Florida International University
FIU Digital Commons

FIU Electronic Theses and Dissertations | University Graduate School

3-26-2008

An Epistemic Event-based Correlation Scheme for Pervasive Network Management

Vinayak Ganapathy
Florida International University

Follow this and additional works at: http://digitalcommons.fiu.edu/etd

Recommended Citation
Ganapathy, Vinayak, "An Epistemic Event-based Correlation Scheme for Pervasive Network Management" (2008). *FIU Electronic Theses and Dissertations.* Paper 190.
http://digitalcommons.fiu.edu/etd/190

This work is brought to you for free and open access by the University Graduate School at FIU Digital Commons. It has been accepted for inclusion in FIU Electronic Theses and Dissertations by an authorized administrator of FIU Digital Commons. For more information, please contact dcc@fiu.edu.

FLORIDA INTERNATIONAL UNIVERSITY

Miami, Florida

AN EPISTEMIC EVENT-BASED CORRELATION SCHEME FOR PERVASIVE NETWORK MANAGEMENT

A dissertation submitted in partial fulfillment of the

requirements for the degree of

DOCTOR OF PHILOSOPHY

in

ELECTRICAL ENGINEERING

by

Vinayak Ganapathy

2008

To: Interim Dean Amir Mirmiran
College of Engineering and Computing

This dissertation, written by Vinayak Ganapathy, and entitled An Epistemic Event-based Correlation Scheme for Pervasive Network Management, having been approved in respect to style and intellectual content, is referred to you for judgment.

We have read this dissertation and recommend that it be approved.

Ronald Giachetti

Kia Makki

Deng Pan

Kang Yen

Niki Pissinou, Major Professor

Date of Defense: March 26, 2008

The dissertation of Vinayak Ganapathy is approved.

Interim Dean Amir Mirmiran
College of Engineering and Computing

Dean George Walker
University Graduate School

Florida International University, 2008

© Copyright 2008 by Vinayak Ganapathy

All rights reserved.

DEDICATION

To my parents and academic parents.

ACKNOWLEDGMENTS

My profound gratitude goes to my major advisor Dr. Niki Pissinou. If not for all those last-chances, countless hours of hands-on guidance, and support in the form of advice, suggestions, and finance, this dissertation would have remained a dream. She has been instrumental in always maintaining my focus, helping me to become a better researcher and a better person. My special thanks go to two members of my dissertation committee: Dr. Kia Makki and Dr. Kang Yen, who have inspired and continued to supported me in my endeavor for higher education. I am also thankful to other members of my dissertation committee: Dr. Ronald Giachetti and Dr. Deng Pan for their suggestions to help me refine this dissertation.

I am extremely grateful for the tremendous support, encouragement and inspiration I have received from my families - whether they be biological, in Miami or in Delhi. Without their love and prayers, my endeavor would have remained incomplete. I specially thank Dr. P. S. Ayyaswamy, for his guidance and encouragement to ponder on the 'big picture', Dr. Vivek Jayaram, who always encouraged me to 'step past the barrier' and ponder on the 'whys' and Dr. Xiwei Zhao who encouraged me to ponder on the 'hows'.

I acknowledge grant support from National Science Foundation, Department of Transportation, Department of Energy and Department of Homeland Security for this research.

ABSTRACT OF THE DISSERTATION

AN EPISTEMIC EVENT-BASED CORRELATION SCHEME FOR PERVASIVE NETWORK MANAGEMENT

by

Vinayak Ganapathy

Florida International University, 2008

Miami, Florida

Professor Niki Pissinou, Major Professor

Computer networks produce tremendous amounts of event-based data that can be collected and managed to support an increasing number of new classes of pervasive applications. Examples of such applications are network monitoring and crisis management.

Although the problem of distributed event-based management has been addressed in the non-pervasive settings such as the Internet, the domain of pervasive networks has its own characteristics that make these results non-applicable. Many of these applications are based on time-series data that possess the form of time-ordered series of events. Such applications also embody the need to handle large volumes of unexpected events, often modified on-the-fly, containing conflicting information, and dealing with rapidly changing contexts while producing results with low-latency. Correlating events across contextual dimensions holds the key to expanding the capabilities and improving the performance of these applications.

This dissertation addresses this critical challenge. It establishes an effective scheme for complex-event semantic correlation. The scheme examines epistemic uncertainty in computer networks by fusing event synchronization concepts with belief theory. Because of the distributed nature of the event detection, time-delays are considered. Events are no longer

instantaneous, but duration is associated with them. Existing algorithms for synchronizing time are split into two classes, one of which is asserted to provide a faster means for converging time and hence better suited for pervasive network management.

Besides the temporal dimension, the scheme considers imprecision and uncertainty when an event is detected. A belief value is therefore associated with the semantics and the detection of composite events. This belief value is generated by a consensus among participating entities in a computer network. The scheme taps into in-network processing capabilities of pervasive computer networks and can withstand missing or conflicting information gathered from multiple participating entities.

Thus, this dissertation advances knowledge in the field of network management by facilitating the full utilization of characteristics offered by pervasive, distributed and wireless technologies in contemporary and future computer networks.

TABLE OF CONTENTS

CHAPTER PAGE

CHAPTER 1 1
1. Introduction 1
1.1 Preamble 1
1.2 Background 2
1.3 Epistemic Theory and Event Correlation 4
1.4 Motivation 9
1.5 Problem Statement 10
1.6 Hypothesis 11
1.7 Objectives 12
1.8 Significance & Contribution 13
1.9 Methodology 14
1.10 Organization of the Dissertation 14

CHAPTER 2 15
2. Related Work 15
2.1 Introduction 15
2.2 Literature on Time Synchronization 16
2.3 Literature on Event-based Technologies 17
2.3.1 Correlation domain 20
2.3.2 Correlation logic 22
2.3.3 Correlation architecture 25
2.4 Distributed event correlation 26

CHAPTER 3 29
3. Synchronizing Time 29
3.1 Introduction 29
3.2 Real-Time Clock (RTC) 29
3.3 Security 30
3.4 Stability 31
3.5 Comparison of Methods of Time Synchronization 32
3.6 Background 32
3.7 Assumptions 33
3.8 Analysis 35
3.8.1 Comparison with reference to convergence of time 35
3.8.1.1 LS strategy 36
3.8.1.2 GS strategy 37
3.8.2 Comparison with reference to mobility 40
3.8.2.1 Mobility and LS strategies 40
3.8.2.2 Mobility and GS strategies 41
3.8.3 In terms of stochastic time-delay 42
3.9 Simulation 42
3.10 Conclusion 44

CHAPTER 4 45
4. Exploiting Epistemic Uncertainty 45
4.1 Introduction 45
4.2 Candidate 1: Bayesian theory 45
4.3 Candidate 2: Dempster-Shafer's theory 48
4.4 Assumptions 51
4.5 Model 51
4.6 Simulation 54
4.7 Drawbacks – Location awareness 59
4.8 Conclusion 60

CHAPTER 5 61
5. Conclusions 61
5.1 Contribution 61
5.2 Future Work 62
5.2.1 Location Awareness 62
5.2.2 Security and Trustworthiness 63

LIST OF REFERENCES 64

APPENDICES 70

VITA 88

LIST OF FIGURES

FIGURE PAGE

Figure 1: Simple message routing on the Internet 5

Figure 2: Distributed event correlation 7

Figure 3: Describing relation between *d*, *h* and *L* for case I. 38

Figure 4: Describing relation between *d* and *L* for case II. 39

Figure 5: Convergence time vs. probabilities when $n = 9$..... 42

Figure 6: Convergence time vs. probabilities when $n = 55$. 43

Figure 7: Convergence probabilities in 500m × 500m..... 44

Figure 8: Correlation transformations (K_i). 46

Figure 9: Architecture showing correlation transformation. 48

Figure 10: Typical pervasive environment modeled as Wireless Sensor/Actuator Network 53

Figure 11: Changes in belief and conflict in probable causes as a function of hop-count. 55

Figure 12: Changes in belief in an unusual pervasive environment. 56

Figure 13: Changes in belief in a pervasive environment partitioned by a Firewall..... 57

Figure 14: Average increase in message bandwidth for an edge manager entity. 58

Figure 15: Average increase in message bandwidth for a central manager entity. 59

Figure 16: Relation between agreement and bandwidth consumed for location aware nodes.. 62

C H A P T E R 1

1. INTRODUCTION

"God ~~couldn't~~ wanted to be everywhere, so he created ~~mothers~~ computer networks."
– Jewish Proverb
"Logic will get you from A to B. ~~Imagination~~ Computer networks will take you everywhere."
– Albert Einstein

1.1 Preamble

The scope of managing computer networks, known as 'network management' in common parlance, encompasses the entirety of computer networks today. What makes it difficult, challenging and complex is the fact that computer networks have been growing rapidly – so rapid that the reader would certainly feel inconvenience in finding suitable definitions for the terms 'computer' and 'networks' covering the complete spectrum of technologies, and, devices and equipment implementing those technologies. It is imperative that no single network management technology can claim to manage a subset of computer networks, let alone the myriad universe of computer networks, the technologies behind them, and, the services supported by them.

And yet, it is human endeavor to research for that one universal solution, that one silver bullet which will solve the entire problem of network management with mathematical certainty. This dissertation is an attempt to belong to the research body whose goal is to make this dream a reality – a tribute and support to scientists and engineers who have put countless hours of efforts in building and managing computer networks in the past, those who have taken charge of it at present, and those who shall do so in future, so as to support and elevate the standard of living and general well-being of mankind.

1.2 Background

In 1990-2000 decade, it was recognized that unlike the widespread adoption of Internet for enterprise scale transactions, there was no parallel for "monitoring and managing information that flows through the global information systems" [40]. Given the explosive growth of computer networks [9, 10] and ease with which they percolated all spheres of human activity, this observation was surprising. It spawned a lot of research concentrating on streamlining fundamental issues behind Internet-scale networking, for example traffic flows [61] and privacy [46], and led to the conclusion that to understand the activities, operation and behavior of computer networks, and then forecast their operational, administrative, management and provisioning states requires management of tremendous amounts of event-based data produced therein.

Events have been used to describe 'occurrences of interesting phenomena' in a system [40]. Depending on context, the physical interpretation of events varies, for example, as in [35, 40, 46]. The concept of events let to an obvious research area – Simple Event Correlation (not related to Simple Event Correlator (SEC) [65]). Simple event correlation is essentially a set of Event-Condition-Action (ECA) rules for inference-based network management systems [32, 24, 68]. Over time, the process was refined, and currently, a number of methods exist for this kind of network management, also known as 'deterministic' event management. Prominent methods which are currently used in popular commercial network management systems are decision trees [27, 49], and Codebook/Correlation Matrix [57, 68]. These network management systems are deterministic because decision trees and correlation matrices are 'static' – a decision connecting a known set of inputs to a given result must be known prior to programming the network management system.

While such systems have been highly successful in the past [3, 5], new security vulnerabilities coupled with edges of networks gaining more computing power, are forcing network management systems based on such methods in becoming burdened with too many rules. Artificial Intelligence (AI) has been used to circumvent this problem, however, such methods till date are either offline, or require tremendous modeling data to be effective [64]. While there is no dearth of modeling data, such models frequently fall behind real-time restrictions imposed on network management systems, especially as pervasiveness progresses.

Complex correlation of events is gaining reputation as a fast and lean alternative to above mentioned methods for network management [7]. Unlike simple event correlation, complex correlation uses multiple dimensions to carry out correlation – instead of correlating only on primitive events generated by agents or proxies, scope of events is broadened to a higher level by correlating timing data in events [17] and topographical information concerning where the events were generated (spatio-temporal or contextual correlation) [7, 35, 52]. These techniques for correlating events are now clubbed under a common title: Complex Event Processing (CEP), a term given by [40]. Most of the work, even till date still relates to business processing environments [63] and application layer integration [16]. To the best of author's knowledge, no application of CEP currently exists or is being pursued in the area of network management. The reasons for this are two-fold:

(a) Unless CEP is integrated with techniques which allow fair amount of distributivity over event processing, it is unlikely to be a popular network management tool which can be integrated into managers and managed devices and equipment. In context of pervasive systems, distributing network management load among a handful of powerful servers does not constitute distributed network management.

(b) Classical CEP (primitive event composition to obtain composite events) can be reduced to the traditional decision tree/correlation matrix methods yielding just another programmatic way to handle incoming event information (e.g., Simple Event Correlator (SEC) [65]).

Furthermore, pervasive computer networks have their own characteristics [1] which are not amenable to either contemporary techniques or CEP techniques as is:

(a) Many applications of pervasive computer networks are based on 'time-series' data. This means that management traffic flowing for such applications will contain a time-ordered series of events. This temporal dimension forces the consideration of events to be associated with duration rather than being instantaneous [17].

(b) Most applications of pervasive computer networks need to handle large volumes of events. Many times, events may turn unexpectedly fulminant, are often modified on-the-fly, are heavily dependent on topography of the computer network, contain conflicting information and deal with rapidly changing contexts, and are constrained to provide results with low latency.

1.3 Epistemic Theory and Event Correlation

This dissertation proposes to extend CEP techniques by considering uncertainty in decision regarding correlation of events. The reason for this proposition, aimed at a departure from 'deterministic' network management, is elucidated by an example as follows.

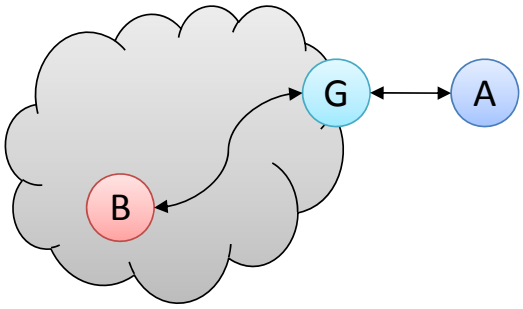

Figure 1: Simple message routing on the Internet

From Figure 1, consider that an entity A wishes to know if an entity B somewhere in the Internet is available for communication. Typically, entity A will send an Internet Control Message Protocol (ICMP) control message, more commonly known as a 'ping', to gateway G, which will route the ICMP message across the Internet via a series of routing elements that can determine the route to entity B. If all goes well, entity B will respond back favorably to the ICMP control message of entity A. However, if entity A fails to obtain a response from entity B, the failure can be attributed to any of the following probable causes:

(a) Transmission Control Protocol/Internet Protocol (TCP/IP) stack embedded in entity B has faulty or disabled ICMP module: Since TCP/IP stack is essentially a piece of configurable software, a given operating system (OS) platform may choose to disable portions of the software, may have an altered design, may be working under reduced resource availability, etc. – a number of reasons by which the TCP/IP stack may not operate as per the actual protocol specifications.

(b) ICMP messages are blocked within the network: Since Firewalls are typically designed to ignore ICMP messages by default, entity A will never know if the ICMP message actually reached entity B. Typically, a Firewall embedded in entity A will automatically discard the ICMP message with no local interactive notification regarding the behavior.

(c) Entity B is disconnected: Physical connectivity to entity B may be absent.

(d) Entity B is shutdown: Entity B may not be at a 'run-level' at which networking services are operational, or, may have entered a reduced power state, or, might have simply been shut down for say, replacement.

(e) ICMP Time-to-Live (TTL) expired in transit: The route between communicating entities A and B might contain higher number of intermediate relay entities than the intervening communicating protocol negotiation allows for.

Except for reason mentioned in (e), no other reason will ever let entity A know the true reason for the failure behind communication of ICMP control message. The importance of true knowledge of failure increases with the fact that entities A or B are representing important customers, or are critical devices, or have stringent Service Level Agreements (SLAs) – any of which can be important to the business within the computer network.

In traditional network management or classical CEP, reasons (a) through (d) will be assigned equal probabilities as a probable cause – 'unbiased' or 'desperate' assignment. This assignment is more popularly known as Laplace's Principle of Insufficient Reason [53]. This means that if the entity, A, is a network manager, it can inform technicians and administrators in charge of the computer network that entity B is not accessible, but it cannot tell them the reason for the failure. At best, it can make an educated guess (for example, in reason (e), where an intermediate entity, such as gateway G sends a TTL expiry message to entity A notifying the exact cause of the problem). Otherwise, it can only claim that reasons (a) through (d) may have occurred with equal probability. Then, the problem is how can a network management system gain higher situational awareness so that network managers can make a more informed decision regarding the state of the computer network?

A two pronged approach may be used to solve this problem:

(a) Use the knowledge of entities C, D,... present in the computer network (such as, use of contextual correlation of data; Figure 2),

(b) Avoid desperation in determining probable causes by other means (such as, use of historical or expert data).

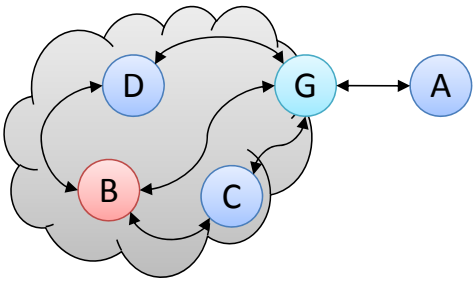

Figure 2: Distributed event correlation

It is intended to exploit knowledge possessed by such entities present in the computer network that can potentially provide historical or 'expert' data regarding communication with entity B (the entity in question). Such entities (entities C and D in Figure 2) can facilitate management entities (entities A and G in Figure 2) to enhance decision-making capabilities regarding failures – knowledge can be exploited via combination of contextual dimensions. New flavors of SNMP (for example, v3, supporting AgentX protocol [12]) allow dynamic Agent-Agent, Manager-Manager communication and makes this scenario industrially feasible.

The process of determining a probable cause when given a (set of) symptom(s) can be considered to be the process of induction – determine and generalize behavior of the system given how a small subset of the system is operating. The system in question could possibly be as large as the Internet, and a small subset is considered as a part of this computer network, for example, the sub-network as shown in Figure 2.

When all possible symptoms are known, the decision regarding the computer network's affliction or amelioration can be obtained using decision trees and codebooks/correlation matrices (deterministic network management). However, all possible symptoms may not be known, decision trees may not be correct, and codebooks/correlation matrices may contain insufficient hamming distances between probable causes. Assuming that all available historical data is considered to have the form of events flowing within or across the computer network, there might be uncertainty in knowledge regarding the available historical data, and, more often than not, there might not be enough available historical data, even if tremendous amounts of it flowed by in the past.

Epistemic theory [45] may be considered as an appropriate technique in dealing with such uncertainty. In particular, to account for incompleteness and causality – two important factors concerning event processing for network management, this dissertation considers Dempster-Shafer's theory [55] as a suitable flavor to analyze the network management problem [8]. Dempster-Shafer's theory is useful whenever symptoms and probable causes cannot be fully enumerated or when data characterizing them is missing [54]. Also, apart from accommodating for missing or conflicting evidence, Dempster-Shafer's theory can combine these from multiple sources of data, whether they are in agreement or in conflict. Such features make the theory useful when data precision is compromised, or, is obtained based on previously collected domain knowledge of unverifiable origin. Computer networks neatly fall in such a category – uncertainty which results from lack of knowledge about the system, and depends heavily upon attributes of the measuring entity. The nature of evidence collected for generating consensus may vary across contextual dimensions.

1.4 Motivation

Motivation behind this dissertation comes from work carried out at Telecommunications & Information Technology Institute (IT^2), Florida International University, to obtain competitive research funding in the area of secure and context aware sensor networks.

Research carried out by the author's major professor suggests that rapid growth of computer networks entails that network management systems must adopt event-based technologies to keep up with growth and diversity [47]. It has been shown that due to numerous advantages offered by event-based technologies, network management systems are robust and flexible. They are robust because even if a portion of a network fails or is afflicted – the distributed system architecture allows them to localize, partition and contain the failure. They are flexible because they offer a variable degree of coupling between elements of a network, thus allowing seamless changes in network constitution and topology.

The research has demonstrated that event-based technologies offer wide-ranging architectural applicability:

(a) Event-based technologies provided mechanisms to distribute functionality and operations across computing equipment, for example, publisher-subscriber architectures, remote procedure calls, remote method invocations, etc. [42],

(b) Events lend themselves well to language analysis, and thus theoretical rigor [52],

(c) Event-based technologies offer variable degrees of inter and intra-system coupling: tight, loose, or hybrid, and thus, provide flexibility in function segregation and interface definition [42],

(d) Manipulation of events via techniques of composition and correlation offers schemes for bandwidth conservation, hierarchical management designs and integration patterns, for example queuing architectures [40].

Consequently, the advantages offered by event-based technologies are central to this dissertation. The approach for event-based network management has been factored in three ways – the extent of the computer network from which the generated events are considered, the techniques used to generate, process and capture events, and, the type of events that are generated, processed and analyzed.

The reader may note that these factors can be directly interpreted as the extent of computer network which the network management system manages, the technique of management employed, and, the kind of network over which the composition is carried out. It is the interplay of these factors that shaped the evolution of network management systems as they are seen today. The only difference is the approach – event-based technologies assert that growth of computer networks, whether by scale, or, by diversity, entail changes in perception and operation of the network management systems.

1.5 Problem Statement

There is little indication that future computer networks will use the same technologies as contemporary computer networks. Future computer networks are envisioned to be pervasive with a majority percentage of enabling applications being based on advantages offered by wireless technologies. Additionally, it is envisioned that they shall tightly integrate with critical functions (civil and defense functions such as such as power grids, intermodal transportation, environment monitoring, reconnaissance, warfare, etc.) which form the lifeline of societies so that the resultant would be a complete digital economy.

Thus, management of neo-contemporary and future computer networks will require research into the use of pervasive network management systems. These systems must enable distributed, large-volume, event-based processing for supporting novel, yet important functionalities such as real-time function distribution among participating manager and managed entities, flexible and efficient intra and inter-system interfaces, and, event correlation and composition along contextual dimensions for conserving management bandwidth and power consumption.

Thus, the problem explored in this dissertation is to design a mechanism that supports, or plugs into pervasive network management architectures so that they can:

(a) Enable network management functionality while handling large volumes of events, but consuming low bandwidth,

(b) Support network management functionality using in-network processing and correlation of events along contextual dimensions,

(c) Maintain reasonable anonymity of participating entities,

(d) Withstand missing and conflicting information gathered from multiple participating entities, and,

(e) Provide management feedback with low latency.

1.6 Hypothesis

A lightweight, distributed, large-volume, event-based technique which exploits epistemic uncertainty to correlate events along contextual dimensions can provide a successful technique for enabling management of large-scale and pervasive contemporary and future computer networks.

1.7 Objectives

This dissertation attempts to tackle two of the many important challenges that face pervasive network management systems –

(a) How long would it take for anyone managing a pervasive computer network to actually be able to start managing the network?

This challenge is related to time. Unlike centralized paradigms which have a cardinal notion of time residing in a single clock, distributed paradigms and indeed, pervasive computer networks operate with multiple clocks as references. All these clocks must be synchronized mathematically if network administrators in charge of the pervasive computer network are to ever obtain a global view of the computer network's operations. In pervasive computer networks, the challenge increases in complexity as entities join or leave the network, demonstrate malicious behavior, may refuse to cooperate in communications relay, etc.

(b) Is there an event-based, distributed and lightweight mechanism that can support pervasive network management functions?

This challenge is related to what future pervasive network management systems will require to operate. Once the notion of time in pervasive computer networks is defined, event-based architectures would be required to provide network management functionality which handles high volumes of events, consume minimal bandwidth and exploit in-network processing. Thus, unlike centralized paradigms where a single entity assumes all network management responsibilities, or, if the need be, delegates few trivial responsibilities to a limited number of trusted peers, all participating entities would be expected to contribute to network management functionality while

consuming a small fraction of bandwidth available to them. This will increase function distributivity in the network so as to compensate for network scalability, and thus, will facilitate the network management functions to adapt to changes in network topology, traffic conditions, enhancement or reduction in any/all management functionality, and, event correlation logic along contextual dimensions.

1.8 Significance & Contribution

This dissertation paves the way for supporting management functions for contemporary and future computer networks with limited addition and overhead. Such an initiative is necessary when growth in scale and diversity of computer networks is rapid and future trends are unknown. To accommodate such a scenario, it is required to develop network management technologies which are capable of managing computer networks which are much more complicated than existing ones. While existing research literature points to the use of event-based technologies as a suitable candidate for the purpose, most applications lie in simple event processing [64], gathering business intelligence [40], or data mining [67]. These approaches rarely exploit important characteristics of pervasive computer networks, for example, in-network processing, correlation of events across contextual dimensions, or, are either too trivial or too complex for majority of network management applications.

The dissertation contributes to existing research on network management by using an existing, well tested mathematical tool incorporating epistemic uncertainty – Dempster-Shafer's theory, to provide a novel solution for contemporary and future network management. It establishes an effective scheme for complex-event semantic correlation – by incorporating epistemic uncertainty, the scheme fuses event synchronization concepts with belief theory. Many engineering applications based on this theory have demonstrated its

validity [54]. This dissertation empirically demonstrates the suitability of the theory (elucidated in Chapter 4), boundary conditions for its application to a pervasive network management environment (elucidated in Chapter 3) and results obtained by application of the theory to many scenarios (again, as elucidated in Chapter 4). Because of the distributed nature of detecting events, the dissertation considers time-delays, thereby associating events with duration, and takes into consideration imprecision and uncertainty in event detection by associating belief values generated by a consensus among participating entities in pervasive computer networks.

1.9 Methodology

This dissertation employs modeling and simulation. Pervasive computer networks have densities high enough that physical experimentation is difficult to implement. Consequently, simulation software was used to provide empirical results. The simulation software used is ns-2 [41]. ns-2 has had a respectable reputation as an accurate discrete-event network simulator. With provision and working examples for many computer network scenarios accurately reflecting real world, this simulation software has been used to provide all the results presented in this dissertation.

1.10 Organization of the Dissertation

Chapter 2 elucidates the existing research literature examined for the purpose of this dissertation. Time synchronization and its requirement for event-based network management are detailed in Chapter 3. A novel method for event-based, in-network, and non-deterministic pervasive network management is detailed in Chapter 4. Conclusion and future work is presented in Chapter 4.

C H A P T E R 2

2. RELATED WORK

Nanos gigantium humeris insidentes.
– ~~Greek~~ Computer Mythology
We do not see ~~things~~ computers as they are; we see ~~things~~ computers as we are.
– Talmud

2.1 Introduction

In this chapter, related work concerning the dissertation is presented as follows:

First, research related to synchronization of time is presented. Much of the research concerns itself with convergence to and sustenance of a single time reference in a distributed system. In Chapter 3 all algorithms presented in this chapter will be split into two classes, one of which will be shown to be the recommended class of algorithms for pervasive event-based network management.

Second, research regarding event-based technologies as it applies to network management is presented. Much of the research concerns itself with techniques for correlating events and their facets.

Third, research regarding Dempster-Shafer's theory is not presented. Much literature regarding the theory, its applicability (for example, [58]), and its extensions exist in comprehensive works, for example, [54], which even includes a large collection of references regarding applications of the theory to various branches of engineering, for example, image processing and robotics.

2.2 Literature on Time Synchronization

Research regarding time synchronization in mobile ad-hoc networks (MANETs) mainly concerns itself with convergence of local time via remote inter-process message exchange. Issues of concern include compensation for physical hardware's clock's skewness and kurtosis [70], influence of network dynamics and, network designs regarding application [31] or resource limitations [15, 28]. Additionally, fancier interfaces may be highly directional, multi-frequency antennas [11, 66] with abilities of time-stamping transmitted packets as far down the physical layer as possible [31, 37]. Relay mechanisms entail the use of a dominating set of nodes which cover the entire network for effectively diffusing the synchronization primitives [20]. The solutions either assume the existence of this set or form one of their own.

Also, a rich set of solutions exist for MANET time synchronization [51] and are available as lightweight time synchronization protocols. Based on assumed criteria and approach adopted for time synchronization, recent research in time synchronization has been categorized into six classes [51] – time sources may be internal or external, synchronization may be carried out continuously or on demand, domain of synchronization may be flat or hierarchical, approach to synchronization may based on determining clock rate versus determining clock offsets, nodes may use an operational clock over the actual physical clock or synchronize invasively, and, synchronization may be carried out instantly or spread over time.

This dissertation considers that contiguous time is an important requirement for applications in MANETs. In research regarding time synchronization, seminal contribution by Lamport on event ordering in distributed systems [36] defines the rule – preserve event order and causality using forward clocks. An important consequence of this has been the adoption of monotonically increasing virtual clocks in any application design concerning distributed

systems, including mobile and ad hoc networks. In IEEE 802.11 IBSS (Independent Basic Service Set) specification [30], clocks leap to the fastest known virtual clock in their neighborhood to achieve convergence – faster clocks synchronize slower clocks (TSF).

The dissertation adds an additional perspective to classify recent research into two distinct groups. They are global time synchronization (GS), which requires clocks in all participating nodes leap to the fastest clock in the system [15, 20, 44, 50, 56, 59, 70], and, local time synchronization (LS), which requires every participating node to preserve its local clock and only record time offset to its neighbors [11, 18, 29, 33, 37, 43]. An exception case of this classification is the result of reference broadcasting synchronization (RBS) [14], which combines both local and global time synchronization strategies. It uses GS when synchronizing within one broadcast area, but uses LS when reference nodes corresponding to different broadcast areas exchange time-synchronization messages. Moreover, a comparison is carried out between these two distinct groups, in terms of their performance on synchronizing time over MANET.

The first known research into the application of IEEE 802.11 standard [30] to MANETs is provided in [43]. This research follows [43] in adopting the use of IEEE 802.11 standard [30] to MANETs as it helps leverage the development of real-time communication protocols to be based on the standard.

2.3 Literature on Event-based Technologies

Events have already been defined as changes in parameters of interest. Parameters may be a computer network's system parameters or individual elements – called managed objects. Associating these events with one another in useful ways is known as event correlation [64].

There are two types of events: primitive and composite [39]. Primitive events are pre-defined in a system and their detection/generation mechanism is embedded in the system. Composite events are formed from primitive and/or other composite events [46], each of which is then called a component event. An 'event correlation engine' detects the occurrences of these composite events. Event correlation may be carried out at multiple points of the computer network - elements, Object Request Broker (ORB), proxy, etc. (spatial event correlation), and at various points of time - periodic or aperiodic, causal etc. (temporal, causal event correlation) and is one of the central techniques in managing high volume event messages [32, 39]. Event correlation may be executed via the following means, alone or in combination:

(a) Compression, suppression, generalization and homogenization:

Compression optimizes the flow of events by representing multiple instances of the same event using a transformation. For example, gauges and counters used to track Management Information Base (MIB) parameters in SNMP. It may be noted that a transformation may result in generation of events different from the events being monitored, for example, when a counter wraps around or crosses a threshold. Suppression optimizes the flow of events by differentiating events based on priorities associated or embedded in them. For example, tuples consisting of events and their priority are passed through a priority queue. Queuing discipline, for example, priority First-In-First-Out (FIFO), event aging and consequently incremental priority assignment, or, unqualified discard, etc. are implementation choices. Generalization [21] optimizes the flow of events by differentiating events based on their origin. For example, tuples consisting of events and their origin passed through to a multiqueue

where each queue pertains to an element, an interface, or part of the network, etc. Homogenization optimizes the flow of events by differentiating events based on the type/classification associated or embedded in them. For example, tuples consisting of events and their type passed through to a multiqueue where each queue pertains to an event's type definition. Homogenization can be preceded by or followed by suppression for higher resolution differentiation of events. When event type is priority itself, homogenization transforms to suppression.

(b) Composition:

Causal Composition [26] – Using a sufficient number of relationships which map events apriori to one another, a rule trail can be generated, which in turn can facilitate automatic determination of cause from a set of symptoms.

Temporal Composition [38] – Using a sufficient number of relationships which map event sequences to one another, a rule trail can be generated, which in turn can facilitate automatic determination of cause from a sequence of symptoms. Temporal Composition is considered more useful because it is more complete than causal composition [26, 64].

The ability of execution is governed by the network administrator, thus, the means of correlation and its execution are related to the extent allowed.

Event correlation was first analyzed by [32] as frequent episodes in alarm sequences. The analysis of event correlation was done apriori via model-based reasoning. Subsequently, research has focused on applying a number of research methods on both apriori and posteriori event correlation. By application of various constraint forms on data mining, rule-based reasoning, and network topology analysis, tighter control and efficiency on data mining,

algorithms for data mining and rule generation has been achieved. Due to complexity and scale of most commercially important networks (wireless and wireline), event correlation has also been studied from the perspective of event and alarm propagation through layered model of networks and model-based reasoning.

Event correlation must be viewed from three perspectives:

(a) Correlation domain (scope)

(b) Correlation logic (definition)

(c) Correlation architecture (site diversity)

2.3.1 Correlation domain

The scope of event correlation is of three kinds [72]:

(a) Event range covers the space of all generator entities within the scope of a single network element (elements covering the same physical and/or data link layer address masks)

(b) Event range covers the space of all generator entities within the scope of a group of network elements in the same subnetwork (network elements covering the same network layer address masks)

(c) Event range covers the space of all generator entities within the scope of a group of network elements in the same administrative domain

By applying the above perspective of correlation domain to any network – for example, IP based networks, it can be seen that as scope of event correlation is broadened, the number of network elements increases in progression. This implies a significant increase in the events generated in the network and also the number of interfaces across which management data must stream between interacting entities. Any event correlation engine must be able to

scale with the correlation domain. By avoiding unique or preferred ingress and egress interfaces in the network, the tendency of network management centralization can be avoided. Advertised or dynamically chosen ingress and egress points shall maintain distributivity.

Some network management products, such as [27, 57], circumvent the problem of scaling correlation domain at the administrative domain level by attacking the problem on the lines of routing – by maintaining an event correlation hierarchy. Having a hierarchy assumes that all peers binding to the same hierarchical level have equal computational and/or communication functionality – intuitive, because majority of homogeneous network elements will be identical functionally. If network elements are heterogeneous, they can be considered functionally equal if they support a common set of functions, or, if the weaker peers are assisted by external function support via proxy devices and appropriately designed convergence protocols. Hierarchical routing has been immensely successful – IP based as well as ISO based environments use hierarchical routing. The nemesis of hierarchical routing lies in mobility and mobility management. However, this again brings back network management centralization, a perspective which can be subject to debate.

The alternative to the event correlation hierarchy is a distributed event correlation system wherein all collaborating peers acknowledge, and respect specialized roles of each other. The challenge lies in how best can each role be utilized efficiently and optimally. The functionality of acknowledging and characterizing the specialized roles of peers can be carried out via share of management information via corresponding layer management entities in peers. Considering a functionally disjoint set of peers, a system task can be broken down to subtasks which are functionally disjoint in such a way that each subtask can be performed by a corresponding peer efficiently and optimally while respecting distributed computing principles.

From the point of view of scope it is also important to consider whether a connection-oriented or connection-less communication is used. This is because a connection-oriented architecture requires additional channel maintenance and handshake. Also, resource awareness may require the elements to maintain lower occupancy (number of in/out links) for better efficiency. Whichever be the architecture used, synchronization points shall allow 'check-pointing' correlation into meaningful units which allow dialog completion between synchronizing peers. The synchronization points can be of two types - major and minor.

2.3.2 Correlation logic

The definition perspective of event correlation is of five kinds [64]:

(a) Case-based reasoning:

Case-based reasoning [21] is a learning system which defines a case over a set of events. This case is then compared to an existing database of cases for maximum similarity. Maximum similarity is automatically implied by maximum relevance of the set of events which define cases via the principle of optimality. However, relevance of events to a case is not easy to define. Incorrect relevance of event sets can result in skewed similarity and consequently incorrect cause determination by correlation engines. If the case library is large, it will not scale well with correlation scope. Of all the kinds of correlation logic discussed in this section, case-based reasoning is the only research area which incorporates learning [64].

(b) Codebook-based reasoning:

Codebook-based reasoning [68] is based on a 2-dimensional correlation matrix between events. One dimension of the matrix is the primitive events generated by the network elements and the other dimension is the composite events or event

notifications considered as alarms or symptoms. The matrix relates each primitive event to an alarm by either probabilistic or (for a simple case) boolean values. For example, a primitive event may be designated to contribute to an alarm 75% of the time (probabilistic) or may (1)/may not (0) contribute (boolean) to an alarm. Once the correlation matrix is generated, it can be refined via correlation scope and/or compression, suppression, generalization, homogenization to generate a codebook. A second matrix is required which relates the primitive events to probable cause(s). Using the codebook, the correlation engine can process an event stream for comparison with event sets defined in it. Once specific primitive events are singled out to maximally contribute to current alarms, they can be back-correlated to the probable cause(s). The degree to which the correlation scope and/or compression, suppression, generalization, homogenization are applied determine the accuracy of codebook-based reasoning. Similar to the concept of hamming distances, if the alarms are too tightly related to singleton primitive events, the codebook will be small but susceptible to incorrectly identify spurious alarms or ignore missing alarms. If the alarms are loosely related to multiple numbers of primitive events, the correlation matrix may not scale well with correlation scope. [57] implements codebook-based reasoning [34].

(c) Model-based reasoning:

Model-based reasoning [72] falls under the domain of artificial intelligence. Sufficient accurate models of each managed entity are used as frameworks which are populated with current network entities' data. Snapshots of the network at given times are extrapolated via modeling to generate the pattern of events which the actual network should generate in future. Given the network state approaches an anomaly, the models

will provide maintenance engineers with most plausible causes of the anomaly via backtracking. Model-based reasoning requires the correlation engine to accurately catalog the entities in the correlation scope, and choose the best model for those entities whose model is not available or modeled before. Once the models are in place, the main job of the correlation engine is to keep ahead of real-time monitoring in obtaining the network state. Object oriented paradigm lends itself easily to model-based reasoning, however, because each network entity needs to be modeled, model-based reasoning does not scale well with increasing correlation scope. [4] and [32] implement model-based reasoning.

(d) Rule-based reasoning:

In rule-based reasoning [64], a non-empty set of event(s) is passed through a logic equation. A true event set implies a true rule and consequently, the correlation engine can notify or trigger management entities for pre-determined actions. Rule-based reasoning is rigid and optimal for well-understood networks only. As the correlation scope increases, number of rules required to cover the event range increases. Consequently, rule-based reasoning does not scale well with correlation scope. Most network element managers and agents implement rule-based reasoning.

(e) State transition graph-based reasoning:

In state transition graph-based reasoning, state transition graphs are finite state machines where the nodes are network's current state and the transitions are the actions to be carried out when in the current state. Sub-sequences of events and current symptoms of the network are used to run the finite state machines. Depending on the transitions defined, the correlation engine determines the plausible cause from

the output of the finite state machine. Just like other reasoning mechanisms, the determination of plausible cause is as good as the definition of transition between various states. Incorrectly defined states will result in misdirected transition paths and consequently determination of incorrect plausible causes.

2.3.3 Correlation architecture

Networks and network elements exhibit a large diversity:

(a) The technology of network elements can range between state-of-the-art to legacy,

(b) Network elements can be manufactured by a broad spectrum of vendors, each interpreting standards and RFCs differently,

(c) Network elements of a given manufacturer can have multiple versions and families,

(d) Network elements may be wireless, mobile,

(e) Multiple gateways at each layer, especially data-link, network, and transport layers will alter protocol state and flow, etc.

This diversity introduces three important parameters which must be taken into account for correlation:

(a) Accuracy of relative temporal distance between events generated at different sections of the network [22]

Inherent differences in bandwidth, topology, and speed can introduce clock skews in event timestamps as events propagate across sections of the network. Uncompensated timestamps can result in false correlation or missed correlation patterns (false alarms which are generated when propagation delay is sufficient to trigger the truth of an attack pattern and undetected alarms which are generated when the propagation delay is sufficient to avoid satisfying the truth requirement of the attack pattern).

(b) Mediation required to translate events to suit the processing capabilities of destined correlation process(es).

(c) Resource delegation for correlation

Mobile agents with privileges of autonomy have been proposed to assist in distributed processing – primarily for attaining optimality in network functions [13]. However, the basis of using mobile agents appears weak, and has limited support among event-based communication models [42]. Unless protected management channels, and trust protocols among collaborating entities are established, the solution does not appear viable in the face of currently vulnerable OSs.

2.4 Distributed event correlation

Originally, distributed system architectures comprising of large orders of miniature computing devices coalesced via an OS designed to work over computer interconnections were proposed to exploit computing scalability and performance, and physical robustness. Computer networks extend the idea of physical robustness by being geographically distributed – based on an assumption that providence and wars cannot afflict large geographic areas simultaneously. However, geographic expansion of a computing system brings forth multiple issues. Without loss of generality, these are called distributed computing issues from a network management point-of-view. Current research literature focuses on four distributed computing issues:

(a) Distributed event models

A number of existing event-based distributed communication models have been investigated [42]. Each has strengths and weaknesses which are claimed to be strongly influenced by applications which the models are intended to serve. Some important

features of the models are methods of domain dispersal of management traffic optionally with anonymity, filtering of events, central and distributed mediation, mobile device support, and service delegation. Common Object Request Broker Architecture (CORBA) has been extensively used in many large scale carrier-grade network management systems and is an industry standard.

(b) Time synchronization

Temporal accuracy of event composition, and event correlation (ECA or otherwise) depends upon time synchronization between the sources of events. Due to this dependency, time synchronization is considered to be critical to distributed system infrastructure [15]. Temporal accuracy can be conserved in the face of clock-skew or propagation skew [22]. However, the current solutions require multiple processes and/or equipment like calibration probes, monitors and controllers to coordinate the effort. Extensions for mobility in ad hoc networks have also been investigated [15].

(c) Language and semantics

Event correlation requires a language and computational model to be formalized [52]. All frameworks proposed for event correlation use or formalize one [2, 23, 26, 35, 72]. The languages are first-order, use finite state automata, and may optionally choose between boolean and short-circuit evaluation of conjunctive or disjunctive conditions. [52] also mentions other specification languages based on Backus-Naur Form (BNF) and its derivatives.

(d) Computing affinity

Though more popularly used in alert correlation for Intrusion Detection Systems, agents (autonomous, mobile, or both) have interesting applications in distributed systems. Autonomous agents separate their context from the host system to resist subversion and achieve a degree of fault tolerance [6]. Mobile agents go a step further by carrying out many other functions apart from data collection – delay analysis, reconfiguration, etc. [13]. Because mobile agents can migrate processes, they can be used to strategically locate correlation engines across the network for optimizing any/all of the above distributed computing issues, and dynamically adapt to changing network conditions and processing power affinity).

C H A P T E R 3

3. SYNCHRONIZING TIME[1]

"Everywhere you go, take ~~a smile~~ time with you."
– Sasha Azevado
"All men by nature desire to ~~know~~ synchronize."
– Britannica

3.1 Introduction

In context of this dissertation, time synchronization is considered from the perspective of pervasive network management. This implies that more often than not, entities requiring synchronization of time wish to do so with minimal communication effort, minimal resource requirements and securely with a guarantee on the stability of time. While minimal communication effort and minimal resource requirements are generic pervasive requirements, security and stability warrant an explanation. Following these explanations, the comparison of two broad classes of time synchronization – GS and LS strategies are compared. Analysis and simulation of the strategies conclude the chapter.

3.2 Real-Time Clock (RTC)

All hardware platforms maintain an internal register with a given precision (typically 16, 32, 64 or 128 bits) which is driven by a hardware interrupt generated by a crystal oscillator. The register maintains a counter whose value increments for a given number of oscillations of the crystal oscillator. For example, if a hardware platform incorporates a 4 MHz crystal oscillator, the hardware will increment the register by one unit once for every 4 million times

---

[1] In this chapter, 'node' unambiguously refers to computing hosts, devices or equipment.

the crystal oscillates. This provides the platform with a clock whose granularity is 1s. While 1s is no longer a suitable, clocks with higher resolutions lying in the range of 25ns to 1ms are more typical today. This setup is known in engineering parlance as a Real-Time Clock (RTC). The physics (piezo-electricity) behind the crystal oscillator decides the number of oscillations which in turn, may depend on numerous environmental factors – a typical one being temperature. Because the environment of pervasive entities may not be controllable, a slight drift in the time-keeping will occur. This error accumulates over time to become sizable enough that it may affect time-based operations of the hardware platform and the software supported by it. It is then that time synchronization is done to correct the error introduced by the drift.

3.3 Security

Security is an important requirement in synchronizing time because of the way software is designed to derive its time from hardware. Typically, the RTC only increments a designated register depending on the count resolution set by the hardware platform, but it is up to the software to fill in the correct initial value and keep it updated. This means that the software in-charge of the platform is the one which decides what time it actually is. This may not be so much of a problem for ephemeral processes; however, indiscriminate change in the RTC value might disturb the states of most other processes, especially those which last for the entire up-time of the software system. Prominent examples of such processes are security and communication processes.

The main issue regarding security is the fact that if the software in one entity chooses a malicious peer entity to provide it with a value of time, the malicious peer entity can control the network behavior of the unsuspecting entity. An existing standard for wireless

communications in MANETs – IEEE 802.11 [30] mandates that a given peer may adopt the fastest available clock in their neighborhood. This effectively means that given a bunch of pervasive entities which used their ad hoc network to exchange time synchronization primitives, all entities will jump to the clock of the fastest peer among themselves. It is apparent that this will lead to a security related problem if a malicious peer decides to continuously broadcast the highest possible RTC value as its time synchronization primitive. This will force its peers to adopt its time and reset their clocks to a null value every time they receive time synchronization primitives. In sum, the malicious peer would have managed to stop the clocks of all unsuspecting peers and therefore collapse the network into a stasis state.

3.4 Stability

As in the case of security, stability is also an important requirement in synchronizing time because of the way software is designed to derive its time from hardware. Stability too may not be so much of a problem for ephemeral processes; however, like security, indiscriminate change in the RTC value might disturb the states of most other processes.

The main issue regarding stability is the fact that if the software in an entity frequently updates the RTC (forcibly or otherwise), many process states and finite state machines might evaluate indeterminate states or error conditions frequently. A change in RTC value would require schedulers for various processes, including those of the OS itself to restart relative to the new RTC value. If done frequently enough, these processes would soak up the majority of the scheduling quota which would have otherwise been allotted to processes of lower priority – effectively leading to a forced process starvation. Since most schedulers execute at very high priorities, it would be easy for a malicious peer entity to disrupt existing networks with unstable time synchronization primitives.

3.5 Comparison of Methods of Time Synchronization

One of the contributions of this dissertation lies in classifying existing methods of synchronizing time into two broad categories. This classification permits the reader to make an informed decision regarding the class of algorithm s/he deems suitable for a given application. This dissertation itself makes such a choice as discussed in Chapter 4, where the ability to correlate events for the purpose of network management strongly depends on how fast and how accurately pervasive entities in a computer network synchronize their time – synchronized time must be available on all entities participating co-operative correlation of events.

3.6 Background

Literature on time synchronization (Chapter 2, Section 2.2) lists many strategies available for time-synchronization in pervasive environments. Then, the options available would be to either develop a new strategy which competes with existing ones or to pick a suitable strategy 'off-the-shelf'. The main goal was to use a time synchronization strategy most suitable for event correlation in a pervasive environment. During the course of research, the authors observed that time-synchronization strategies of a particular class of algorithms fared better than others and therefore it was not necessary to create a new strategy [71]. Engineers wishing to corroborate particular features necessary for their implementation of a pervasive environment could choose from algorithms in this class – GS strategies. The choice of GS strategies of time-synchronization was arrived at by a comparison of GS and LS strategies. This comparison was based on a MANET environment – a typical extreme of pervasive environments. The evaluation was carried out with reference to convergence time – the time taken by the entire MANET to reach a time-synchronized state from ab-initio. This essentially translated to how fast the entire MANET could be time-synchronized.

During comparison, it was found that the suitability of a given strategy was highly dependent on MANET topology on which the comparison was carried out. Thus, the aim of the comparison was modified to determine upper bounds to synchronization efforts for particular MANET topologies. Thus, geometries that compel slowest time synchronization for a given strategy would be candidates for determining the upper bound to time synchronization.

3.7 Assumptions

The comparison of GS and LS strategies required certain assumptions to simplify mathematical analysis:

(a) Assumption A_1: A MANET has a seed node.

The research follows the beacon mechanism as described in IEEE 802.11 TSF [30]. The node which happens to be the first member of a MANET is defined as a seed node. This seed node instantiates beacons as part of its operating routine. The interval time between these beacons is known as Target Beacon Transmission Time (TBTT). Other nodes willing to participate in the MANET are required to conform to beacon periods as defined by the seed node. At each TBTT, all participating nodes contend for a beacon. This may result in beacon collisions – a performance factor that has been disregarded because it equitably affects both GS and LS strategies. Following contention, only one node manages to beacon in a broadcast area – typically defined by its radio range. Due to hidden terminal phenomenon [30], some node may receive two or more beacons in the same TBTT. Such cases are treated as collisions.

(b) Assumption A_2: MANET nodes conform to a spatially homogeneous distribution.

It is assumed that each participating node experiences the presence of equal numbers of neighbors regardless of its spatial position within the MANET. Thus, any node i has (n-1) neighbors, where, n is the unit size within a broadcast area. Furthermore, it is assumed that the MANET consists of N nodes, and, $N = |\text{MANET}| = m \cdot n$, $m \in \text{I+}$. Notwithstanding the strategy of time-synchronization, the only known method for a node to exchanging data regarding time is by periodic broadcast of a beacon.

(c) Assumption LS_1: For a given topology, LS time synchronization is considered complete when each participating node has broadcast a beacon at least once.

Participating nodes maintain a non-invasive local clock – one whose value is not changed by knowledge of external, and perhaps more precise and accurate clocks. Instead, the nodes maintain a vector containing time-differences between their local clocks and all participating node clocks within the broadcast area. This vector, henceforth called dT-vector, facilitates time inter-conversion between transacting node-pairs. (A_2) simplifies the problem formulation by allowing the dT-vector to maintain a fixed size of (n-1) entries, and therefore impose a constant working-memory cost on the node. Some LS strategies may violate (A_2) by implementing on-demand services to gain energy-efficient performance.

(d) Assumption GS_1: GS time synchronization is considered complete when each participating node has attained the fastest MANET clock.

In GS strategies, a MANET is considered time synchronized when the fastest time is dispersed throughout the MANET. Although an arbitrary clock can be chosen as a predefined standard, IEEE 802.11 specification [30] suggests the selection of the

fastest clock available among all participating nodes within an IBSS in ad-hoc mode. For a MANET, this implies selection of the fastest available clock among all participating nodes within a broadcast area, or, within the radio range of an air interface. Participating nodes maintain an invasive local clock – one whose value is changed every time a faster clock is detected within the broadcast area. The change can be carried out instantaneously, or spread over a period of time [31, 43].

3.8 Analysis

This section compares GS and LS strategies, in terms of convergence time, mobility, and stochastic time delay. The section is divided in three parts. The first sub-section compares the strategies from the perspective of convergence of time, the second compares them from the perspective of mobility, and, the third compares them from the perspective of stochastic time-delay.

3.8.1 Comparison with reference to convergence of time

Due to beacon contention, GS and LS strategies involve a fair degree of randomness, and an exact figure regarding time-convergence cannot be guaranteed. Instead, it is intended to determine the probability that the entire MANET has been synchronized by a particular time interval. This time interval is expressed in terms of TBTTs.

Let k denote the number of elapsed TBTTs. Then, GS probability that the entire MANET has been synchronized by k shall be compared to LS probability that the entire MANET has been synchronized by k. A higher probability of one strategy to be completely time-synchronized by k will lead to the conclusion that it performs better than the other in terms of time-convergence. Furthermore, (A_2) implies that each node has to contend with its (n-1) neighbors to beacon at each TBTT interval: 1, 2,…, k.

3.8.1.1 LS strategy

Given (A_1), (A_2) and (A_{LS1}), the entire MANET can be substituted by m IBSSs, each IBSS consists of n nodes. Then, synchronization of the entire MANET implies that each of the m IBSSs are also synchronized and, in turn, this implies that each participating node within each IBSS has beaconed at least once. The following analysis first considers probability of synchronizing time within an IBSS, and then generalizes it to apply throughout the MANET.

Let $P(i, k)$ denote the probability that i nodes of an IBSS have beaconed in k TBTTs. Since only one node can beacon at each TBTT,

$$P(i, k) = 0, \text{ whenever } k < i. \qquad (R_{LS1})$$

The argument behind (R_{LS1}) implies that n nodes of an IBSS need at least n TBTTs for each node to successfully beacon, and consequently satisfy A_{LS1}.

Trivially, $$P(1, 1) = 1, \text{ and } P(1, k) = \frac{1}{n^{k-1}}, k > 0. \qquad (R_{LS2})$$

The argument behind (R_{LS2}) is that there is always a node beacon at each TBTT within an IBSS. Moreover, it is rare that the same node beacons at each of k TBTTs.

As shown in Appendix I, lemma 1:

$$P(i, i) = \frac{i!}{(n-i)! \cdot n^i}. \qquad (R_{LS3})$$

In addition,

$$P(i, k) = P(i, k\text{-}1) \cdot \frac{i}{n} + P(n\text{-}1, k\text{-}1) \cdot \frac{n-i+1}{n}, \text{ when } i > 1. \qquad (R_{LS4})$$

The argument behind (R_{LS4}) is that the ith node can beacon in k TBTTs in only two ways – when i nodes have already beaconed in k-1 TBTTs and no new node will beacon at the next TBTT, or, when (i-1) nodes have beaconed in k-1 TBTTs. Then, only one of the rest (n-i+1) nodes (a new node) will beacon after failing to do so in the previous (k-1) TBTTs.

Using (R_{LS1}) through (R_{LS4}), individual probabilities $P(n, k)$ can be generated for all k. Then, the probability $P(N, k)$ that the entire MANET is synchronized after k TBTTs is:

$$P(N, k) = [P(n, k)]^m. \qquad (R_{LS5})$$

The argument behind (R_{LS5}) is that time-synchronization of each IBSS is independent of others.

3.8.1.2 GS strategy

(A_{GS1}) implies that the fastest time has been delivered through to the farthest participating node in a MANET. This is because the probability of the farthest node in a MANET to be time-synchronized will be lower than any of the intermediate participating nodes. Given a typical MANET, many message delivery routes may exist. Due to inherent nature of MANETs, time diffused along one route may influence and accelerate diffusion along other routes. To simplify the analysis of such a scenario, two geometries are considered – when participating nodes lie along a 1 dimensional line (case I), and, participating nodes lie along a ring (case II). All other geometries offer paths which will reduce convergence times. In both cases I and II, d denote the Cartesian distance between any pair of neighboring nodes, L denote the Cartesian distance between the node possessing the fastest time and the node farthest from it, and, hop distance h is defined as $h = ½(n\text{-}1)\cdot d$, so that using (A_2), a given hop will essentially cover $½(n\text{-}1)$ nodes on either side of the beacon node (by symmetry, (n-1) should be even). It may be observed in case I that the farthest node shall be at one end of the 1-dimensional line. This implies that (A_2) shall be violated at the farthest node. However, even if it is conservatively assumed that (A_2) does hold, the calculated probability of time synchronization will only be lower. Thus, the result from calculations in case I still provide a lower bound.

Case I: Given (A_2), consider a MANET with all participating nodes along a straight line (Figure 3). Let node A, the origin of the line possess the fastest time and node *B* be the

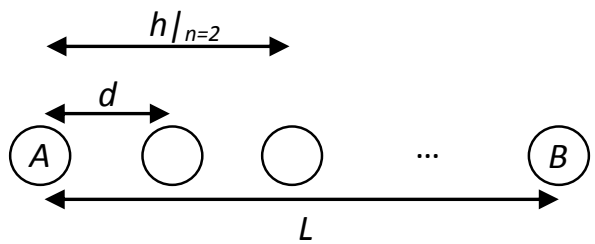

Figure 3: Describing relation between *d*, *h* and *L* for case I.

farthest node in the MANET. Then, the position of intermediate nodes *i* shall be integral multiples of *d*. Also, the farthest node B at distance *L* shall be as follows:

$$L = l \cdot d,\ l \in \mathrm{I}^{+}. \qquad (\mathrm{R}_{GS1})$$

The nodes are partitioned in two – those lying within the single-hop range *h*, of the beacon node, and, the outliers. Then, using (A_{GS1}), the probability $P(L, k)$ at which the node farthest from the fastest node is synchronized successfully within *k* TBTTs can be determined as follows:

By (A_1), (A_2), any node *i* will contend with *n*-1 neighbors to broadcast its beacon in a TBTT. So, within single-hop range *h*:

$$P(i \cdot d, 1) = \frac{1}{n}, \text{ for } i = 1, 2, \ldots, (n-1)/2, \text{ and,} \qquad (\mathrm{R}_{GS2})$$

for nodes outside the single-hop range,

$$P(i \cdot d, 1) = 0, \text{ for } i = (n+1)/2, (n+3)/2, \ldots, \infty \qquad (\mathrm{R}_{GS3})$$

For all nodes i lying within the single hop range h, from Appendix II:

$$P(i \cdot d, j) = 1 - \left(\frac{n-1}{n}\right)^j, \; i \leq (n\text{-}1)/2 \text{ and } j = 1, 2, \ldots, k,$$

and, for all nodes i lying outside the single-hop range h, Appendix II, lemma 2 gives:

$$P(i \cdot d, j) = P(i \cdot d, j\text{-}1) + \{P((i - \tfrac{n-1}{2}) \cdot d, j\text{-}1) - P(i \cdot d, j\text{-}1)\} \cdot \frac{1}{n},$$

$$i > (n\text{-}1)/2 \text{ and } j = 2, 3, \ldots, k. \quad (R_{GS4})$$

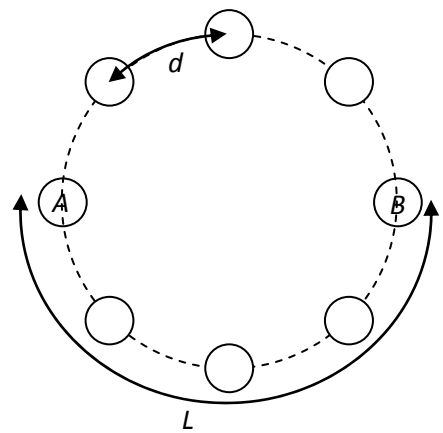

Figure 4: Describing relation between d and L for case II.

Case II: Extending the MANET geometry with all participating nodes to lie along a ring (Figure 4), the distance L shall correspond to the circumferential distance between two nodes lying along the diameter. Thus, (R_{GS1}) holds when the total number of nodes in this geometry is $2l$. Circular geometry facilitates analysis by presenting only two routes along which the fastest time can disseminate from one node to another. As a conservative simplification to calculate the probability that the farthest node is time-synchronized, the coupling effect of the two routes is only considered at the farthest node. Since two independent, symmetric routes exist for achieving time synchronization, the probability P_Θ (L, k) for successfully synchronizing time within k TBTTs with coupling effect is given by:

$$P_\Theta(L, k) = 2P(L, k) - P(L, k)^2 \quad (R_{GS5})$$

As mentioned above, (R_{GS5}) only provides a simple estimation; however, it suggests that as the number of routes between the pair of fastest node and farthest node increases, the probability of time-synchronization actually improves. In fact, a circular geometry presents worst possible MANET topology for GS time synchronization in 2-dimensions. Thus, the probability for any 2- or 3-dimensional MANET to be time-synchronized within k TBTTs will be higher due to increased numbers of alternative routes available for diffusing the fastest time.

3.8.2 Comparison with reference to mobility

3.8.2.1 Mobility and LS strategies

In an LS strategy, whenever a node receives the beacon from a neighbor, it measures its time difference with the neighbor, populating one element in the dT-vector. When the node receives beacons from all its neighbors, the dT-vector is fully populated, and the node is considered to be locally time-synchronized.

On the other hand, to diffuse a common time between any two nodes in a MANET, it is necessary to know the MANET's topology. A route between two given nodes must be probed firstly. Then, hop by hop time transformation can be carried out along the route. Hence, LS protocols must be aided by a routing protocol to synchronize time over the MANET.

Moreover, each dT-vector corresponds to given topology – a spatial synchronized state. Whenever the MANET topology changes, the state of time synchronization is lost, and dT-vectors at some or all nodes requires an update. Thus, a dynamic MANET topology may retard time synchronization when using LS strategies. This can be illustrated with the movement of a node in a set of k TBTT intervals. The k TBTT intervals can be split in two groups: T_1: 0 to k_1, and T_2: k_1+1 to k. In T_1, a node is considered to be in a region C with n-1

neighbors, and, in T_2, the node moves to a region D with another n-1 neighbors. Regions C and D must be different so that the node can be considered to have moved in terms of the topology. During T_1, the node receives beacons from m_1 nodes in region C. When the node moves to region D, the node shall receive beacons from another m_2 nodes. Given that,

$$n\text{-}1 < m_1+m_2 \leq 2(n\text{-}1), \qquad (R_{LS6})$$

the probability that the mobile node recovers its time-synchronized state at the end of T_2 can be calculated as two subsequent events – during T_1, n-m_2-1 nodes have beaconed, and, during T_2, m_2+1 nodes must beacon (the additional unit term, "1" of "m_2+1", refers to the mobile node itself). The probability of the mobile node recovering its time-synchronized state at the end of T_2 is:

$$\begin{aligned} P_{T2} &= P(n\text{-}m_2\text{-}1, k_1)\cdot P[(n, k) \mid (n\text{-}m_2\text{-}1, k_1)] \\ &= P[(n, k), (n\text{-}m_2\text{-}1, k_1)] \end{aligned} \qquad (R_{LS7})$$

As the event (n, k) and the event $(n\text{-}m_2\text{-}1, k_1)$ are independent,

$$P_{T2} = P(n, k)\cdot P(n\text{-}m_2\text{-}1, k_1) < P(n, k), \text{ for } P(n\text{-}m_2\text{-}1, k_1) < 1. \qquad (R_{LS8})$$

Thus, a node's mobility will retard MANET's time synchronization.

3.8.2.2 Mobility and GS strategies

GS has advantage over LS in terms of mobility – GS strategies would not be influenced by node mobility as only reference time (here, the fastest time) needs to be adopted within the MANET. In fact, a node's movement can accelerate distribution of reference time during synchronization. Following the analysis of a ring MANET (Case II), the probability that the farthest node is synchronized, $P(L, k)$, is only related to distance and elapsed time. Mobility does not affect farthest distance L and elapsed time k, making GS robust to mobility.

3.8.3 In terms of stochastic time-delay

In practice, GS strategies are not feasible without compensation for lack of precision. At each beacon, precision errors, i.e., stochastic time-delay, will accumulate as distances and elapsed times increase, resulting in distortion of reference time. In fact, a MANET's reference time will never converge to fastest time, but will drift with accumulation of precision error. On the other hand, LS strategies are robust to precision errors. Even through every node still needs to beacon its time, every receiver would not update its own time, but only record the offset to update its dT-vector – precision errors may affect the dT-vector, but they would not accumulate.

3.9 Simulation

In this section, theoretical results and simulation regarding the comparison of GS and LS are presented. As node density is given, MANET topology would not affect convergence time of LS. However, for GS, ring topology is considered.

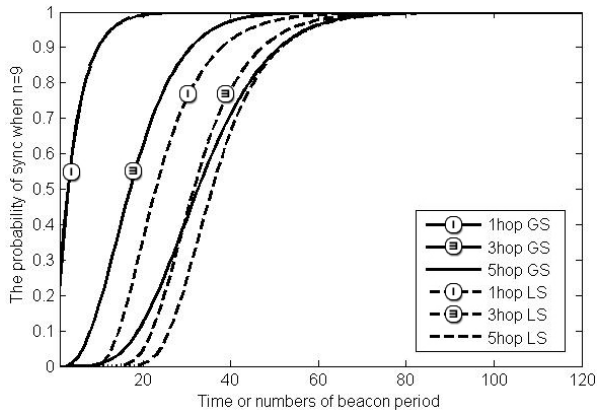

Figure 5: Convergence time vs. probabilities when n = 9.

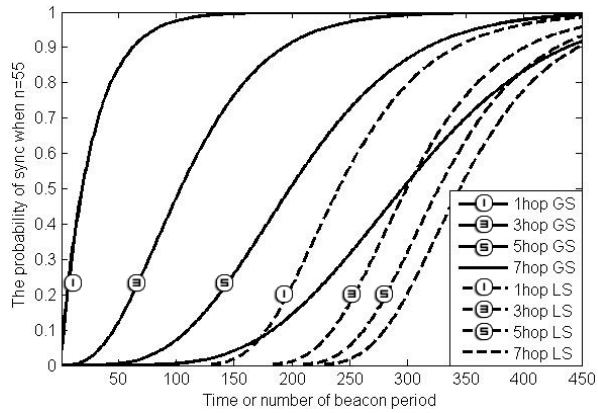

Figure 6: Convergence time vs. probabilities when $n = 55$.

By analyzing Figure 5 and Figure 6, it is observed that when $n > 9$ and $h < 5$, GS performance is better than the LS with reference to convergence time. It shows that GS performs better than LS, even if the MANET accommodates 7 hops, so long as node density is high ($n \geq 55$).

GS has advantage over LS when nodes are mobile. If the participating nodes are synchronized to the fastest time, time-sync state will not be influenced by mobility. In fact, mobility will actually promote the distribution of fastest time. On the other hand, LS will suffer due to mobility as a large overhead is involved when participating nodes have to adjust their dT-vectors to accommodate for new neighbors.

The simulation was carried out using network simulation software ns-2 [41], within a rectangular region of 500m × 500m. The broadcast radius for each node was chosen to be the ns-2 default (250m). Thus, the maximum possible hop count is 3 (along the region's diagonal). It is apparent from Figure 7 that GS strategies achieve faster convergence.

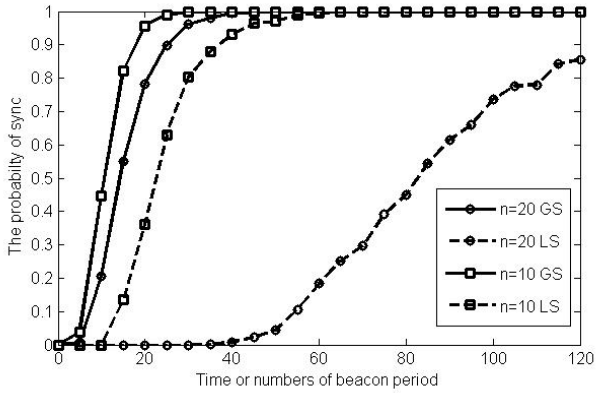

Figure 7: Convergence probabilities in 500m × 500m

3.10 Conclusion

This research presents a theoretical basis for comparing GS and LS time-synchronization strategies. It has been pointed out that GS strategies have guaranteed faster convergence probabilities than LS strategies whenever node density > 9 nodes/broadcast area, and maximum hop count of the MANET is < 5. Such a result proves useful for application to event-based pervasive network management systems discussed in Chapter 4 where these bounds present minimum time required before any correlation based on events is carried out.

CHAPTER 4

4. EXPLOITING EPISTEMIC UNCERTAINTY

What's in a ~~name~~ theory?
- Shakespeare
The devil lies in ~~details~~ computer networks.
- Anonymous

4.1 Introduction

This chapter delves into application of Dempster-Shafer's theory to support a high-volume, event-based, in-network and non-deterministic pervasive network management. First, an argument for providing network management functionality based on classical probability is presented. Then, due to its inherent drawbacks, a second argument supporting Dempster-Shafer's theory for the same functionality will be presented. Following the description of the model and its assumptions, simulation data verifying the model's applicability conclude the chapter.

4.2 Candidate 1: Bayesian theory

The advantage offered by Baye's theorem lies in its ability to determine probabilities that are causally 'inverse'. In other words, it is possible to determine the probability of an earlier event given that another event is known to have occurred later on in time. From the perspective of network management, this entails changes in notation for easier understanding. Conditional probability will be notated in terms of events, E, and probable causes, PC, so that, the probability P of occurrence of probable-cause PC_i given that the event E has already occurred is given by:

$$P(PC_i|E) = \frac{P(E \cap PC_i)}{\sum_j P(E \cap PC_i)} \qquad (R_{BT1})$$

Values on right hand side (RHS) of (R_{BT1}) are generally obtained experimentally. As sample-size *j*, which denotes number of entities at which correlation between *E*, and PC_i is known increases, conditional probability that *E* is due to PC_i will be more accurate. Enumeration *i* denotes the universe of probable-causes which can afflict the network.

(R_{BT1}) does not lend itself well to event composition – it does not have a form that can be used to preserve information content of events as they pass through the network, or, under some special circumstances, preserve information content under certain correlation transformations such as generalization and suppression. In fact, the form is no different from traditional apriori methods such as decision trees and codebook/correlation matrix methods. However, Bayesian theory offers leverage by considering the concept of partial event set.

Given correlation transformations, K_i, such transformations can be denoted diagrammatically as follows (Figure 8):

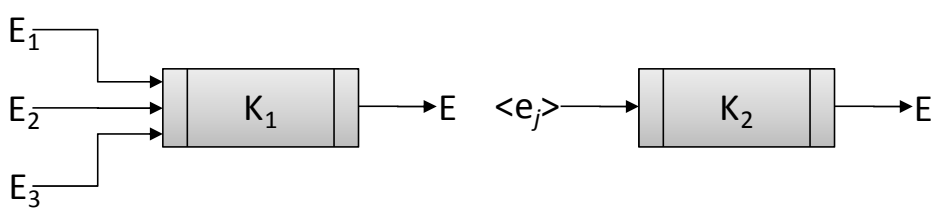

Figure 8: Correlation transformations (K_i).

In previous research, it is implicitly assumed that correlation transformations, K_i, are lossless. In other words, it is assumed that information content is completely preserved under such transformations. However, the assumption does not hold true when particular members of the partial event set, e_j, are missing, false, delayed or corrupt. Using Bayesian theory,

correlation transformations, K_i, can calculate conditional probabilities $P(E|e_j)$ for each e_j, and whenever all e_i are known, $P(E|e_j) = P(E)$, defined implicitly in regular correlation transformations.

Extending (R_{BT1}) using knowledge of partial event sets [19]:

$$P(PC_i|e_j) = \frac{P(PC_i \cap e_j)}{P(e_j)} \qquad (R_{BT2})$$

Since e_j may not contribute to E for some value of j:

$$P(PC_i|e_j) = \frac{P(PC_i \cap E \cap e_j) + P(PC_i \cap E' \cap e_j)}{P(e_j)} \qquad (R_{BT3})$$

Applying Bayesian theorem again:

$$P(PC_i|e_j) = \frac{P(PC_i \cap E \cap e_j) \cdot P(E \cap e_j) + P(PC_i \cap E' \cap e_j) \cdot P(E \cap e_j)}{P(e_j)} \qquad (R_{BT4})$$

This can further be written as:

$$P(PC_i|e_j) = P(PC_i|E \cap e_j) \cdot P(E|e_j) + P(PC_i|E' \cap e_j) \cdot P(E'|e_j) \qquad (R_{BT5})$$

(R_{BT5}) can be solved but requires lot of computation at devices and equipment responsible for processing events – processing event properties for E and each instance j of e_j. If design of correlation transformations, K_i, is such that if missing, false, delayed or corrupt instances of e_j are sparse in number, they are ignored, and then, (R_{BT5}) can be reduced to a cascading conditional form:

$$P(PC_i|e_j) = P(PC_i|E) \cdot P(E|e_j) + P(PC_i|E') \cdot P(E'|e_j) \qquad (R_{BT6})$$

(R_{BT6}) is simple but must be used with caution – there are no guarantees regarding apriori knowledge of missing, false, delayed or corrupt e_j, even though they are considered apriori by causal considerations alone.

Architecturally, (R_{BT6}) can be envisioned in the network as shown in Figure 9. Each K_i (here, i = D, F) shall calculate $P(E|e_j)$ and $P(E'|e_j)$, and pass this information to K_A where E_i will be used to obtain a probable cause PC_A.

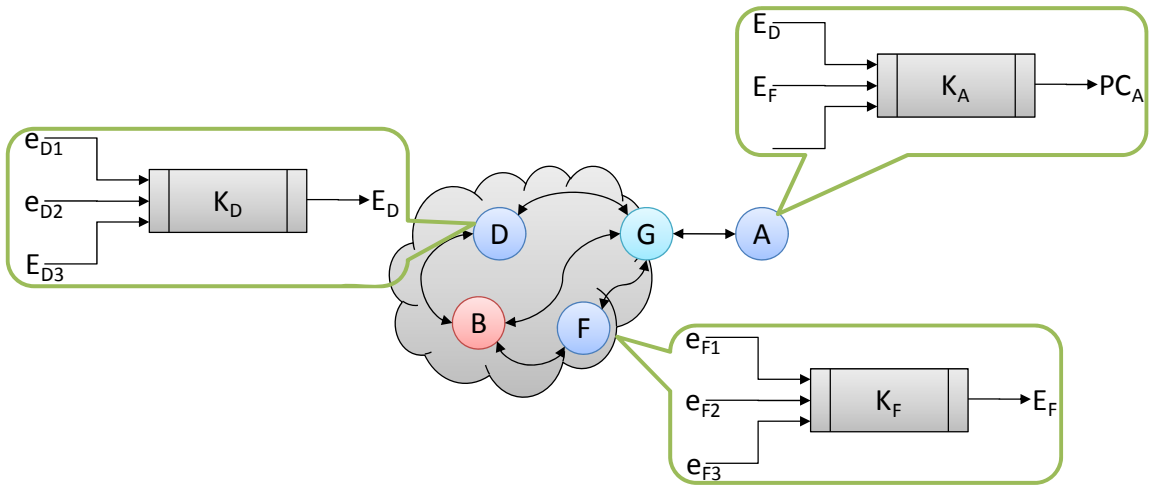

Figure 9: Architecture showing correlation transformation.

While architecture presented in Figure 9 is suitable for distributed network management, it still requires full enumeration of symptoms and probable causes to allow management entities such as A, D, and F to correctly determine causes ("C" in ECA) and consequent actions ("A" in ECA) suitably.

4.3 Candidate 2: Dempster-Shafer's theory

The advantage offered by Dempster-Shafer's theory is ability to assign a degree of belief to events with regards to probable-causes, as they traffic through a network. Applying Dempster-Shafer's theory to pervasive network management, it is found that assigning a belief of say 10% to a particular event E_j as being a symptom for a probable-cause PC_i does not

mean a disbelief of 90% to E_j as being a symptom for PC_i. In other words, if the network management system believes that once it observes the presence of event E_j, there is a 10% chance that a problem can be attributed to probable-cause PC_i, then, it does not automatically mean that there is a 90% chance that the problem cannot be attributed to probable-cause PC_i. Therefore, to accommodate such scenarios, a degree of belief is assigned to events. This is customarily known as mass of the events, denoted by m. Then, each event E_j, or a sequence of events $<e_j>$, each of which may be a symptom of a probable cause PC_i, are all assigned a finite mass m, to denote a degree of belief in the events as being symptoms for probable causes PC_i. Any event E_j or sequence of events $<e_j>$ which do not contribute to PC_i for any i, are not assigned any mass. All events E_j and sequences of events $<e_j>$ are considered to be elements of a mutually exclusive and exhaustive set known as 'frame of discernment', denoted by M.

The set M, its elements, or its subsets may be mapped to each probable cause PC_i to denote that the set M, its elements, or its subsets are actually symptoms of the probable causes PC_i. Mathematically, this mapping is nothing but the power set of M, denoted by $\pi(M)$:

$$\pi(M) = \{\varphi, \{E_1\}, \{E_2\}, \ldots, \{< e_j >\}, \ldots, \{E_1, E_2\}, \ldots \{E_{j-1}, < e_j >\}, \ldots, \{E_1, E_2, \ldots, < e_j >, \ldots\}\} \tag{R_{DS1}}$$

According to Dempster-Shafer's theory, elements of $\pi(M)$ may have a Real mass in the continuous interval [0, 1], and the sum of masses of all elements in $\pi(M)$ is unity:

$$m|\pi(M) \longrightarrow [0,1] \tag{R_{DS2}}$$

$$\sum_{x \in \pi(M)} m(x) = 1 \tag{R_{DS3}}$$

Each correlation transformation K_i can define its own degrees of belief to each event or sequence of events encountered at K_i:$\{m_{Ki}\}$. As events traffic through different parts of a network, various correlation transformations can orthogonally combine their masses using Dempster's Rule of Combination:

$$m_{K1} \oplus m_{K2} \oplus \cdots \oplus m_{Ki}(x_1 \cap x_2 \cap \cdots x_i)$$

$$= \frac{\sum_{x_1 \cap x_2 \cap \cdots x_i \neq \varphi} m_{K1}(x_1) \cdot m_{K2}(x_2) \cdots m_{Ki}(x_i)}{1 - \sum_{x_1 \cap x_2 \cap \cdots x_i \neq \varphi} m_{K1}(x_1) \cdot m_{K2}(x_2) \cdots m_{Ki}(x_i)} \qquad (R_{DS4})$$

If the output of a certain correlation transformation K is channeled to another correlation transformation G, then, the masses shall combine as:

$$m_{G1} \oplus (m_{K1} \oplus m_{K2} \oplus \cdots \oplus m_{Ki}(x_1 \cap x_2 \cap \cdots x_i))$$

$$= \frac{\sum_{y_1 \cap (x_1 \cap x_2 \cap \cdots x_i) \neq \varphi} m_{G1}(y_1) \cdot \sum_{x_1 \cap x_2 \cap \cdots x_i \neq \varphi} m_{K1}(x_1) \cdot m_{K2}(x_2) \cdots m_{Ki}(x_i)}{1 - \sum_{y_1 \cap (x_1 \cap x_2 \cap \cdots x_i) \neq \varphi} m_{G1}(y_1) \cdot \sum_{x_1 \cap x_2 \cap \cdots x_i \neq \varphi} m_{K1}(x_1) \cdot m_{K2}(x_2) \cdots m_{Ki}(x_i)} \qquad (R_{DS5})$$

While the form of (R_{DS5}) may appear complicated, it is actually amenable for application to a distributed system scenario. Events which form evidence in a correlation transformation K, are not recomputed in correlation transformation G. This "consensus" operation, which is a property of orthogonal sums, provides a strong mathematical support to many important network management processes, prominently, reduction in computation requirement, and reduction in bandwidth for management traffic. Additionally, this operation also introduces anonymity – a feature which may be desirable for democratic contexts in pervasive computer networks, but is subject to debate. This research will provide credibility to all these claims in later sections.

4.4 Assumptions

The following are the assumptions governing the pervasive modeling environment:

(a) Assumption A_1: The pervasive environment is time synchronized.

A necessary assumption for accuracy of correlated events is that every participating managed, managing, or proxy entity is aware of a global or a relative time which is constant throughout the network. Although Dempster-Shafer's theory does not imply a component of time, this assumption generalizes correlation over contextual domains.

(b) Assumption A_2: The pervasive environment implements routing.

Analyses of network management and associated features assume the presence of a routing mechanism within the pervasive mechanism. The analyses does not rely on contextual route changes accruing to the process of management itself, however, security and trust in routing mechanisms may be integrated into the management process to minimize the effect of collusion and therefore enhance network management outreach (Chapter 5).

4.5 Model

Pervasive environments have unique characteristics which must be exploited for a network management system to successfully manage such computer networks:

(a) In-network processing – Since pervasive environments typically contain resource constrained entities, energy efficiency is gained by distributed computing, whether it be for local consumption or peer support. This must also be applicable to data used for network management. Management traffic flowing within the computer network must be of a form which can be processed in a distributed manner.

(b) High event volume – A high entity density typically results in a high volume of generated events. More often than not, an unexpected fulmination of events, known as 'event storms' may occur if distributed event-based management is applied to pervasive environments as is. Thus, network management paradigms for pervasive environments should be able to handle large volumes of events, often modified on-the-fly by correlation along contextual dimensions, heavily dependent on topology of the computer network, containing conflicting information and dealing with rapidly changing contexts while producing results with low-latency.

(c) Relay – A centralized network management paradigm dictates that the computer network must have a single point of control. While this requirement is necessary for managing the network per se, its functional architecture is relaxed for distributed processing of management information. This implies that even though management information is aggregated at various points within the network, the information must be relayed to a central point. This 'manager' entity, which is the recipient of this summarized information, may be equated to the sink in traditional wireless sensor/actuator networks. Thus, just like the case of wireless sensor/actuator networks, while the relay may be arbitrary in length, and span different communication media, there are associated drawbacks. Increase in relay lengths typically increases the energy consumption of the pervasive environment, increases data latency and increases communication failure rates accruing to dropped packets.

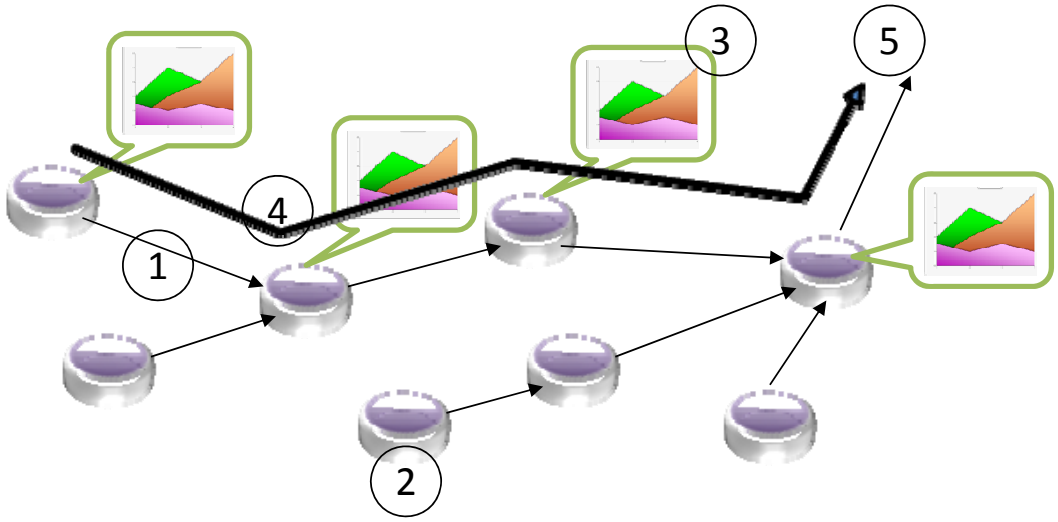

Figure 10: Typical pervasive environment modeled as Wireless Sensor/Actuator Network

Consequently, management channels dependent on the relay must reduce bandwidth consumption while maintaining network management functionality.

(R_{DS5}) can be directly applied to a scenario as shown in Figure 10 so that it can be modeled as shown in Figure 9. As a key, '1' denotes management traffic, '2' denotes participating managed entities, '3' denotes summarized management data, '4' denotes management traffic flow, and, '5' denotes upstream traffic to a manager entity. Masses of events E_j, collected at each participating managed entity j, can be combined orthogonally at each entity in the relay until the aggregate reaches the 'manager' (sink). This method provides us with two advantages:

(a) Management data is summarized at each entity in the relay – While this is the main advantage offered by Dempster-Shafer's theory, it also affirms in-network processing which is an important characteristic of pervasive environments. The technique used for summarizing network management data may include correlation along contextual dimensions.

(b) Bandwidth required for management data is lowered – This affirms the second requirement of pervasive environments, that is, to mitigate the increase in energy consumption and communication failure rates due to long relay paths.

The simulations based on this model are described next.

4.6 Simulation

The two main goals of simulations are:

(a) To demonstrate that Dempster-Shafer's theory is applicable and appropriate means for pervasive network management, and,

(b) To demonstrate that bandwidth required for management traffic is significantly lower when supplementing event correlation with Dempster-Shafer's theory.

These goals are demonstrated in a scenario explained as follows:

The simulation environment consists of a random collection of participating managed entities which relay packets of fixed size. Each packet consists of belief assigned to reasons considered as possible causes of various symptoms observed by each participating entity. Depending on their individual state, symptoms observed, and preferences, participating entities may or may not assign beliefs to particular probable causes (therefore, non-deterministic). The entities also query their neighbors regarding the status of particular internal processes (in-network processing). These queries influence individual belief assignments; however, they do not accrue towards the management bandwidth consumed by the network ('consensus' operation). The simulation records assigned beliefs at each entity in a relay of up to 10 hops. Based on results from Chapter 3, it is assumed that relay lengths are generally confined within this hop range.

Figure 11 shows a typical scenario where network operation is 'normal' and no participating entity can pin-point observed problems in the network. As a key, 'U' denotes an unknown cause, 'R_i' denotes probable cause *i*, and, 'Total' denotes aggregate belief in a normal pervasive environment.

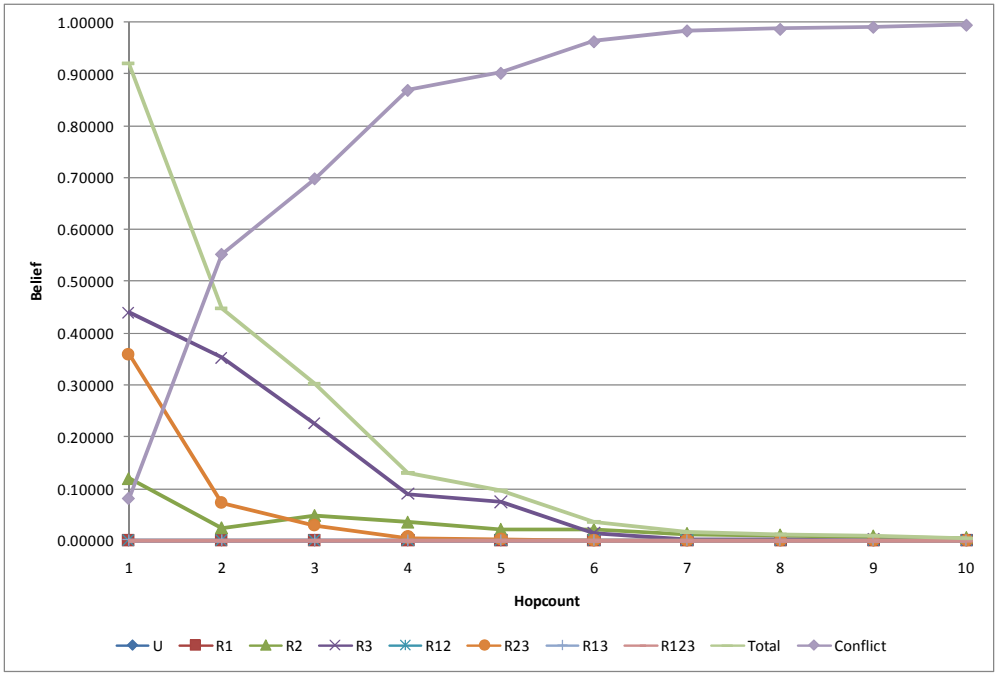

Figure 11: Changes in belief and conflict in probable causes as a function of hop-count.

When no participating entity in a given relay presents with consistent agreement on a given probable cause, the overall 'conflict' of belief rises along the relay. Notwithstanding the initial beliefs on individual probable causes, or their power-set combinations, the belief gradually diminishes along the relay length – this implies that without significant agreement regarding probable cause of a symptom or a set of symptoms, the overall belief is significantly eroded as more and more conflict is observed among entities along the length of the relay.

When a given probable cause (here reason R_2) does show agreement among a group of entities, the belief in the probable cause shows a marked consistency denoting general agreement regarding a network problem. In the simulation, a critical service was deliberately shut down in a portion of the network. Entities lying in that particular portion of the network (here entities within hop 4 through 7 in Figure 12) increased overall belief in a probable cause R_2. Since the probable cause is the only major source of symptoms observed in the network, its belief value approaches that of the total belief of the relay. The consistency in belief towards a single probable cause also stemmed the overall conflict observed within the network for the duration of hops lying within the affected region (hops 4 through 7 in Figure 12).

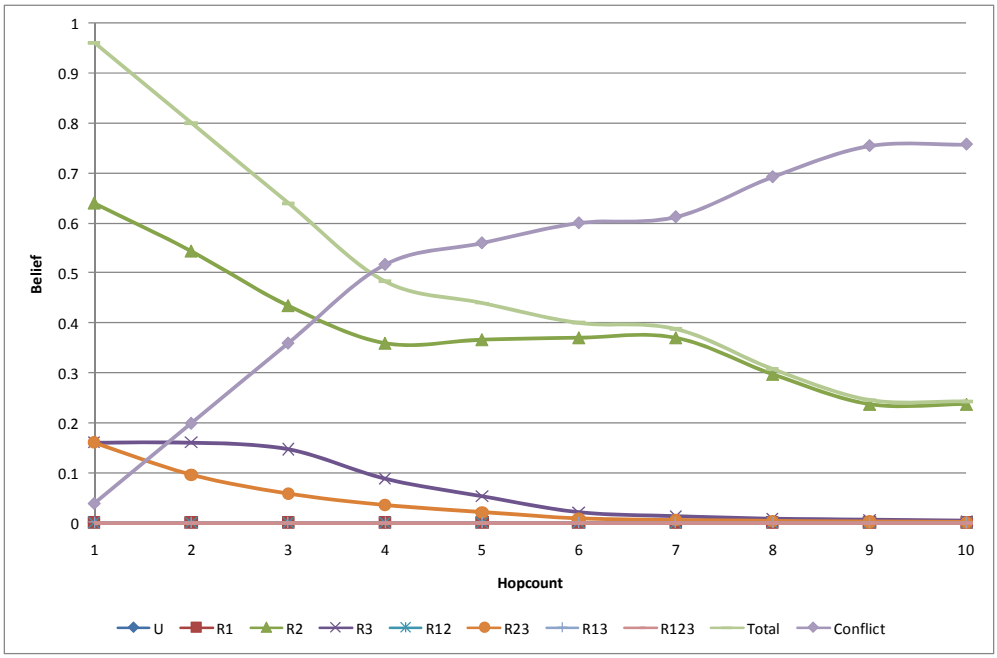

Figure 12: Changes in belief in an unusual pervasive environment.

In simulation for Figure 13, a Firewall is deliberately used to partition the network (here after hop 6). Entities outside the Firewall (here hops 1 through 6) are exposed to a viral attack which consistently flags a particular probable cause (here reason R_2). This is elucidated by a consistent belief in the reason. Since entities within Firewall's perimeter are not affected by the particular attack, the nodes show a disagreement, and therefore significant erosion in the belief on probable cause R_2 is observed. This is complemented by marked increase in conflict regarding overall belief. As before, since probable cause R_2 is the only major source of symptoms observed in the network, its belief value approaches that of total belief of the relay.

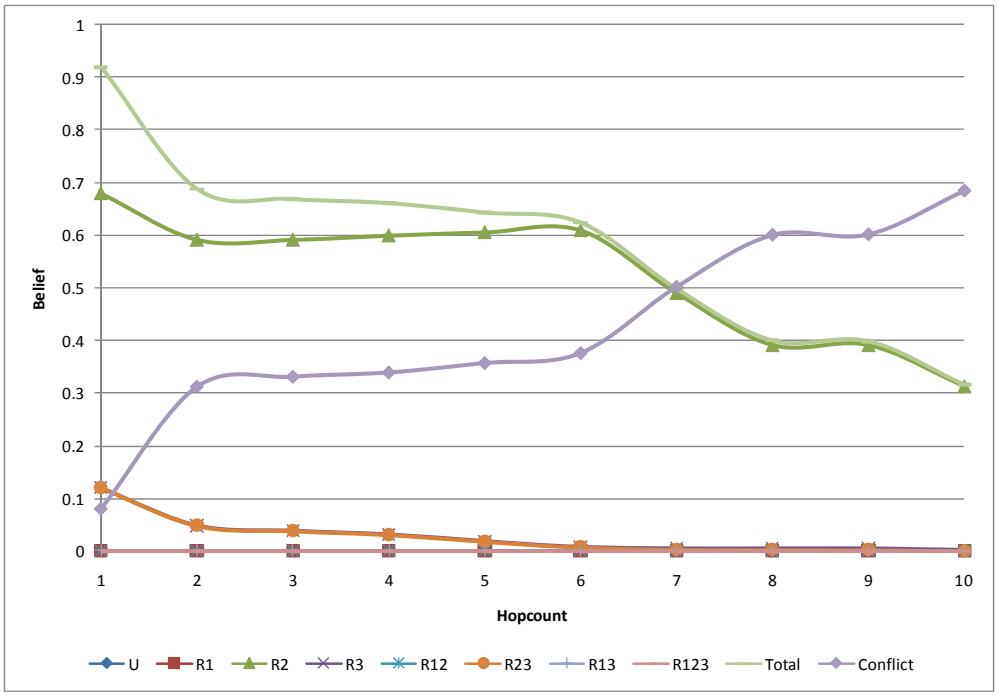

Figure 13: Changes in belief in a pervasive environment partitioned by a Firewall.

The following simulations (Figure 14 and Figure 15) show average bandwidth consumption for pervasive network management based on Dempster-Shafer's theory versus a popular network management protocol: SNMP. Since pervasive network management employs in-network processing, overall bandwidth required is significantly lower as relay length increases. However, for relay lengths that are typically small (within single-digit range), the consumption is not significant.

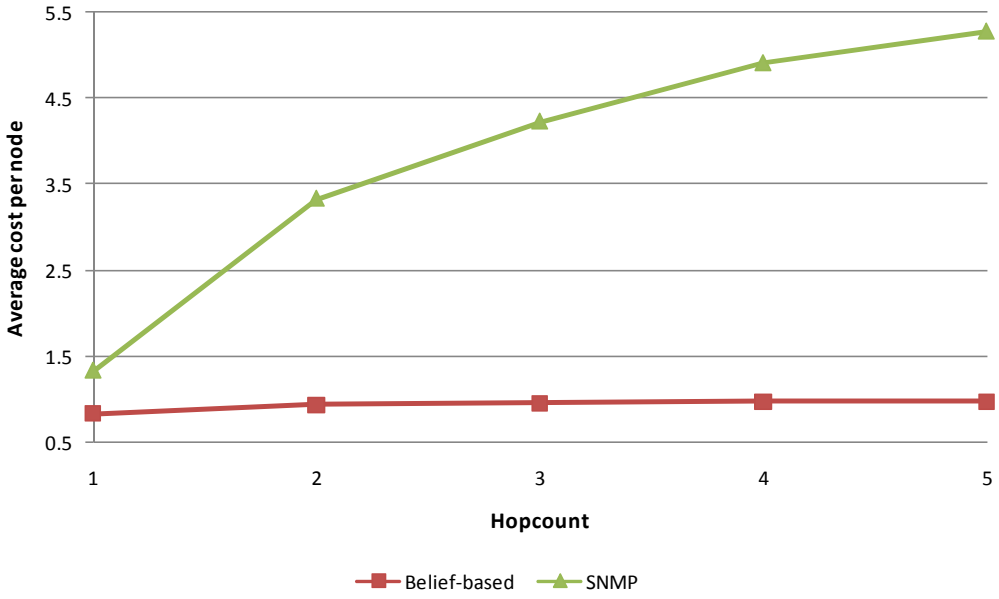

Figure 14: Average increase in message bandwidth for an edge manager entity.

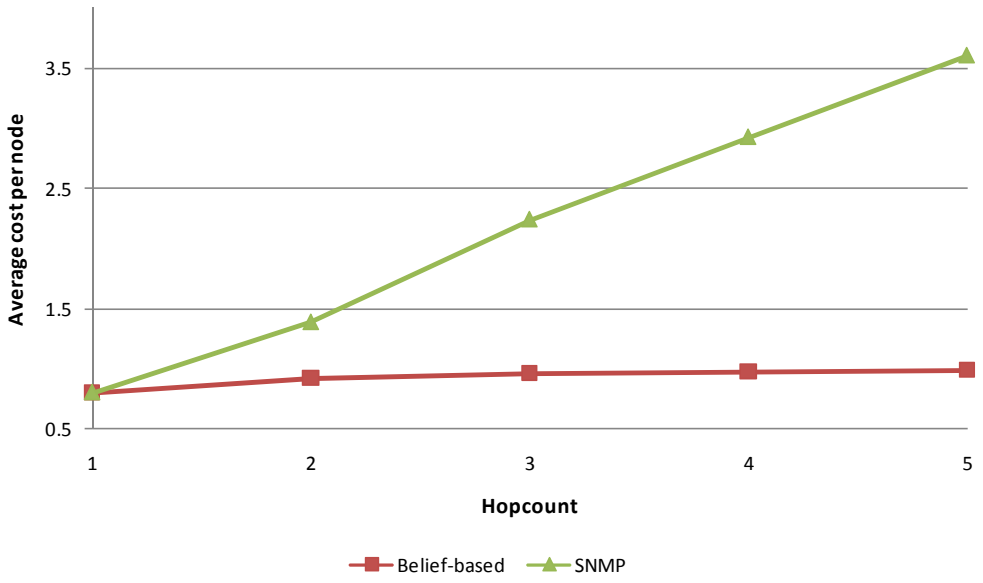

Figure 15: Average increase in message bandwidth for a central manager entity.

4.7 Drawbacks – Location awareness

While this research brings forth a new event-based pervasive network management scheme based on a lightweight scheme exploiting epistemic uncertainty – Dempster-Shafer's theory, it suffers from a major drawback as illustrated in Figure 12 and Figure 13 – location awareness. Due to missing information regarding location awareness, probable causes deemed as major afflictions in a portion of the computer network can only be noticed by manager entities within or adjacent to that portion of the computer network. This is true even if summarization of network management data does consider spatial correlation – outside the range of interest, conflict with other probable causes quickly erodes the belief in the major probable cause. For example, in Figure 12, belief that reason R_2 is a dominant probable cause in the region of the computer network occupied by entities in the hops 4 through 7 is already eroded significantly by the 9th hop.

4.8 Conclusion

This research presents us with a high-volume, event-based network management scheme suitable for pervasive computer networks. The scheme is based on Dempster-Shafer's theory. Bandwidth required for management traffic is significantly lower when supplanted with in-network processing. In-network processing adds computation overhead to participating entities. However, distribution of overhead allows network management response less latent – no single point offers a processing bottleneck. The scheme withstands missing and conflicting information gathered from multiple participating entities and summarizes management data along contextual dimensions. Anonymity is maintained – it is difficult to trace a probable cause to an entity or a group of entities. However, advantages of this feature may be debatable for environments where privacy preservation makes forensic operations difficult.

CHAPTER 5

5. CONCLUSIONS

We shall not cease from exploration
And the end of all our exploring
Will be to arrive where we started
And know the ~~place~~ pervasive computer network for the first time.
– T.S. Eliot

5.1 Contribution

The dissertation presents a novel application of Dempster-Shafer's theory to high volume, event-based, in-network, network management of pervasive computer networks. Additionally, it provides analytical bounds of time synchronization after which such an application can be exploited.

(a) First, currently available algorithms for synchronizing time in a pervasive computer network are split in two classes – those which employ an invasive clock and those which do not. It is shown that for a given lower bound of entities participating in a network, one class of algorithms, namely, those relying on an invasive clock – global time synchronization (GS), is able to achieve a faster time convergence, and therefore, are better prepared to support an event-based network management system.

(b) Second, a novel application of Dempster-Shafer's theory as an appropriate means for high-volume, event-based, network management in pervasive computer networks is presented and demonstrated. It elucidates an effective method of in-network processing, coupled with low bandwidth consumption – two important features for pervasive network management systems. Since only very little computation is involved

at each participating entity, the application is well-suited for a pervasive computer networks, prevents a processing bottleneck in the network, and therefore provides low-latency response. The theory itself withstands missing and conflicting information gathered from multiple participating entities, and supports summarizing of network management data at participating entities along contextual dimensions.

5.2 Future Work

5.2.1 Location Awareness

Although a lightweight scheme for pervasive network management, this novel application has its drawback – its blindness towards location awareness. One of the first objectives of the future work is to remove this drawback without compromising on the advantages gained by the application. While this is possible by considering certain entities to be aware of the topology of their neighborhood, preliminary results indicate that the choice of such entities is an NP-hard problem. Surprisingly, the insight does offer promise that the relationship between possibility of agreement and bandwidth consumed for an algebraically increasing number of location aware nodes is simple (Figure 16).

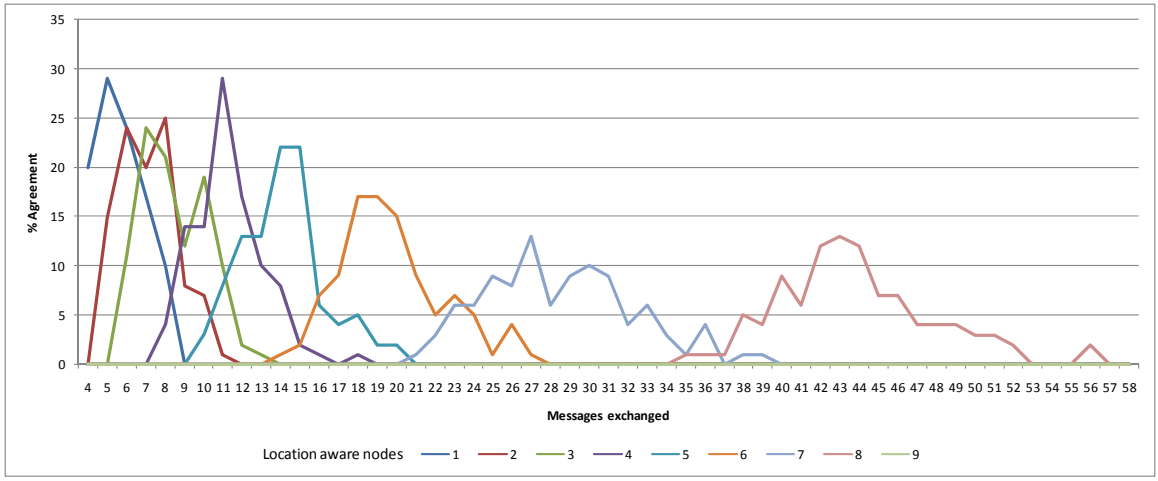

Figure 16: Relation between agreement and bandwidth consumed for location aware nodes.

5.2.2 Security and Trustworthiness

As stated in the assumptions governing Chapter 4, security and trustworthiness are considered given by the underlying routing mechanism. This implicit faith in the system can blind the network management scheme towards vulnerabilities exploiting the routing mechanism [60]. As the network management shall rely on the routing mechanism to provide it with security and trustworthiness primitives, a malicious intent which succeeds in subverting the routing mechanism can effectively redirect network management traffic into obfuscated sections of the computer network. This is complicated by the fact that wireless technologies essentially use a shared medium for communication. Thus, management traffic flowing from and through wireless devices and equipment faces an extra level of trustworthiness and security complexity.

Research being conducted by author's major advisor indicates that this entails the development of a comprehensive trust platform that ties a policy-based approach to a behavioral model. The platform will enable detection and isolation of entities that breach the platform, which itself is actually a network management function. Tying the network management scheme to the platform will allow it to be independent of externally provided trust and security primitives.

In sum, the future of this research would be to introduce location awareness and tie network management to a comprehensive trust platform providing integration of security and trust primitives to network management itself.

LIST OF REFERENCES

[1] Abowd, G. E. and Mynatt, E. D., "Charting Past Present and Future Research in Ubiquitous Computing", ACM Transactions on Computer-Human Interaction, Special issue on HCI in New Millennium, vol. 7, no. 1, pp. 29-58, February 2000.

[2] Albaghdadi, M., Briley, B., Evens, M., Sukkar, R., Petiwala, M. and Hamlen, M., "A Framework for Event Correlation in Communication Systems", In Al-Shaer and Pacifici, eds., "Management of Multimedia on the Internet", vol. 2216, pp. 271-284, Springer-Verlag, 2001.

[3] Appleby, K., Goldszmidt, G. and Steinder, M., "Yemanja - A Layered Fault Localization System for Multi-Domain Computing Utilities", International Journal of Network and Systems Management, vol. 10, no. 2, pp. 171-194, Plenum Press, NY, 2002.

[4] Aprisma Spectrum network manager is now CA Spectrum network manager, last accessed on March 1, 2008 at http://www.ca.com/us/products/product.aspx ?id=7832

[5] Balakrishnan, H., Kaashoek, M. F., Karger, D., Morris, R. and Stoica, I., "Looking up data in P2P Systems", Communications of the ACM, Special issue: Technical and social components of peer-to-peer computing, vol. 46, no. 2, pp. 43-48, February 2003.

[6] Balasubramanian, J. S., Garcia-Fernandez, J. O., Isacoff, D., Spafford, E. H. and Zamboni, D., "An Architecture for Intrusion Detection using Autonomous Agents", 14th Annual Computer Security Applications Conference, pp. 13-24, 1998.

[7] Bass, T., "Event Stream Processing versus Complex Event Processing", Inaugural International Conference on Distributed Event-Based Systems (DEBS), Toronto, Canada, June 2007.

[8] Chen, T. M. and Venkataramanan, V., "Dempster-Shafer theory for Intrusion Detection in Ad Hoc Networks", IEEE Journal of Internet Computing, vol. 9, no. 6, pp. 35-41, Nov. 2005.

[9] Cisco, Inc., "Basics of Network Management", last accessed on February 12, 2008 at http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/nmbasics.htm

[10] computer. (2008). In Encyclopedia Britannica. Retrieved January 16, 2008, from Encyclopedia Britannica Online: http://www.britannica.com/eb/article-216039

[11] Dai, H. and Han, R., "Tsync: A lightweight bidirectional time synchronization service for wireless sensor networks", ACM SIGMOBILE Mobile Computing and Communications Review, vol. 8, no. 1, pp.125-139, 1994.

[12] Daniele, M., Wijnen, B., Ellison, M. and Francisco, D., "Agent Extensibility (AgentX) Protocol version 1", The Internet Society, Internet Engineering Task Force RFC, January 2000.

[13] Deeter, K., Singh, K., Wilson, S., Filipozzi, L. and Vuong, S., "Aphids: A Mobile Agent-based Programmable Hybrid Intrusion Detection System", Lecture Notes in Computer Science, vol. 3284, pp. 244-253, Springer-Verlag, 2004

[14] Elson, J., Girod, L. and Estrin, D., "Fine-grained Network Time Synchronization using Reference Broadcasts", in SIGOPS Operating Systems Reviews, vol. 36, pp. 147-163, Dec. 2002.

[15] Elson, J. and Römer, K., "Wireless sensor networks: a new regime for time synchronization", ACM SIGCOMM Computer Communications Review, vol. 33, no. 1, pp. 149-154, Jan. 2003.

[16] Esper Event Stream Processor, last accessed on March 1, 2008 at http://esper.codehaus.org/

[17] Galton, A., "Temporal Logics and Their Applications", Academic Press, ISBN13: 978-0122740602, March 1988.

[18] Ganeriwal, S., Kumar, R., Adlakha, S. and Srivastava, M. B., "Network-wide Time Synchronization in Sensor Networks", NESL Technical Report, 2003.

[19] Giarratano, J. C. and Riley, G., "Expert Systems: Principles and Programming", Thomson Learning, Inc., 4th Edition, ISBN13: 978-0534384470, October 2004.

[20] Graham, S., Kumar, P. R., "Time in general-purpose control systems: the Control Time Protocol and an experimental evaluation", 43rd IEEE Conference on Decision and Control (CDC '04), vol. 4, pp. 4004-4009, Dec. 2004.

[21] Grer, D. W., Khan, I., Ogier, R. and Keffer, R., "An Artificial Intelligence Approach to Network Fault Management", SRI International, 1996.

[22] Griffith, R., Hellerstein, J. L., Diao, Y. and Kaiser, G., "Dynamic Adaptation of Rules for Temporal Event Correlation in Distributed Systems", Columbia University Technical Report CUCS-003-05, New York, January 2005.

[23] Guofei, J. and Cybenko G., "Temporal and Spatial Distributed Event Correlation for Network Security", American Control Conference, vol. 2, pp. 996-1001, IEEE Computer Society, June 2004.

[24] Gupta, M. and Subramanian, M., "Preprocessor Algorithm for Network Management Codebook", 1st ACM Conference on Intrusion Detection and Network Monitoring, Santa Clara, CA, 1999.

[25] Harrington, D., Presuhn, R. and Wijnen, B. "An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks", The Internet Society, Internet Engineering Task Force RFC, December 2002.

[26] Hasan, M., Sugla, B. and Vishwanathan, R., "A Conceptual Framework for Network Management Event Correlation and Filtering Systems", IEEE/IFIP International Symposium of Integrated Network Management, pp. 233-246, 1999.

[27] HP Openview (now rechristened under brand name "HP Software"), last accessed on February 12, 2008 at https://h10078.www1.hp.com/cda/hpms/display/main/hpms_content.jsp?zn=bto&cp=1-11-85_4000_100__

[28] Holger, K. and Willig, A., "A Short Survey of Wireless Sensor Networks", Technical Report, Telecommunication Networks Group, Technical University of Berlin, 2003.

[29] Hu, A. and Servetto, S. D., "Algorithmic Aspects of the Time synchronization Problem in Large-Scale Sensor Networks", Mobile Network Applications Conference, vol. 10, no. 4 pp. 491-503, Aug. 2005.

[30] IEEE 802.11 (ISO/IEC 8802-11: 1999), "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specification", 1999 Edition.

[31] Intanagonwiwat, C., Govindan, R., Estrin, D., Heidemann, J., Silva F., "Directed Diffusion for Wireless Sensor Networking", IEEE Transactions on Networking, February 2003.

[32] Jakobson, G. and Weissman, M., "Real-time Telecommunication Network Management: extending event correlation with temporal constraints", 4th International Symposium on Integrated Network Management, vol. 4, pp. 290-201, Chapman & Hall, London, UK, 1995.

[33] Karp, R., Elson, J., Estrin, D. and Shenker, S., "Optimal and Global Time Synchronization in Sensornets", CENS Technical Report 0012, Center for Embedded Networked Sensing, UCLA, Apr. 2003.

[34] Klinger, S., Yemini, S., Yemini, Y., Ohsie, D. and Stolfo, S., "A Coding Approach to Event Correlation", 4th International Symposium on Integrated Network Management IV, pp. 266-277, Chapman & Hall, London, UK, 1995.

[35] Kruegel, C., Toth, T. and Kerer, C., "Decentralized Event Correlation for Intrusion Detection", International Conference on Information Security and Cryptology (ICISC '01), Lecture Notes in Computer Science, Springer Verlag, Korea, December 2001.

[36] Lamport, L., "Time, clocks, and the ordering of events in a distributed system", in Communications of the ACM, vol. 21, no. 7, pp. 558-565, Jul. 1978.

[37] Liao, C., Martonosi, M. and Clark, D. W., "Experience with an adaptive global-synchronizing clock algorithm", 11th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA), pp. 106-114, New York, June 1999.

[38] Liu, G., Mok, A. K. and Konana, P., "A Unified Approach for Specifying[Timing Constraints and Composite Events in Active Real-Time Database Systems", 4th IEEE Real-Time Technology and Applications Symposium (RTAS '98), IEEE Computer Society, Washington D.C., 1998.

[39] Liu, G., Mok, A. K. and Yang, E., "Composite Events for Network Event Correlation", IEEE/IFIP International Symposium on Integrated Network Management, Boston, MA, May 1999.

[40] Lucklam, D., "Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems", Addison-Wesley Professional, ISBN13: 978-0201727890, May 2002.

[41] McCanne, S. and Floyd, S., "ns Network Simulator", last accessed on March 2, 2008 at http://www.isi.edu/nsnam/ns/

[42] Meier, R. and Cahill, V., "Taxonomy of Distributed Event-Based Programming Systems", 22nd IEEE International Conference on Distributed Computing Systems (ICDCSW '02), pp. 585-588, IEEE Computer Society, Washington, DC, July 2002.

[43] Mock, M., Frings, R., Nett, E. and Trikaliotis, S., "Continuous clock synchronization in wireless real-time applications", 19th IEEE Symposium on Reliable Distributed Systems (SRDS'00), pp. 125-133, Nurnburg, Germany, Oct. 2000.

[44] Moon, S., Skelley, P. and Towsley, D., "Estimation and Removal of Clock Skew from Network Delay Measurements", 19th International IEEE Conference on Computer Communications (INFOCOM '99), New York, Mar, 1999.

[45] Oberkampf, W., Helton, J. and Sentz, K., "Mathematical Representations of Uncertainty." AIAA Conference Proceedings: Non-Deterministic Approaches Forum (AIAA2001-1645), Seattle, WA, April 2001.

[46] Parekh, J., "Privacy-Preserving Distributed Event Correlation", Columbia University, March 2007.

[47] Pissinou, N., Makki, K., König-Ries, B., "Mobile users in heterogeneous environments with middleware platform", IEEE Computer Communications, vol. 26, no. 7, pp. 700-707, 2003.

[48] Pissinou, N., Snodgrass, R. T., Elmasri, E., Mumick, et al., "Towards an Infrastructure for Temporal Databases", ACM Sigmod Record, vol. 23, no. 1, pp. 35-51, 1994.

[49] Quinlan, J. R., "Simplifying Decision Trees", International Journal of Human-Computer Studies, vol. 51, no. 2, pp. 497-510, 1999

[50] Römer, K., "Time Synchronization in Ad Hoc Networks", 2nd ACM International Symposium on Mobile Ad Hoc Networking & Computing (MobiHoc'01), Long Beach, Oct. 2001.

[51] Römer, K. and Blum, L. M., "Time Synchronization and Calibration in Wireless Sensor Networks", I. Stojmenovic, ed., "Wireless Sensor Networks", Wiley and Sons, 2005.

[52] Sanchez, C., Sankaranarayanan, S., Sipma, H. B., Zhang, T., Dill, D. and Manna, Z., "Event Correlation: Language and Semantics", 3rd International Conference on Embedded Software (EMSOFT '03), Lecture Notes in Computer Science, vol. 2855, pp. 323-339, Springer Verlag, Philadelphia, PA, 2003.

[53] Savage, L. J., "Foundations of Statistics", Dover Publications, 2nd Edition, ISBN13: 978-0486623498, June 1972.

[54] Sentz, K. and Ferson, S., "Combination of Evidence in Dempster-Shafer's Theory", Sandia National Laboratories Technical Report, SAND 2002-0835, April 2002.

[55] Shafer, G., "Mathematical Theory of Evidence", Princeton University Press, ISBN13: 978-0691081755, March 1976.

[56] Sichitiu, M. L. and Veerarittiphan, C., "Simple, Accurate Time Synchronization for Wireless Sensor Networks", IEEE Wireless Communications and Networking Conference (WCNC'03), IEEE, vol. 2, pp. 16-20, Mar. 2003.

[57] SMARTS Incharge, last accessed on February 12, 2008 at http://www.nsai.net/products/incharge-asm.shtml

[58] Smets, P., "What is Dempster-Shafer's Model?", Advances in Dempster-Shafer Theory of Evidence, Yager, R. R., Kacprzyk J. and Fedrizzi, M. (eds.), pp. 5-34, John Wiley & Sons, Inc. ISBN13: 978-0471552488, New York, 1994

[59] Solis, R., Borkar, V. S. and Kumar, P. R., "A New Distributed Time Synchronization Protcol for Multihop Wireless Networks", 45th IEEE Conference on Decision and Control (CDC '06), San Diego, Dec. 2006.

[60] Staniford, S., Paxson, V. and Weaver, N., "How to 0wn the Internet in Your Spare Time", 11th USENIX Security Symposium (SEC '02), August 2002.

[61] Stoica, I., "Stateless Core: A Scalable Approach for Quality of Service in the Internet", Lecture Notes in Computer Science, vol. 2979, no. 16, Springer, ISBN13: 978-3540219506, 2004.

[62] Strauss H., "Towards a Ubiquitous Future", ITU-T New Initiatives on Ubiquitous Network Societies, April 2005, Geneva.

[63] TIBCO BusinessEvents, last accessed on March 1, 2008 at http://www.tibco.com/software/complex_event_processing/businessevents/default.jsp

[64] Tiffany, M., "A Survey of Event Correlation Techniques and Related Topics", Research Paper, Georgia Institute of Technology, May 2002.

[65] Vaarandi, R., "SEC - A Lightweight Event Correlation Tool", 2002 IEEE Workshop on IP Operations and Management, pp. 111-115, October 2002.

[66] Wang, H., Yip, L., Maniezzo, D., Chen, J. C., Hudson, R. E., Elson, J. and Yao, K., "A Wireless Time-Synchronized COTS Sensor Platform Part II – Applications to Beamforming", IEEE CAS Workshop on Wireless Communications and Networking, Pasadena, CA, Sept. 2002.

[67] Witten, I. H. and Frank, E., "Data Mining: Practical Machine Learning Tools and Techniques", Morgan Kauffman, 2nd Edition, ISBN13: 978-0120884070, June 2005.

[68] Yemini, S., Kliger, S., Mozes, E., Yemini, Y. and Ohsie, D., "High Speed and Robust Event Correlation", IEEE Communications Magazine, vol. 34, no. 5, pp: 82-90, May 1996

[69] Zeffler, K.-P., "First Results from EURESCOM P408 Pan-European-TMN Laboratory "PET-Lab" - A Joint Research Effort in TMN", IEEE Network Operations and Management Symposium, vol. 2, pp. 482-491, Kyoto, Japan, April 1996.

[70] Zhang, L., Liu, Z. and Xia, C. H., "Clock Synchronization Algorithms for Network Measurements", 21st IEEE International Conference on Computer Communications (INFOCOM '02), New York, Jun. 2002.

[71] Zhao, X., Ganapathy, V., Pissinou, N. and Makki, K., "Revisiting Global Time Synchronization", 50th IEEE Global Communications Conference (GLOBECOM '07), Washington D.C., November, 2007.

[72] Zheng, Q., Xu, K., Lv, W. and Ma, S., "Intelligent Search of Correlated Alarms for GSM Networks with Model-based Constraints", Computing Research Repository (CoRR), June 2002.

A P P E N D I X I

Lemma 1: Inside a n-node IBSS, the probability that i nodes have beaconed after i TBTTs is given by:

$$P(i, i) = \frac{i!}{(n-i)!} \cdot \frac{1}{n^i},$$

Proof: Omitting beacon collision, the event (1, 1), that a node beacons after a TBTT, will surely occur. So, $P(1, 1) = 1$.

The event (2, 2), that two different nodes beacon after two TBTTs, can only be achieved via event (1, 1), because only one node is permitted to beacon at each TBTT. Additionally, the node that beacons at the 2nd TBTT must be different from the node that has already beaconed at the 1st TBTT. Thus,

$$P(2, 2) = P(1, 1) \cdot \frac{n-1}{n} = \frac{n}{n} \cdot \frac{n-1}{n} = \frac{n!}{(n-2)!} \cdot \frac{1}{n^2}.$$

Similarly, event (3, 3) can only evolve from the event (2, 2). Additionally, the node that beacons at the third TBTT must be different from the nodes that beacon at the first two TBTTs. Thus,

$$P(3, 3) = P(2, 2) \cdot \frac{n-2}{n} = \frac{n!}{(n-2)!} \cdot \frac{1}{n^2} \cdot \frac{n-2}{n} = \frac{n!}{(n-3)!} \cdot \frac{1}{n^3},$$

By induction, the probability of the event (4, 4), …, (i, i), …, (n, n) is obtained as:

$$P(4, 4) = P(3, 3) \cdot \frac{n-3}{n} = \frac{n!}{(n-4)!} \cdot \frac{1}{n^4}, \ldots,$$

$$P(i, i) = \frac{i!}{(n-i)!} \cdot \frac{1}{n^i}, \ldots \ldots, P(n, n) = \frac{n!}{n^n} \qquad \square$$

APPENDIX II

As the fastest time is diffused via intermediate nodes to the farthest node, the probability $P(L, k)$ that the farthest node is synchronized can be used to denote the probability that the entire MANET has been synchronized. For an intermediate node i lying between the source node and the farthest node, $P(i \cdot d, k)$ implies the probability that the fastest time has diffused a distance $i \cdot d$ in k TBTTs. This may not necessarily imply that fastest time has not diffused beyond node i towards $(i+1)^{th}$ node or further. To determine the probability that the fastest time has actually diffused into i^{th} node and no further than that, a correction probability factor needs to be applied to determine a more accurate snapshot regarding time synchronization of the ith node:

$$P_{cut}(i \cdot d, k) = P(i \cdot d, k) - P((i+1) \cdot d, k) \qquad (R_{GS6})$$

Analogous to case I, the nodes are partitioned in two – those lying within the single-hop range h, and the outliers, so that, using (A_{GS1}), the probability at which the node farthest from the fastest node is synchronized can be determined as follow.

When $i < \frac{1}{2}(n-1)$, the fastest time disperses within the radio range of a single hop:

$$P(i \cdot d, k) = 1 - \left(\frac{n-1}{n}\right)^k, \; i = 1, 2, \ldots, k \qquad (R_{GS7})$$

The argument behind (R_{GS7}) is that after k TBTTs, the probability that the node with the fastest time has no chance to beacon is $[(n-1)/n]^k$, which is equal to 1- $P(i \cdot d, k)$. For outlier nodes, the probability will be influenced by correction provided by (R_{GS6}).

Lemma 2: All nodes of MANET are dispersed homogeneously along a line with inter-node distance *d*, and not all within a unit hop distance. Then, when $i > \frac{1}{2}(n-1)$, the probability that the ith node is successfully time synchronized within *k* TBTTs is given by:

$$P(i, k) = P(i, k-1) + \{P[(i - \tfrac{n-1}{2})\cdot d, k-1] - P(i, k-1)\} \frac{1}{n}, \; i > \frac{n-1}{2} \text{ and } k = 2, 3, \ldots, \infty \quad (R_{GS4})$$

Proof:

$$P(i\cdot d, k) =$$

$$P(i\cdot d, k-1)$$

$$+\left\{P\left(\left(i-\frac{n-1}{2}\right)\cdot d, k-1\right) - P\left(\left(i-\frac{n-1}{2}+1\right)\cdot d, k-1\right)\right\}\cdot\frac{1}{n}$$

$$+\left\{P\left(\left(i-\frac{n-1}{2}+1\right)\cdot d, k-1\right) - P\left(\left(i-\frac{n-1}{2}+2\right)\cdot d, k-1\right)\right\}\cdot\frac{1}{n}$$

$$+\left\{P\left(\left(i-\frac{n-1}{2}+2\right)\cdot d, k-1\right) - P\left(\left(i-\frac{n-1}{2}+3\right)\cdot d, k-1\right)\right\}\cdot\frac{1}{n}$$

$$\ldots$$

$$+\{P((i-1)\cdot d, k-1) - P(i\cdot d, k-1)\}\cdot\frac{1}{n}$$

$$= P(i\cdot d, k-1) + \left\{P\left(\left(i-\frac{n-1}{2}\right)\cdot d, k-1\right) - P(i\cdot d, k-1)\right\}\cdot\frac{1}{n}$$

The argument behind (R_{GS4}) is that the ith node can be synchronized in k TBTTs in the following ways – when i^{th} nodes have been synchronized in k-1 TBTTs, or, when $(i-(n-1)/2)^{th}$ node just synchronized its time with the fastest node in the k-1 TBTT and shall beacon at the next TBTT, or, when $(i-(n-1)/2+1)^{th}$ node just synchronized its time with the fastest node in the k-1 TBTT and shall beacon at the next TBTT, and so on, until, when $(i-1)^{th}$ node just synchronized its time with the fastest node in the k-1 TBTT and shall beacon at the next TBTT. □

APPENDIX III

(a) ns-2 source code for Chapter 3 (ns-allinone 2.29).

```
# gts.tcl
set val(chan)    Channel/WirelessChannel
set val(prop)    Propagation/TwoRayGround
set val(netif)   Phy/WirelessPhy
set val(mac)     Mac/802_11
set val(ifq)     Queue/DropTail/PriQueue
set val(ll)      LL
set val(ant)     Antenna/OmniAntenna
set val(ifqlen)  50
set val(nn)      16
set val(rp)      AODV

set ns_ [new Simulator]
set tracefd [open gts.tr w]
$ns_ trace-all $tracefd

# set up topography object
set topo [new Topography]

$topo load_flatgrid 500 500
create-god $val(nn)

set chan_ [new $val(chan)]

$ns_ node-config  -adhocRouting $val(rp) \
  -llType $val(ll) \
  -macType $val(mac) \
  -ifqType $val(ifq) \
  -ifqLen $val(ifqlen) \
  -antType $val(ant) \
  -propType $val(prop) \
  -phyType $val(netif) \
  -channel $chan_ \
  -topoInstance $topo \
  -agentTrace ON \
  -routerTrace OFF \
  -macTrace OFF \
  -movementTrace OFF

 for {set i 0} {$i < $val(nn)} {incr i} {
   set node_($i) [$ns_ node]
   $node_($i) random-motion 0
 }

for {set i 0} {$i < $val(nn)} {incr i} {
```

```
  $node_($i) set X_  [expr ($i*165)%660]
  $node_($i) set Y_   [expr ($i*165-($i*165)%660)/4]
  $node_($i) set Z_  0.0
}

for {set i 0} {$i < $val(nn)} {incr i} {
  set p_($i) [new Agent/GTS]
  $ns_ attach-agent $node_($i) $p_($i)
}

set period   30
set rep_num  1000
for {set i 0} {$i < $rep_num} {incr i} \
{
   $ns_ at [expr ($i)*($period+5)] "$p_(0) be_fastest"
   for {set j 1} {$j < [expr $period+1]} {incr j} \
   {
     for {set k 0} {$k < $val(nn)} {incr k} \
     {
       $ns_ at [expr $i*($period+5)+$j]
    }
 }

 for {set l 0} {$l < $val(nn)} {incr l} \
 {
   $ns_ at [expr ($i+1)*($period+5)-3]
   $ns_ at [expr ($i+1)*($period+5)-2.5]
 }
 $ns_ at [expr ($i+1)*($period+5)-1.5]
}
$ns_ at [expr ($period+5)*$rep_num+9]
$ns_ at [expr ($period+5)*$rep_num+10]
$ns_ at [expr ($period+5)*$rep_num+10.01]
$ns_ halt
proc stop {} {
    global ns_ tracefd
    $ns_ flush-trace
    close $tracefd
}
$ns_ run
```

```
# lts.tcl
set val(chan)   Channel/WirelessChannel
set val(prop)   Propagation/TwoRayGround
set val(netif)  Phy/WirelessPhy
set val(mac)    Mac/802_11
set val(ifq)    Queue/DropTail/PriQueue
set val(ll)     LL
set val(ant)    Antenna/OmniAntenna
set val(ifqlen) 50
set val(nn)     16
set val(rp)     AODV

set ns_ [new Simulator]
set tracefd [open lts.tr w]
$ns_ trace-all $tracefd

set topo [new Topography]
$topo load_flatgrid 500 500

create-god $val(nn)

set chan_ [new $val(chan)]

$ns_ node-config -adhocRouting $val(rp) \
  -llType $val(ll) \
  -macType $val(mac) \
  -ifqType $val(ifq) \
  -ifqLen $val(ifqlen) \
  -antType $val(ant) \
  -propType $val(prop) \
  -phyType $val(netif) \
  -channel $chan_ \
  -topoInstance $topo \
  -agentTrace ON \
  -routerTrace OFF \
  -macTrace OFF \
  -movementTrace OFF

for {set i 0} {$i < $val(nn)} {incr i} {
  set node_($i) [$ns_ node]
  $node_($i) random-motion 0 ;# disable random motion
}

for {set i 0} {$i < $val(nn)} {incr i} {
  $node_($i) set X_ [expr ($i*165)%660]
  $node_($i) set Y_ [expr ($i*165-($i*165)%660)/4]
  $node_($i) set Z_ 0.0
}

for {set i 0} {$i < $val(nn)} {incr i} {
  set p_($i) [new Agent/LTS]
 $ns_ attach-agent $node_($i) $p_($i)
}
```

```
set n [new RandomVariable/Uniform]
$n set max_ 0.0003
$n set min_ 0.0

set period 120
set rep_num 1000
for {set i 0} {$i < $rep_num} {incr i} \
{
  for {set j 1} {$j < [expr $period+1]} {incr j} \
  {
    for {set k 0} {$k < $val(nn)} {incr k} \
    {
       $ns_ at [expr $i*($period+5)+$j+[$n value]+0.001]
    }
  }
  for {set k 0} {$k < $val(nn)} {incr k} \
  {
    $ns_ at [expr ($i+1)*($period+5)-3]
    $ns_ at [expr ($i+1)*($period+5)-2.5]
  }
  $ns_ at [expr ($i+1)*($period+5)-1.5]
}

$ns_ at [expr ($period+5)*$rep_num+9]
$ns_ at [expr ($period+5)*$rep_num+10]
$ns_ at [expr ($period+5)*$rep_num+10.01]
$ns_ halt"
proc stop {} {
 global ns_ tracefd
 $ns_ flush-trace
 close $tracefd
}

$ns_ run
```

```
// gts.h

#ifndef ns_gts_h
#define ns_gts_h

#define NODE_NUM 16

#include "agent.h"
#include "tclcl.h"
#include "packet.h"
#include "address.h"
#include "ip.h"

struct hdr_gts {
  bool time_mark;
  int seq;
  static int offset_;
  inline static int& offset() { return offset_; }
  inline static hdr_gts* access(const Packet* p) {
    return (hdr_gts*) p->access(offset_);
  }
};

class GTS_Agent : public Agent {
public:
 GTS_Agent();
 int seq;
 bool time_mark;
 static int flag[NODE_NUM];
 static int has_syn;
 static int hasnot_syn;
 static float success;
 static float repeats;
 virtual int command(int argc, const char*const* argv);
 virtual void recv(Packet*, Handler*);
};
#endif
```

```
// gts.cc
#include "gts.h"

int hdr_gts::offset_;
static class GTS_HeaderClass : public PacketHeaderClass {
  public:
   GTS_HeaderClass() : PacketHeaderClass("PacketHeader/GTS",
   sizeof(hdr_gts)) {bind_offset(&hdr_gts::offset_);}
} class_gts_hdr;

static class GTS_Class : public TclClass {
 public:
   GTS_Class() : TclClass("Agent/GTS") {}
   TclObject* create(int, const char*const*) {return (new GTS_Agent());}
} class_gts;

int GTS_Agent::flag[NODE_NUM]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
int GTS_Agent::has_syn=0;
int GTS_Agent::hasnot_syn=0;
float GTS_Agent::success=0.0;
float GTS_Agent::repeats=0.0;
GTS_Agent::GTS_Agent():Agent(PT_GTS),seq(0){bind("packetSize_", &size_);}

int GTS_Agent::command(int argc, const char*const* argv)
{
  if (argc == 2) {
  if (strcmp(argv[1], "send") == 0) {
    Packet* pkt = allocpkt();
    hdr_ip* iph = HDR_IP(pkt);
    hdr_gts* th = hdr_gts::access(pkt);
    iph->daddr() = IP_BROADCAST;
    iph->dport() = iph->sport();
    seq++;
    th->seq = seq;
    th->time_mark = time_mark;
    flag[here_.addr_]=0;
    send(pkt, (Handler*) 0);
    return (TCL_OK);
 }
 if (strcmp(argv[1], "be_fastest") == 0) {
   time_mark=TRUE;return (TCL_OK);}
   if (strcmp(argv[1], "reset") == 0) {
     time_mark=FALSE;
     has_syn=0;
     hasnot_syn=0;
     flag[here_.addr_]=0;
     return (TCL_OK);
   }
   if (strcmp(argv[1], "cal_pro") == 0) {return (TCL_OK);}
   if (strcmp(argv[1], "check") == 0) {
     repeats++;
     if(!time_mark) {hasnot_syn++;} else {has_syn++;}
     if(has_syn==NODE_NUM) {success++;}
```

```
      return (TCL_OK);
    }
  }
  return (Agent::command(argc, argv));
}

void GTS_Agent::recv(Packet* pkt, Handler*)
{
  if (flag[here_.addr_]) {Packet::free(pkt); return; } else
  {
    flag[here_.addr_]=1;
    hdr_gts* hdrgts = hdr_gts::access(pkt);
    if((hdrgts->time_mark)&&(!time_mark)){time_mark=TRUE;}
    Packet::free(pkt);
    return;
   }
}
```

```
// lts.h

#ifndef ns_lts_h
#define ns_lts_h

#define NODE_NUM 16

#include "agent.h"
#include "tclcl.h"
#include "packet.h"
#include "address.h"
#include "ip.h"

struct hdr_lts {
 int seq;
 static int offset_;
 inline static int& offset() {return offset_;}
 inline static hdr_lts* access(const Packet* p) {
    return (hdr_lts*) p->access(offset_);
 }
};

class LTS_Agent : public Agent {
public:
 LTS_Agent();
 int seq;
 static int flag[NODE_NUM];
 static int syn[NODE_NUM];
 static int has_syn;
 static int hasnot_syn;
 static float success;
 static float repeats;
 virtual int command(int argc, const char*const* argv);
 virtual void recv(Packet*, Handler*);
};
#endif
```

```
// lts.cc

#include "lts.h"
#include <math.h>

int hdr_lts::offset_;
static class LTS_HeaderClass : public PacketHeaderClass {
  public:
    LTS_HeaderClass() : PacketHeaderClass("PacketHeader/LTS",
    sizeof(hdr_lts)) { bind_offset(&hdr_lts::offset_); }
} class_lts_hdr;

int LTS_Agent::has_syn=0;
int LTS_Agent::hasnot_syn=0;
float LTS_Agent::success=0.0;
float LTS_Agent::repeats=0.0;

int LTS_Agent::syn[NODE_NUM]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
int LTS_Agent::flag[NODE_NUM]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};

static class LTS_Class : public TclClass {
  public:
    LTS_Class():TclClass("Agent/LTS") {}
    TclObject* create(int, const char*const*) {
    return (new LTS_Agent());
  }
} class_lts;

LTS_Agent::LTS_Agent():Agent(PT_LTS),seq(0){bind("packetSize_", &size_);}

int LTS_Agent::command(int argc, const char*const* argv)
{
  if (argc == 2) {
    if (strcmp(argv[1], "send") == 0) {
      // allocate a packet
      Packet* pkt = allocpkt();
      // get the access of the ip head and lts head of the packet
      hdr_ip* iph = HDR_IP(pkt);
      hdr_lts* th = hdr_lts::access(pkt);

      // assign the broadcast address and port of the ip head
      iph->daddr() = IP_BROADCAST;
      iph->dport() = iph->sport();
      // increase the sequence number of lts head
      seq++;
      th->seq = seq;
      flag[here_.addr_]=0;
      send(pkt, (Handler*) 0);
      return (TCL_OK);
   }
   if (strcmp(argv[1], "reset") == 0) {
      has_syn=0;
      hasnot_syn=0;
```

```
      syn[here_.addr_]=0;
      flag[here_.addr_]=0;
      return (TCL_OK);
    }
    if (strcmp(argv[1], "cal_pro") == 0) {return (TCL_OK);}
    if (strcmp(argv[1], "check") == 0) {
      repeats++;
      if (syn[here_.addr_]) {has_syn++;} else {hasnot_syn++;}
      if (has_syn==NODE_NUM) {success++;}
      return (TCL_OK);
    }
  }
  return (Agent::command(argc, argv));
}

void LTS_Agent::recv(Packet* pkt, Handler*)
{
   hdr_ip* hdrip = hdr_ip::access(pkt);
   if (flag[here_.addr_]==1) {Packet::free(pkt);return;}
   flag[here_.addr_]=1;
   if (flag[hdrip->saddr()]==1) {Packet::free(pkt); return;}
   flag[hdrip->saddr()]=1;
   syn[hdrip->saddr()]=1;
   Packet::free(pkt);
   return;
}
```

(b) C++ source code for Chapter 4 (Microsoft C++ 2005, boost 1.34.1)

```
// dst-normal.cpp

#include "stdafx.h"
#include "iostream"
#include "conio.h"
#include <stdlib.h>
#include <stdio.h>
#include <time.h>

#define UNKNOWN 6554

#define UNIFORM 16384
#define REASONS 3
#define NEIGHBORS 5
#define POWERSET 8
#define ROUNDS 10

int _tmain(int argc, _TCHAR* argv[])
{
 int test_print = 1, file_print = 1;
 int i, j, i3, j3;
 int a1[REASONS][NEIGHBORS], i1, j1;
 int a2[REASONS][NEIGHBORS], i2, j2;
 float R[(POWERSET+1)][(POWERSET+1)], F[ROUNDS][(POWERSET+2)];
 FILE *f;

 srand((unsigned)time(NULL));

 for (j1=0; j1<NEIGHBORS; j1++)
 {
   for (i1=0; i1<REASONS; i1++)
   {
     a1[i1][j1] = 0;
     a2[i1][j1] = 0;
   }
 }
 for (i2=0; i2<(POWERSET+1);i2++) {
   for (j2=0;j2<(POWERSET+1);j2++) {R[i2][j2] = 0.0;}
 }
 for (i3=0; i3<ROUNDS;i3++) {
   for (j3=0;j3<(POWERSET+2);j3++) {F[i3][j3] = 0.0;}
 }

 for (j1=0; j1<NEIGHBORS; j1++){
   for (i1=1; i1<REASONS; i1++) {
     i = rand();
     if (i < UNIFORM) {a1[i1][j1]=1;} else {a1[i1][j1]=0;};
     i = rand();
     if (i < UNIFORM) {a2[i1][j1]=1;} else {a2[i1][j1]=0;};
   }
}
```

```
if (test_print == 1)
{
 for (i1=0; i1<REASONS; i1++) {
   for (j1=0; j1<NEIGHBORS; j1++) {
     printf ("%0d ",a1[i1][j1]); }
     printf("\n");
 }
 printf("\n");
 for (i1=0; i1<REASONS; i1++) {
   for (j1=0; j1<NEIGHBORS; j1++) {
     printf ("%0d ",a2[i1][j1]); }
     printf("\n");
 }
 printf("\n");
}

for (j1=0; j1<NEIGHBORS; j1++) {
 if (a1[0][j1] == 1 && a1[1][j1] == 0 && a1[2][j1] == 0) {R[2][0]++;}
 if (a1[0][j1] == 0 && a1[1][j1] == 1 && a1[2][j1] == 0) {R[3][0]++;}
 if (a1[0][j1] == 0 && a1[1][j1] == 0 && a1[2][j1] == 1) {R[4][0]++;}
 if (a1[0][j1] == 1 && a1[1][j1] == 1 && a1[2][j1] == 0) {R[5][0]++;}
 if (a1[0][j1] == 0 && a1[1][j1] == 1 && a1[2][j1] == 1) {R[6][0]++;}
 if (a1[0][j1] == 1 && a1[1][j1] == 0 && a1[2][j1] == 1) {R[7][0]++;}
 if (a1[0][j1] == 1 && a1[1][j1] == 1 && a1[2][j1] == 1) {R[8][0]++;}
 if (a2[0][j1] == 1 && a2[1][j1] == 0 && a2[2][j1] == 0) {R[0][2]++;}
 if (a2[0][j1] == 0 && a2[1][j1] == 1 && a2[2][j1] == 0) {R[0][3]++;}
 if (a2[0][j1] == 0 && a2[1][j1] == 0 && a2[2][j1] == 1) {R[0][4]++;}
 if (a2[0][j1] == 1 && a2[1][j1] == 1 && a2[2][j1] == 0) {R[0][5]++;}
 if (a2[0][j1] == 0 && a2[1][j1] == 1 && a2[2][j1] == 1) {R[0][6]++;}
 if (a2[0][j1] == 1 && a2[1][j1] == 0 && a2[2][j1] == 1) {R[0][7]++;}
 if (a2[0][j1] == 1 && a2[1][j1] == 1 && a2[2][j1] == 1) {R[0][8]++;}
}

for (i=2; i<(POWERSET+1); i++) {
 R[i][0] /= (float) NEIGHBORS;
 R[0][i] /= (float) NEIGHBORS;
 R[1][0] += R[i][0];
 R[0][1] += R[0][i];
}
R[1][0] = 1-R[1][0];
R[0][1] = 1-R[0][1];

for (i2=1; i2<(POWERSET+1); i2++) {
  for (j2=1; j2<(POWERSET+1); j2++) {
    R[i2][j2]=R[0][j2]*R[i2][0]; }
}

F[0][0] = R[1][1];
F[0][1] =
R[2][2]+R[1][2]+R[2][1]+R[2][5]+R[5][2]+R[2][7]+R[7][2]+R[2][8]+R[8][2];
 F[0][2] =
R[3][3]+R[1][3]+R[3][1]+R[3][5]+R[5][3]+R[3][6]+R[6][3]+R[3][8]+R[8][3];
```

```
 F[0][3] =
R[4][4]+R[1][4]+R[4][1]+R[4][6]+R[6][4]+R[4][7]+R[7][4]+R[4][8]+R[8][4];
 F[0][4] = R[5][5]+R[1][5]+R[5][1]+R[5][8]+R[8][5];
 F[0][5] = R[6][6]+R[1][6]+R[6][1]+R[6][8]+R[8][6];
 F[0][6] = R[7][7]+R[1][7]+R[7][1]+R[7][8]+R[8][7];
 F[0][7] = R[8][8];

 for (i3=0; i3<POWERSET; i3++) {
   F[0][POWERSET] += F[0][i3];
 }
 F[0][POWERSET+1] = 1 - F[0][POWERSET];

 if (test_print == 1) {
   for (i2=0; i2<(POWERSET+1); i2++) {
 for (j2=0; j2<(POWERSET+1); j2++)
 {
 printf ("%2.3f ",R[i2][j2]);
 }
 printf("\n");
 }
 printf("\n");
 for (j=0; j<(POWERSET+2); j++) {printf("%2.3f ",F[0][j]);}
 char ch = _getch();
 printf("\n");
}

int a4[REASONS][NEIGHBORS], i4, j4;
int k, l;

for (k=1; k<ROUNDS; k++){
  for (j4=0; j4<NEIGHBORS; j4++) {
    for (i4=0; i4<REASONS; i4++) {
      a4[i4][j4] = 0;
      a4[i4][j4] = 0; }
}

for (j4=0; j4<NEIGHBORS; j4++) {
  for (i4=0; i4<REASONS; i4++) {
    i = rand();
    if (i < UNIFORM) {a4[i4][j4]=1;} else {a4[i4][j4]=0;}; }
}

for (l = 0; l < POWERSET; l++) {
  R[0][l+1] = F[k-1][l];
  R[l+1][0] = 0;
}

#ifdef DEBUG_PRINT
for (i=0; i<(POWERSET+1); i++) {
 for (j=0; j<(POWERSET+1); j++) { printf ("%2.3f ",R[i][j]); }
   printf("\n"); }
   printf("\n");
#endif
```

```
for (j4=0; j4<NEIGHBORS; j4++) {
 if (a4[0][j4] == 1 && a4[1][j4] == 0 && a4[2][j4] == 0) {R[2][0]++;}
 if (a4[0][j4] == 0 && a4[1][j4] == 1 && a4[2][j4] == 0) {R[3][0]++;}
 if (a4[0][j4] == 0 && a4[1][j4] == 0 && a4[2][j4] == 1) {R[4][0]++;}
 if (a4[0][j4] == 1 && a4[1][j4] == 1 && a4[2][j4] == 0) {R[5][0]++;}
 if (a4[0][j4] == 0 && a4[1][j4] == 1 && a4[2][j4] == 1) {R[6][0]++;}
 if (a4[0][j4] == 1 && a4[1][j4] == 0 && a4[2][j4] == 1) {R[7][0]++;}
 if (a4[0][j4] == 1 && a4[1][j4] == 1 && a4[2][j4] == 1) {R[8][0]++;}
}

#ifdef DEBUG_PRINT
for (i=0; i<(POWERSET+1); i++) {
 for (j=0; j<(POWERSET+1); j++) {
   printf ("%2.3f ",R[i][j]); }
   printf("\n");
}
printf("\n");
#endif

for (i=2; i<(POWERSET+1); i++) {
  R[i][0] /= (float) NEIGHBORS;
  R[1][0] += R[i][0];
}
R[1][0] = 1 - R[1][0];

#ifdef DEBUG_PRINT
for (i=0; i<(POWERSET+1); i++) {
 for (j=0; j<(POWERSET+1); j++) {
   printf ("%2.3f ",R[i][j]); }
   printf("\n");
}
printf("\n");
#endif

for (i4=1; i4<(POWERSET+1); i4++) {
 for (j4=1; j4<(POWERSET+1); j4++) {
   R[i4][j4]=R[0][j4]*R[i4][0]; }
}

#ifdef DEBUG_PRINT
for (i=0; i<(POWERSET+1); i++) {
 for (j=0; j<(POWERSET+1); j++) {
   printf ("%2.3f ",R[i][j]); }
   printf("\n");
}
printf("\n");
#endif

F[k][0] = R[1][1];
 F[k][1] =
R[2][2]+R[1][2]+R[2][1]+R[2][5]+R[5][2]+R[2][7]+R[7][2]+R[2][8]+R[8][2];
 F[k][2] =
R[3][3]+R[1][3]+R[3][1]+R[3][5]+R[5][3]+R[3][6]+R[6][3]+R[3][8]+R[8][3];
```

```
 F[k][3] =
R[4][4]+R[1][4]+R[4][1]+R[4][6]+R[6][4]+R[4][7]+R[7][4]+R[4][8]+R[8][4];
 F[k][4] = R[5][5]+R[1][5]+R[5][1]+R[5][8]+R[8][5];
 F[k][5] = R[6][6]+R[1][6]+R[6][1]+R[6][8]+R[8][6];
 F[k][6] = R[7][7]+R[1][7]+R[7][1]+R[7][8]+R[8][7];
 F[k][7] = R[8][8];

for (i=0; i<POWERSET; i++) { F[k][POWERSET] += F[k][i];}
F[k][POWERSET+1] = 1 - F[k][POWERSET];

if (test_print == 1) {
  for (i4=0; i4<REASONS; i4++)  {
   for (j4=0; j4<NEIGHBORS; j4++) {
    printf ("%0d ",a4[i4][j4]);  }
    printf("\n");
 }
 printf("\n");

for (i=0; i<(POWERSET+1); i++) {
 for (j=0; j<(POWERSET+1); j++) {
   printf ("%2.3f ",R[i][j]); }
   printf("\n"); }
   printf("\n");

  for (l=0; l<(POWERSET+2); l++) {
    printf ("%2.3f ",F[k][l]); }
    printf("\n");
    printf("\n");
  }
}

if (test_print == 1) {
  for (i=0; i<ROUNDS; i++) {
    for (j=0; j<(POWERSET+2); j++) {
      printf ("%2.3f ",F[i][j]); }
      printf("\n"); }
      printf("\n");
}
if (file_print == 1) {
  if (f = fopen("result.txt", "a+"), f != NULL) {
    for (i=0; i<ROUNDS; i++) {
      for (j=0; j<(POWERSET+2); j++) {
      fprintf (f, "%2.3f ",F[i][j]); }
      fprintf(f,"\n"); }
      fprintf(f,"\n"); fclose(f); } }
  return 0;
}
```

V I T A

VINAYAK GANAPATHY

04/2008	Department of Electrical and Computer Engineering, Florida International University, Miami, Florida, Doctoral candidate in Electrical Engineering. Dissertation: A New Pervasive Network Management Scheme, Major Advisor: Dr. Niki Pissinou.
08/2007	Telecommunications and Information Technology Institute, Florida International University, Miami, Florida. M.S. Telecommunications and Networking.
05/2001	Institute of Informatics and Communication, University of Delhi South Campus, Delhi, India. M. Tech. Information Technology.
07/1998	Department of Physics, St. Stephen's College, University of Delhi, Delhi, India. B. Sc. (Hons.) Physics.

PUBLICATIONS

1. Zhao, X., Ganapathy, V., Pissinou, N., Makki, K., "Self-Organized Forensic Support in MANETs", 10th Workshop on Advances on Parallel and Distributed Processing Symposium (APDCM '08), Miami, FL, Apr. 10-12, 2008 (*to appear*).

2. Zhao, X., Ganapathy, V., Pissinou, N., Makki, K., "Revisiting Global Time Synchronization", 50th Annual IEEE Global Communications Conference (GLOBECOM '07), Nov. 26-30, Washington D.C., 2007.