Observation of the effect of variation caused by α with fixed β

In order to observe the variation caused by α on the contact pattern where the meeting times between the participating node pairs were changed continually, an experiment was conducted. The β was fixed to 0.975. In order to observe the effect of the variation caused by α on this scenario, the α value is varied from 0.6 to 1.0 in steps of 0.1. The slice point is then calculated for each combination of α and the percentage of change induced by the new unique node pair scenario. The graph obtained is as shown in Figure 4.6. From the results, we observe that the constant time duration algorithm found a slice point in a close range of 51 for upto 50% for $\alpha = 0.60$, upto 40% for $\alpha = 0.70$ and 0.80, upto 30% for $\alpha = 0.90$ and upto 20% for $\alpha = 1.00$. From the results, we infer that as the α value increases, the slice point found by the *Constant Time Duration Slicing Algorithm* is in a closer range to 51 lesser and lesser as the changes are induced following the time dissimilarity scenario for higher α values.

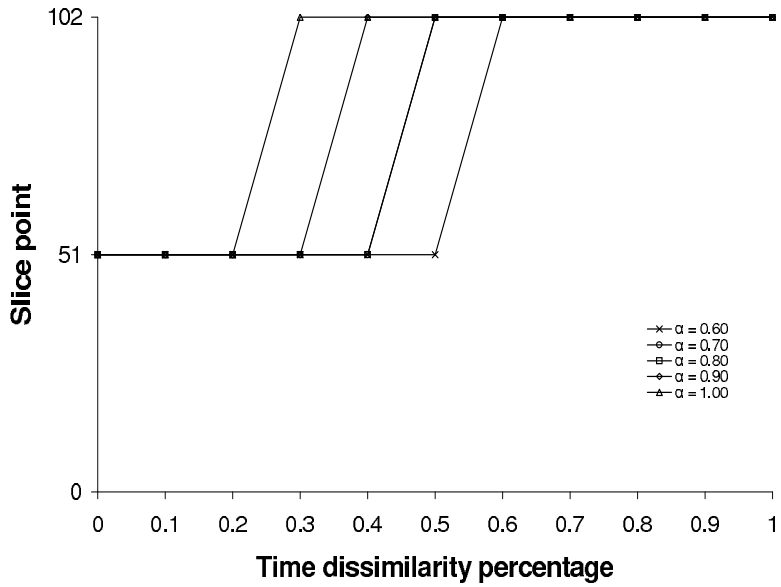

Figure 4.6: Effect of α on slice point in time dissimilarity scenario

4.5.0.9 Observation of the effect of variation caused by α and β

In order to observe the variation caused by α and β on the time dissimilarity scenario, a new parameter is defined. Tolerance level is defined as the maximum percentage of changes induced by the time dissimilarity scenario that still resulted in detecting a slice point in the range 51+10% or 51-10% of the slice point. In this experiment, the base pattern was generated by the procedure mentioned in Section **??**. The base pattern simulated 400 meetings between 25 nodes, of which only 125 node pairs met each other over the time period from 0 to 50. This base pattern was then repeated for 20 cycles by inducing the changes in each cycle so as to simulate the time dissimilarity scenario. Thus, the contact pattern was generated and the slice point was set to 51. To simulate the time dissimilarity scenario, we changed the meeting times of the contact occurring between participating nodes in each slice. This dissimilarity in time was induced from 0% to 100% of the meetings in the contact pattern in steps of 10% of the total number of meetings in contact pattern. The changes were induced in all the 20 cycles. In order to observe the effect of the variation caused by α and β on the tolerance level, the α value is varied from 0.75 to 1.0 in steps of 0.05 and the β value is varied from 0.80 to 1.0 in steps of 0.05. The slice point is then calculated for each combination of α and β for each percentage of change induced by the new unique node pair scenario from 0.0 to 1.0 in steps of 0.1 until the slice point obtained goes out of range of the tolerance level. The result obtained is as shown in Figure 4.5.0.9. From the results, we observe that the tolerance level of the constant time duration algorithm decreases as the

β value increases. The tolerance level is also independent of α. From the results, we infer that the *Constant Time Duration Slicing Algorithm* can tolerate a minimum of 10% of the changes caused by the time dissimilarity scenario.

Table 4.3: Effect of α and β on slice point in time dissimilarity scenario

α \ β	**0.80**	**0.85**	**0.90**	**0.95**	**1.00**
0.75	1.0	1.0	0.7	0.2	0.1
0.80	1.0	1.0	0.7	0.2	0.1
0.85	1.0	1.0	0.7	0.1	0.1
0.90	1.0	1.0	0.7	0.1	0.1
0.95	1.0	1.0	0.6	0.2	0.1
1.00	1.0	1.0	0.5	0.1	0.1

4.6 The Random Scenario

In this experiment on the random scenario, the base pattern was generated by the procedure mentioned in Section **??**. The base pattern simulated 400 meetings between 25 nodes, of which only 125 node pairs met each other over the time period from 0 to 50. This base pattern was then repeated for 20 cycles by inducing the changes in each cycle so as to simulate the random scenario. Thus, the contact pattern was generated and the slice point was set to 51. To simulate the random scenario, we induced changes in all of the 20 cycles by choosing one of the following scenarios: new unique node pair scenario, new meeting scenario or time dissimilarity scenario randomly for each of the cycles. Note that, the *Constant Time Duration Slicing Algorithm* can declare the contact pattern generated to be repetitive by specifying the slice point to be anything other than 1020 (the time the last meeting occurred in the entire contact pattern, thus making the entire contact pattern as one slice). However, the proposed algorithm is assumed to have successfully found the contact pattern to be repetitive only if the slice point is found to be in close range of 51. The failure in finding a slice point and its subsequent declaration as a non-repetitive contact pattern is depicted in the results by the slice point 1020.

4.6.0.10 Observation of the effect of variation caused by α and β

In order to observe the variation caused by α and β on the random scenario, a new parameter is defined. Tolerance level is defined as the maximum percentage of changes induced by the

random scenario that still resulted in detecting a slice point in the range 51+10% or 51-10% of the slice point. In order to observe the effect of the variation caused by α and β on the tolerance level, the α value is varied from 0.75 to 1.0 in steps of 0.05 and the β value is varied from 0.80 to 1.0 in steps of 0.05. The slice point is then calculated for each combination of α and β for each percentage of change induced by the random scenario from 0.0 to 1.0 in steps of 0.1 until the slice point obtained goes out of range of the tolerance level. The result obtained is as shown in Figure 4.6.0.10. From the results, we observe that the tolerance level of the *Constant Time Duration Slicing Algorithm* decreases as the β value increases. The tolerance level is also independent of α. From the results, we infer that the *Constant Time Duration Slicing Algorithm* can tolerate a minimum of 10% of the changes caused by the random scenario.

Table 4.4: Effect of α and β on slice point in random scenario

α \ β	**0.80**	**0.85**	**0.90**	**0.95**	**1.00**
0.75	0.3	0.2	0.1	0.1	0.1
0.80	0.3	0.2	0.1	0.1	0.1
0.85	0.3	0.2	0.1	0.1	0.1
0.90	0.3	0.2	0.1	0.1	0.1
0.95	0.3	0.2	0.1	0.1	0.1
1.00	0.3	0.2	0.1	0.1	0.1

4.7 Discussion

After evaluating the *Constant Time Duration Slicing Algorithm* for contact patterns under different scenarios, we noted a few interesting observations. The *Constant Time Duration Slicing Algorithm* was able to tolerate at least 10% of the irregularities that were induced into a perfectly repeating pattern for each of the three cases, individually or combined. We also observed that the detection of the slice point in the close range to its actual value was done by the *Constant Time Duration Slicing Algorithm* for higher values of β (typically in the range $0.8 \leq \beta \leq 1.0$). The detection of the slice point close to its actual value by the *Constant Time Duration Slicing Algorithm* is independent of α for the given range of β .

Chapter 5

An Alternative Distributed Approach

In this chapter, an alternative distributed approach where each node can independently determine its contact time with other nodes is discussed. Firstly, we describe the motivation that led to this approach. We propose an algorithm to find the period of repetition on a per-node basis assuming the notion of a pre-specified period. The experimental results of the proposed algorithm are discussed in the last section of the chapter.

5.0.1 Motivation

In Chapter **??**, we proposed an algorithm to find the period of repetition in the contact pattern of a set of nodes. This approach is equivalent to a centralized approach where all the nodes send their contact histories to a particular node. This node calculates the period over which the contacts of *all* these nodes repeat. The period of repetition is then communicated back to each node. Each node can now determine the time it is going to come in contact with another node using this period. However, it must be observed that since the period of repetition is calculated considering the contacts of all the nodes, it is likely that the period is expanded to accommodate the contacts of all the node pairs. Such an expanded period would overlook the contacts that would occur between specific node pairs that meet every t time units, t being less than the expanded period. For instance, A and B meet every 4 hours, but the centralized approach declared that all the nodes A, B, C, in the system meet every 24 hours. Hence, it can be noticed that A and B is waiting unnecessarily for every 24 hours to exchange a message, while it could have exchanged the message within 4 hours itself. Thus, it becomes necessary to calculate the period of repetition on a per node pair basis. Whenever a node A wants to send a message to a node B, it can calculate the period of repetition of

contacts between itself and B using its own contact pattern. It would now be redundant to follow a centralized approach since the information needed to compute the period is available with the node itself. Also, the reduced impact on the network traffic would be an added advantage. Another practical assumption is made with regard to the format of the contact pattern. Until now, we have assumed that the contact pattern of each node contains the meeting times of itself with every other node, the meeting times increasing in a monotonic fashion. However, after examining several real-world traces such as *UMass DieselNet* [1], *NUS Spring Semester contact pattern* [15] and others, we concluded that it was practical to assume that meeting times are aggregated into certain pre-defined periods, within which they are increasing monotonically. For instance, in the *UMass DieselNet* contact pattern, the logs are maintained on a per day basis, the meeting time within each day is specified as hour : minute: seconds. As mentioned earlier, Delay Tolerant Networks (DTNs) follow a store-carry-forward approach. DTN routing algorithms are designed to meet objectives such as reliability, speed of delivery and others which are specific to the application. In the following sections, we propose an algorithm which calculates the period of repetition on a per node pair basis at the local node itself. The parameters of the algorithm can be manipulated to meet the objectives of the routing algorithm.

5.0.2 The Alternative Distributed Approach

The DTN routing algorithm uses the Probabilistic Contact Graph (PCG) to determine the next intermediate node to forward the message. An example of the probabilistic contact graph is shown in Figure 5.1:

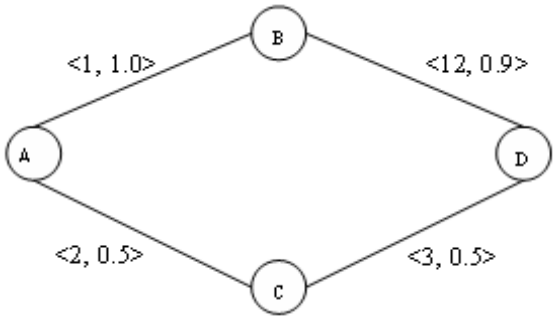

Figure 5.1: A Probabilistic Contact Graph

This example shows that node A meets node B at 1 PM on 100% of the repetition periods. Node A meets node C at 2 PM only on 50% of the repetition periods. Suppose speed of

delivery is the DTN routing objective, the path A - C - D is to be chosen, since the message reaches D by 3 PM. Suppose reliability is the DTN routing objective, the path A - B - D has higher probability of delivery (90%) than A - C - D (25

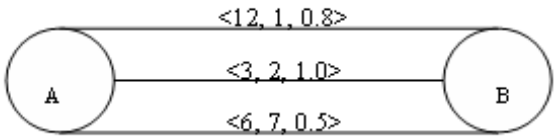

Figure 5.2: An extended version of the PCG for the start day 1

With the introduction of the notion of the aggregation of meeting time into certain predefined periods (say days), the PCG between a pair of nodes A and B can be extended for a particular start day 1 as shown in Figure 5.2. $\langle$t,d,p$\rangle$ denotes that A meets B at time t every d days with p probability of meeting. $\langle 12, 1, 0.8\rangle$ implies that A meets B at 12 PM every day on 80% of all the days starting from day 1. They also meet at 3 PM on every alternate day on 100% of all the days starting from day 1. They meet at 6 PM once in a week with 50% probability. The start day becomes an integral part of the PCG to determine which day of the period are we currently in and is assigned the value of the day that particular version of the PCG was generated. Suppose the current day is 6 and start day is 1, we need to wait another day for using the link $\langle 3, 2, 1.0\rangle$ or can now use the link $\langle 6, 7, 0.5\rangle$. The major objective of determining when a local node would meet another node for the purpose of forwarding a message is scaled down to the generation of the PCG between those two nodes. The extended version of the PCG can be reduced to the PCG by merging the semantically equal edges. For instance, $\langle 3, 2, 1.0\rangle$, $\langle 3, 4, 1.0\rangle$ and $\langle 3, 6, 1.0\rangle$ can all be merged into $\langle 3, 2, 1.0\rangle$. The global solution can now be easily achieved by generating the PCG for all other node pairs. As can be observed, the basic problem that needs to be solved is to determine over a period of how many days does a node A repeatedly meet another node B at a particular time x.

5.0.3 A Local Slicing Algorithm

The simplified problem can be stated as follows. Given the presence or absence of a meeting between a pair of nodes A and B for n days at a particular time x, determine the period of days over which the nodes A and B meet at least once at that particular time x. We

propose an algorithm to solve the stated problem as shown in Algorithm 5.0.1. The bit string bits indicates the presence or absence of the meetings on all the n days for the time x. bits is copied to *comparison_matrix*[1], to check if a meeting has occurred everyday. The *probability*[1] is calculated as the ratio of the number of days the meeting occurred to the total number of days. If *probability*[1] is 1.0 or above *gamma*, the algorithm exits with 1 as the period of repetition. Otherwise, we slice the *comparison_matrix*[1] for $DURATION = 2$. For each of these slices, we determine whether at least one bit is a 1. If yes, the corresponding bit for *comparison_matrix*[2] is set to 1, otherwise 0. Thus, we compress the number of bits by using the *comparison_matrix*[highest factor of $DURATION$ except itself]. For instance, if *comparison_matrix*[1] is 10 01 00 11, then *comparison_matrix*[2] is 1101. Similarly, if *comparison_matrix*[5] is 00000 00000 00101 11001, then *comparison_matrix*[10] is 01. The probability for $DURATION = 2$ is determined as the ratio of the number of bits set to 1 in *comparison_matrix*[2] to the total number of bits in *comparison_matrix*[2] (i.e n/2). The same process is repeated for other $DURATION$ values. $No\_of\_bits\_set\_to\_1(b, l)$ returns the number of bits that ae set to 1 in a bit string b of length l. $highest\_factor\_except\_itself$(x) returns the second highest factor of an integer x. Additionally, the algorithm is parameterized over the routing objectives such as reliability and speed of delivery. If the routing objective is reliability, the duration that has the highest probability of nodes meeting each other is returned as period. If speed of delivery is the routing objective, the smallest duration that is above the threshold *gamma* is returned as period. For example, if the $DURATION$ 5 and 12 have their probabilities as 0.85 and 0.95 while the *gamma* is 0.8, 5 is returned as period if routing objective is speed of delivery and 12, if routing objective is reliability.

5.0.4 Simulation and Results

Experimentation was performed to check the correctness of the algorithm. The experiments were conducted for the meeting of nodes A and B over a period of 365 days. The contacts were forced to repeat every 7 days. When the contacts were repeating on every 7^{th} day for all the 365 days, the proposed algorithm was successful in identifying the period of repetition. The graph depicting the increase in probability as the $DURATION$ approaches 7 is shown in Figure 5.3.

From the figure, it can be observed that the probability increases as the $DURATION$ approaches 7. For every $DURATION$ above 7, the meeting probability remains unity. It can thus be concluded that if reliability is the routing objective, then the $DURATION$

Algorithm 5.0.1 *Local Slicing Algorithm*: *LocalSlicing*()

1: INPUT:A bit string *bits* of length n where bit i is 1 if node A met B at time x on day i, otherwise 0. A threshold *gamma* must be specified for *probability* value if the routing objective is speed of delivery.
2: var $DURATION$
3: var *sub_duration* // to use previously stored result
4: var list *probability* // An array of probabilities for each $DURATION$

5: $DURATION \leftarrow 1$
6: $comparison\_matrix[DURATION] \leftarrow bits$
7: $probability[DURATION] \leftarrow \frac{No\_of\_bits\_set\_to\_1(comparison\_matrix[DURATION], \lceil \frac{n+1}{DURATION} \rceil)}{\lceil \frac{n+1}{DURATION} \rceil}$
8: **if** routing objective = reliability **then**
9: **if** $probability[DURATION] = 1.0$ **then**
10: return $DURATION$
11: **end if**
12: **end if**
13: **if** routing objective = speed of delivery **then**
14: **if** $probability[DURATION] \geq gamma$ **then**
15: return $DURATION$
16: **end if**
17: **end if**

18: **for** $DURATION = 2$ to $\lceil \frac{n}{2} \rceil$ **do**
19: $sub\_duration \leftarrow highest\_factor\_except\_itself(DURATION)$
20: $grouping \leftarrow \frac{DURATION}{sub\_duration}$
21: $compress(comparison\_matrix[DURATION], \lceil \frac{n}{DURATION} \rceil, comparison\_matrix[sub\_duration], \lceil$
22: $probability[DURATION] \leftarrow \frac{No\_of\_bits\_set\_to\_1(comparison\_matrix[DURATION], \lceil \frac{n+1}{DURATION} \rceil)}{\lceil \frac{n+1}{DURATION} \rceil}$
23: **if** routing objective = reliability **then**
24: **if** $probability[DURATION] = 1.0$ **then**
25: return $DURATION$
26: **end if**
27: **end if**
28: **if** routing objective = speed of delivery **then**
29: **if** $probability[DURATION] \geq gamma$ **then**
30: return $DURATION$
31: **end if**
32: **end if**
33: **end for**
34: **if** routing objective = reliability **then**
35: **if** $probability[DURATION] = 1.0$ **then**
36: return smallest $DURATION$ with maximum probability[ ] value.
37: **end if**
38: **end if**
39: **if** routing objective = speed of delivery **then**
40: **if** $probability[DURATION] \geq gamma$ **then**
41: return n
42: **end if**
43: **end if**

Algorithm 5.0.2 procedure *compress*: *compress*()

```
1: INPUT:A bit string A of length len_a, a bit string B of len_b, an integer for grouping.
2: INPUT:Compress every grouping bits of B into 1 (if atleast one bit is 1) in A, 0 otherwise.
3: i ← 0
4: k ← 0
5: while i < len_b do
6:    flag ← 0
7:    for j = 0 to grouping do
8:       if B[i + j] = 1 then
9:          A[k] ← 1
10:         k ← k + 1
11:         flag ← 1
12:         break
13:      end if
14:   end for
15:   if flag = 0 then
16:      A[k] ← 0
17:      k ← k + 1
18:   end if
19:   i ← i + grouping
20: end while
```

corresponding to the highest meeting probability must be chosen. If speed of delivery is the routing objective, the smallest $DURATION$ having a meeting probability above a pre-specified threshold called gamma must be chosen. In order to examine the tolerance of the algorithm to irregular meetings, two experiments were conducted. Both used the perfectly repeating bit string as the base pattern. In the first experiment, new meetings were added in steps of 10% from 0 to 100% of the total number of days and the effect on the period was observed. The results are as shown in Figure 5.4. It can be observed that as meetings occur on more number of days at the same time, the algorithm is successfully decreasing the period.

In the second experiment, the existing meetings in the perfectly repeating bit string were decreased in steps of 10% from 0% to 100% of the total number of existing meetings. The results are shown in Figure 5.5. As the number of meetings is decreased, the algorithm successfully increases the period to accommodate the remaining meetings.

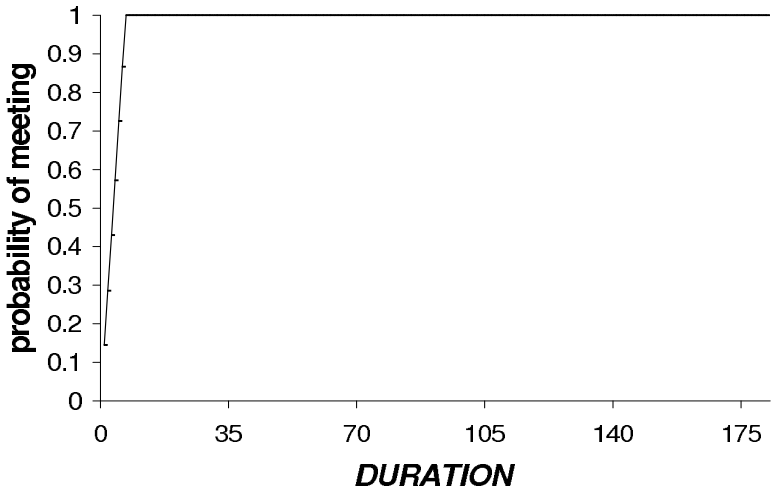

Figure 5.3: Effect of varying $DURATION$ on the probability of meeting

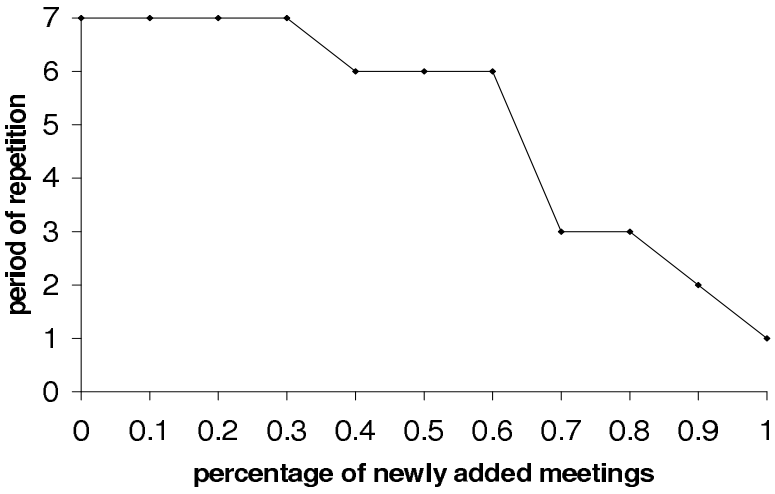

Figure 5.4: Effect of increasing the number of meetings on period

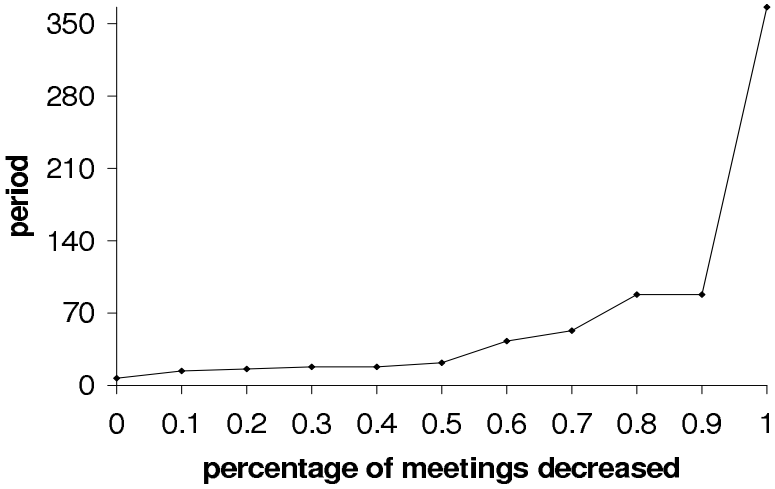

Figure 5.5: Effect of decreasing the number of meetings on period

Chapter 6

Conclusion

In this thesis, we have proposed two similarity measures, namely *Euclidean distance based similarity measure* and *set similarity measure* that is useful to determine the similarity between two contact patterns on a scale of 0 to 1. After experimentation with these proposed measures over simulated contact patterns, we concluded that *Euclidean distance based similarity measure* is more tolerant and less susceptible to irregularities in the contact patterns. Later, we proposed the *Constant Time Duration Slicing Algorithm* to find the periodicity in a contact pattern using the *Euclidean distance based similarity measure* as a similarity measure. The proposed algorithm was evaluated for correctness using simulated and real-world contact patterns. We concluded that the *Constant Time Duration Slicing Algorithm* tolerates a minimum of 10% irregularity in the contact patterns. We also proposed an alternative approach to predict when a node was going to come in contact with another node, given their contact history. An algorithm, *Local Slicing Algorithm* was proposed to determine the period over which a pair of nodes met at least once. Experiments were performed to evaluate the correctness of the proposed algorithm.

Future Work

Some of the possible work that could be carried ahead is as follows. The proposed algorithm *Constant Time Duration Slicing Algorithm* can tolerate only a minimum of 10% irregularity in the contact patterns. Slicing algorithms which can tolerate a higher level of irregularity is necessary. The *Local Slicing Algorithm* does not take into account the effect of the number of meetings occurring within each repetition period on the probability of meeting. Additional improvements can be made on the distributed approach.

Bibliography

[1] 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.

[2] J. Su, A. Goel, and E. de Lara, "An empirical evaluation of the student-net delay tolerant network," in *MobiQuitous 2006: Third Annual International Conference on Mobile and Ubiquitous Systems*, July 2006, pp. 1–10.

[3] L. Song and D. F. Kotz, "Evaluating opportunistic routing protocols with large realistic contact traces," in *CHANTS '07: Proceedings of the second ACM workshop on Challenged networks*. New York, NY, USA: ACM, 2007, pp. 35–42.

[4] 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, 2003, pp. 27–34.