Distributed Monitoring System for Mobile Ad Hoc Networks: Design and Implementation

Hanif S. Kazemi

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science

in

Electrical Engineering

Dr. Luiz A. DaSilva, Chair

Dr. Scott F. Midkiff

Dr. Mohamed Eltoweissy

May 4, 2007

Arlington, Virginia

Keywords: MANET, distributed monitoring, cooperation, assessment, implementation

Distributed Monitoring System for Mobile Ad Hoc Networks: Design and Implementation

Hanif S. Kazemi

(Abstract)

Mobile Ad hoc NETworks (MANETs) are networks in which the participating nodes can move freely without having to worry about maintaining a direct connection to any particular fixed access point. In a MANET, nodes collaborate with each other to form the network and as long as a node is in contact with any other member of the network, it – at least in theory – is part of the network and can communicate with all other nodes.

An important function of network management is to observe current network conditions: at the node level, this may mean keeping track of arriving and departing traffic load; at the network level, the system must monitor active routes and changes in network topology.

In this research, we present the design and implementation of a distributed network monitoring system for MANETs. Our system is completely distributed, generates no additional traffic on the network and produces a dynamic picture of the network level and node level information on a graphical user interface.

In our proposed scheme, multiple monitoring nodes collaborate to achieve a reasonably accurate snapshot of the network conditions. These monitoring nodes passively sniff network traffics and gather information from the network to construct partial network views. They then transmit their findings to a management unit where these local views are put together to produce a comprehensive picture of the network. The communication between all management nodes (a monitoring unit and a management node) takes place in an out-of-band communication link. Therefore, our monitoring solution does not depend on the MANET to perform, hence is robust to network partitioning, link breaks, node's death and node misbehavior in the monitored MANET.

Our solution provides a snapshot of the network topology that includes information about node-level behavior ratings and traffic activity.

The information provided by our monitoring system can be used for network management as well as for security assessment, including anomaly detection. Information regarding individual nodes' behavior can be used for detecting selfishness in the network. Also, an approximation of arriving and departing traffic levels at each node is important in the context of quality of service, load balancing and congestion control. Furthermore, the network topology picture can provide valuable information to network management in detecting preferred routes, discovering network partitioning and in fault detection.

We developed a proof-of-concept implementation of our system, which works with the Optimized Link State Routing (OLSR) protocol. Through experimental studies with up to 10-node MANETs, we were able to determine the feasibility and workability of our system. The scheme proved to be robust with respect to mobility, rapid changes in the network topology and node connectivity. Throughout our experiments we observed that our system replicated changes in the network on the GUI with less than two seconds delay. Also, when deployed in a high-traffic environment, with multiple TCP and UDP flows throughout the network, the system was able to report traffic load on each node accurately and consistently.

On average, CPU consumption on monitoring nodes was about 3.5% and the GUI never took up more than 4% of the processing power (general-purpose laptop computers were used throughout the experiments). Also, the overall storage capacity needed for archiving the information files was estimated as 1 Mbytes for monitoring a 10-node MANETs for 30 minutes.

Unobtrusive and distributed nature of our proposed approach helps the system to adapt to the constantly changing nature of MANETs and be able to provide valuable network management, security assessment and traffic analysis services, while requiring only modest processing and storage resources. The system is capable of quickly responding to changes in the network and is non-intrusive, generating no additional traffic on the MANET it monitors.

Acknowledgements

I would like to extend my gratitude to Dr. Luiz A DaSilva, my academic and research advisor, for embarking with me on this research. You guided me on every aspect of this project and I thank you for your generosity in providing me with access to your lab and equipments. I am also grateful for your time, your endless patience and your excellent writing skills that helped me to produce this work.

I am also thankful to Dr. Midkiff and Dr. Eltoweissy for serving in my committee and for taking your time to review my work.

I owe the greatest deal of gratitude, respect and appreciation toward my parents, Masoud and Fahimeh, who offered me their profound love and unlimited sacrifice throughout my life and never stopped supporting me. I owe every bit of my success to them.

I would also like to extend my most sincere appreciations to Mohammad and Pegah Esfahani, without whose help I would not be where I am today.

I am also grateful to my friend Kaveh Mehrjoo, who introduced me to JPCAP and has never hesitated to offer me his sincere advice. I would also like to thank my brothers, Farid and Saeid, for their help throughout my research, providing me with much needed logistics and cooperation.

I am also thankful to my colleagues at S.W.I.F.T., particularly Amit Karnik and Kamil Ince, whose vast technical expertise was a valuable resource for me throughout my work.

List of Figures

List of Tables

1 Introduction

Wireless mobile ad hoc networks [1-3] are self-organizing and autonomous networks, consisting of mobile nodes that act as routers and collaborate with each other to form a network. A MANET may operate in standalone fashion or as a stub network connected to the Internet. Use of MANETs is expected to increase substantially as personal computing becomes more popular. For example, the “One Laptop per Child” program [4] has developed cheap laptops and plans for mass distribution to developing countries for education. These laptops will use ad hoc wireless mesh networking to form their own communication networks out of the box [5]. Another example is Ozone Inc. [6], an Internet service provider in France that uses ad hoc technologies to provide widespread commercial seamless Internet access in Paris.

Also, because MANETs do not require the costly communication infrastructure of legacy wireless networks and can be easily deployed, they are a viable solution for military battle sites [7], disaster sites [8], vehicular wireless communication [9], sensor networks [10], personal computing devices [1], and pervasive computing [11].

1.1 Motivation

Some of the features that characterize MANETs are dynamic network topology, high likelihood of network partitioning, undefined geographical coverage area, unlimited number of participating nodes in the network and limited bandwidth, storage capacity, battery life and processing power.

These characteristics pose a significant challenge to the management of mobile ad hoc networks. In particular, designing and implementing a monitoring solution that is capable of providing up-to-date information regarding the network’s overall health remains a challenge.

1.2 Our Contributions

In this thesis, we introduce a new framework to address the challenges of monitoring MANETs. We formulate the overall design structure of a monitoring solution that is generally applicable to mobile decentralized networks. We also present a detailed design and implementation of our framework for a MANET running the Optimized Link State Routing (OLSR) protocol.

The major contributions from this research are as follows:

- Formulation of a distributed framework for monitoring and performance assessment of mobile ad hoc networks.
- A monitoring solution for MANETs that is capable of producing an up-to-date picture of the network topology without requiring complete coverage of the monitored network by a single monitoring node.
- A traffic assessment scheme that provides statistics of node-level traffic activity that can be used in providing services such as load balancing, congestion control, etc.
- A solution for dynamic assessment of individual nodes' cooperation in forwarding packets for their neighbors, which helps identify selfish nodes in a mobile ad hoc network.
- An implementation of a MANET monitoring and assessment system designed based on the methods introduced in this thesis. The monitored MANET uses OLSR as its underlying routing protocol.

1.3 Structure of Document

The remainder of this thesis document is arranged as follows. In the next chapter we review background and previous work in the area of monitoring and fault detection of MANETs. In Chapter 3, we formulate and present our proposed monitoring and assessment framework for mobile ad hoc networks. In Chapter 4, we describe the implementation of our proposed solution for monitoring a MANET that uses OLSR as its routing protocol. We present the results of our performance evaluation in Chapter 5.

Finally, in Chapter 6 we summarize the findings and contributions of this thesis and discuss directions for future work.

2 Review of Related Work

Many network management systems depend on a monitoring scheme to provide them with current information regarding the network's health. A network monitoring tool is charged with providing network statistics. Then, based on this information, the network management system decides on the appropriate course of action to protect the network's well-being. Mobile ad hoc networks are no exception. A MANET management system, which could be implemented in conjunction with our monitoring solution, also needs to know about the network's health before it can decide on the type and magnitude of the required management tasks. In this chapter, we discuss several proposed MANET monitoring solutions and identify the differences between previous works in the literature and our proposed scheme.

2.1 Introduction

There is extensive research in the area of MANET management. However, it is noteworthy that a comprehensive monitoring solution is yet to emerge. Such a scheme would need to provide two types of information: (1) network-level information, such as a picture of network topology, and (2) node-level information, such as individual node's cooperation level and level of traffic activity.

MANET nodes often have limited battery power and storage capacity, and wireless radio link capacity and quality varies dynamically. These issues result in a need for paradigms particularly suited to monitoring and managing MANETs.

In this chapter, we survey the relevant work in the area of MANET monitoring and then we review approaches to selfishness detection in MANETs. The cited research serves as background to our work.

2.2 MANET Monitoring

In legacy networks, poll and trap features of the Simple Network Management Protocol (SNMP) are the most common ways of querying network elements regarding their current state. However, in its original format, SNMP is not always practical for mobile ad-hoc networks, where network partitioning is possible and direct access to nodes is not guaranteed. Some research has focused on developing a customized SNMP for ad hoc networks, while others take a different approach.

In [12], Badonnel, Festor and State propose a customized SNMP and management information base (MIB) for ad hoc networks. They propose the design of a MIB that contains ad hoc specific parameters such as NodeAdhocArea, InterNodeAdhocTraffic, NodeCoverage, and NodeRoutingParticipation. The authors suggest a probe-based architecture which is based on (1) a set of ad hoc monitoring probes spread over the network (the range of each monitoring probe defining an ad hoc area) and on (2) a manager that is responsible for constructing the whole network view. An ad hoc probe corresponds to a node that uses packet capture utilities to capture and analyze network messages and construct a partial network view. This local view is then propagated via the customized SNMP to the ad hoc network manager, who builds the whole information model by integrating the information collected from the different ad hoc areas. This architecture allows incomplete coverage of the network, meaning that detailed traffic and cooperation information is only available for nodes that are directly under the coverage of a regional ad hoc probe node.

In [13] the authors propose GUERILLA, a hierarchical monitoring and management system where management agents throughout the network are responsible for collecting information regarding their neighbors and relaying their findings to a network manager. The network manager, after processing incoming information from several management agents, sets network policies and transmits them to the network managers. To gather information and enforce policies on ad hoc nodes, GURERILLA required development and deployment of a new SNMP information base on all ad hoc nodes. Also, since the

detailed design and implementation of the system is yet to emerge, its effects on network bandwidth utilization is unclear.

Another solution that relies on polls and traps for gathering information from a mobile ad hoc network is presented in [14]. In their solution, the authors propose the design of ANMP, an ad hoc SNMP, where a new MIB would be installed on every ad hoc node. Wide deployment of ANMP, which is based on SNMP version 3, would provide specific MANET information to any authorized agent. A monitoring solution based on ANMP would have three-level hierarchy. The first level consists of regular MANET nodes which autonomously form groups based on proximity, reputation and size. Each group selects a leader that acts as the ANMP agent (second level) for the group. ANMP agents use the ad hoc MIB to poll MANET nodes and collect information regarding their status. The agents then transfer their findings to the network manager (third level) where their findings are used to produce a comprehensive report on the monitored MANET.

In another approach to collection information from a MANET network, Belding-Royer, Ramachandran and Almeroth propose DAMON [15], a hierarchical agent-sink architecture for monitoring mobile ad hoc networks. In this scheme, nodes are equipped with a collector module which is responsible for collecting statistics from the MANET by analyzing the routing protocol's topology and traffic control messages. At any time, a few nodes in the network play the role of management agents and, after collecting network information, periodically aggregate and relay their findings to the sink(s). Sinks constantly broadcast beacon messages to declare their presence to the agents. Transmission of monitoring information from nodes to agents and agents to sink(s) is done through the MANET.

A comparison between different approaches to monitoring MANET is presented in Table 2-1.

Features:	ANMP [14]	Probe-Based & MIB [12]	GUERILLA [13]	DAMON [15]	Our Proposed Solution
New software must be installed on all MANET nodes	●		●	●	
Injects management traffic into the MANET	●	●	●	●	
Places constraints on MANET resources (nodes' battery life and CPU consumption)	●		●	●	
Requires development and deployment of a new MIB	●	●	●		
Can provide network topology picture	●	●	●	●	●
Can provide cooperation assessment for nodes under coverage	●	●	●	●	●
Can address MANET partitioning problem					●
Is cheat-proof against a group of malicious or selfish nodes		●	●	●	●
Is tolerant to the loss of some management units		●		●	●

Table 2-1: Comparison of different approaches to monitoring MANETs.

Although monitoring solutions that rely on a middle management layer to collect information from MANET nodes are scalable and can in fact provide node and network-level information, they introduce management overhead to the network. Also, reporting status or responding to polls requires processing by MANET nodes and reduces battery life. Furthermore, these approaches fall short of addressing the problem of network partitioning, where a group of nodes are out of reach of any network manager.

In our proposed solution, however, we do not inject any traffic into the monitored MANET. We employ promiscuous packet capture tools on Monitoring Units (MUs) for gathering information and out-of-band communications for transmitting the findings to one or multiple network manager nodes. Also, because our Monitoring Units collect information passively and are not part of the monitored MANET, network partitioning does not affect the performance of our monitoring solution. Furthermore, we do not require MANET nodes to install new software or run any daemon, hence preserving nodes' limited resources. Our scheme's distributed nature provides scalability; the number of MUs is not predetermined and can increase as the size of the MANET increases.

2.3 MANET Fault Detection

Selfishness, where a node or a group of nodes in a mobile ad hoc network refuse to forward their peers' packets, could significantly affect the performance of a MANET. Many researchers have concentrated their efforts on developing an approach that is reliable and precise in detecting selfish nodes. These solutions can be generally classified into two categories: reputation-based systems [16-19] and credit-based systems [20].

In [20] the authors propose SPRITE, a credit-based approach where each node receives credit for forwarding other nodes' packets and must spend credit if it is the source of the packet. SPRITE incorporates a global Credit Clearance Service (CCS) to manage incentives. In this system, nodes must keep a receipt of every packet they forward and submit these receipts to the CCS to receive credit for their cooperation. Subsequently the CCS debits a node's account when that node is the recipient of other nodes' services.

Because a credit-based approach requires a central clearing service to award and debit credit, it cannot provide adequate solutions for many MANET applications where a connection to a fixed node cannot be guaranteed.

Reputation-based approaches tend to be more suited to the MANET characteristics. In reputation-based schemes, MANET nodes evaluate the behavior of their peers and potentially share the results of their evaluations with one another. Once there is a

consensus that a particular node is a non-cooperative one, all other members of the network avoid requesting or providing services from and to the misbehaving node.

P. Michiardi and R. Molva introduce the COllaborative REputation (CORE) mechanism in [17]. CORE is a protocol for enforcing cooperation between nodes in MANETs. In this approach, each node is provided with a Watchdog (WD) and a Reputation Table (RT) component. In CORE, to assess the performance of a node, the Watchdog component requests a particular service from that neighbor and evaluates its behavior. When misbehavior is detected, the Reputation Table gets updated. The protocol requires sharing of reputation tables among all nodes. Although CORE by itself has not, to our knowledge, been deployed in any real-world system, many other reputation-based systems follow its general approach.

The CONFIDANT protocol proposed by Buchegger and Boudec in [16] is another example of a reputation-based scheme. CONFIDANT consists of four main components: the Reputation System, the Monitor (Neighborhood Watch), the Trust Manager, and the Path Manager. Working together as a system, they monitor and evaluate the behavior of every neighbor, detect misbehavior, rate paths and share this information with other nodes through ALARM messages. The Monitor component, by employing a Watchdog mechanism in promiscuous mode, continuously monitors the behavior of its one-hop neighbors. If abnormal behavior (i.e. non-forwarding) is detected, it is reported to the Reputation System. Depending on the significance and frequency of misbehavior, the Reputation System modifies the rating of the subjected node. Once the rating of a node falls bellow a threshold, control is passed to the Path Manager, which controls the route cache. In such cases, ALARM messages are sent out by the Trust Manager and propagate throughout the network, informing every other node of the newly detected changes.

The TWOACK system, proposed by Balakrishnan, Deng and Varshney in [18], is a link reputation-based system. This system uses a special type of acknowledgment packets called TWOACK packets, which are assigned a fixed route of two hops (or three nodes) in the direction opposite to that of the data packets. When a node that is two hops away from the forwarding node receives a data packet, it sends back a TWOACK packet

carrying the unique ID of the corresponding received data packet. The route for the TWOACK packet is derived from the source route of the original data packet. The purpose of the TWOACK packet is to notify the original source of the packet that the data packet has successfully reached a node that is two hops away. This set of procedures is repeated for every set of three consecutive nodes. If the TWOACK message is not received for a particular packet within a certain period of time, the link between the neighboring node and the two-hop neighbor is marked as a misbehaving link. Over time, any link that has a low cooperation grade will be removed from the list of reliable paths and therefore avoided by nodes in the network.

Baker, Guili and Marti, in [19], introduce PATHRATER, a watchdog-reputation system that works on top of source routing protocols in MANETs. In PATHRATER, each node employs a watchdog mechanism to promiscuously monitor its neighbors and evaluates their behavior in terms of forwarding. If misbehavior is detected, a message is sent to preceding nodes all the way back to the original source of the packet. This way, misbehaving nodes will be avoided when the source routing protocol is choosing a path for the packets. When currently available paths are mostly marked as misbehaving paths, new paths must be discovered to guarantee a certain level of quality of service, requiring the generation of path discovery messages.

Features:	SPRITE [20]	CORE [17]	CONFIDANT [16]	TWOACK [18]	PATHRATER [19]	Our System
New software must be installed on all MANET nodes	●	●	●	●	●	
Injects extra traffic to the MANET	●	●	●	●	●	
Places constraints on MANET resources (node's battery life and CPU consumption)	●	●	●	●	●	
Requires direct access to a central node	●					
Requires cheat-proof software	●	●	●	●	●	
Provides automated detection of misbehavior	●	●	●	●	●	
Can perform in the face of MANET partitioning						●
Is cheat-proof against collusion among malicious or selfish nodes				●		●
Is applicable to all MANET routing protocol types (proactive, reactive)						●

Table 2-2: Comparison of different approaches to misbehavior detection in MANETs.

Although reputation-based selfishness detection systems can successfully detect most selfishness scenarios in a system, they fall short from a few perspectives. Each reputation system requires a new watchdog/evaluator tamper-proof software suite to be installed on all MANET nodes. Also, to monitor and report misbehavior, nodes need to consume their battery, CPU power and bandwidth, which are all scarce resources in a MANET. Moreover, most reputation-based mechanisms are unable to detect group selfishness, where a group of nodes collude to misbehave but not report it to other nodes.

Unlike reputation-based or credit-based cooperation assessment systems, our proposed solution is unobtrusive; it does not inject management traffic into the network, does not need nodes to respond to queries or evaluate their neighbors. Additionally, use of independent and distributed Monitoring Units across the network helps our system to produce the expected results even when there are network partitions, or when a group of nodes collude to misbehave. Furthermore, our proposed scheme is not exclusive to a particular routing protocol and can be implemented based on any MANET routing protocol's specifications.

A feature-comparison between different approaches to selfishness detection in MANETs is presented in Table 2-2.

2.4 Summary

In this chapter, we surveyed the related work in the area of MANET monitoring that provided the background for our research. To better show the relevance of the previous works to our research, we reviewed them in two main categories pertinent to our work: MANET monitoring (Section 1.2) and MANET fault detection systems (Section 1.3). In each section, we explained how our proposed system is related, yet different from the relevant works in this area.

In the next chapter, we will present the detailed design architecture of our proposed solution for monitoring mobile ad hoc networks.

3 Design Architecture

3.1 Introduction

Mobile ad hoc networks present a different set of characteristics than those of fixed and managed wireless networks. A MANET has virtually no limitations in terms of geographical area. Also, network nodes in a MANET can be highly mobile, causing an unstable and rapidly changing topology. Although these characteristics are among advantages of MANETs, they also present challenges in the monitoring and management of mobile multi-hop networks.

The ability of the network manager to determine the true topology of a MANET at all times requires a carefully designed monitoring system with a robust structure. Due to the unstable nature of these networks and rapid changes in link connectivity, network partitioning is a common problem. When network partitioning happens in a network deployed in a geographically dispersed area, there remain few choices as to how to observe and collect enough information to produce a comprehensive network topology picture. Our solution relies on multiple monitoring stations that collaborate and combine information to maintain an accurate and up-to-date view of the current topology.

Furthermore, individual nodes' level of cooperation in forwarding packets for their neighbors is also valuable information for the MANET manager. Ad hoc networks function based on the cooperation and collaboration between nodes. In such networks, each node is responsible for forwarding packets originated by any of its peers. Forwarding is the most fundamental service that each node ought to provide in such networks. However, forwarding should not be taken for granted, as some nodes might act selfishly in order to preserve their resources such as battery life or CPU consumption. Therefore there needs to be a method to detect such selfish nodes in the network.

Moreover, access to statistics regarding node-level traffic activity, i.e. received traffic rate and transmitted traffic rate, is imperative for a management solutions in providing quality of service, load balancing, congestion control, fault detection, etc. For example, when a node is performing at its 100% capacity, the management solution can instruct the

neighboring nodes to avoid directing any extra traffic towards it, hence avoiding loss of traffic.

Our monitoring solution allows the network manager to identify nodes that are overwhelmed or appear to be behaving uncooperatively. Contrary to other methods for detecting selfish behavior, like reputation-based systems [13, 16, 17],or path-rating systems [19], our proposed solution does not require tamper-proof software or secure hardware to be installed on all the nodes. It also does not inject extra traffic to rate the node's traffic and cooperation level and is completely independent from the network elements.

3.2 High Level Design of the System Architecture

Considering the problem of undefined geographical coverage, possibility of network partitioning and highly changing topology, we propose a distributed nature for our MANET monitoring system. In our proposed solution, a number of Monitoring Units (MUs) are deployed throughout the MANET. These MUs are responsible for collecting information regarding network behavior (network topology picture, link changes, etc.).

The information collected by the Monitoring Units is delivered to a management node, where they are consolidated, analyzed and presented on an easy-to-understand graphical user interface (GUI). Both network-level information and node-level information are presented in the GUI.

Our solution does not assume complete coverage of the network and therefore is well suited to mobile environments. Moreover, our proposed scheme does not inject any additional traffic into the network. The interaction between the Monitoring Units and management unit take place out-of-band, resulting in no overhead to the monitored network. Figure 3-1 represents an overview of the structure and a sample scenario for our MANET monitoring solution.

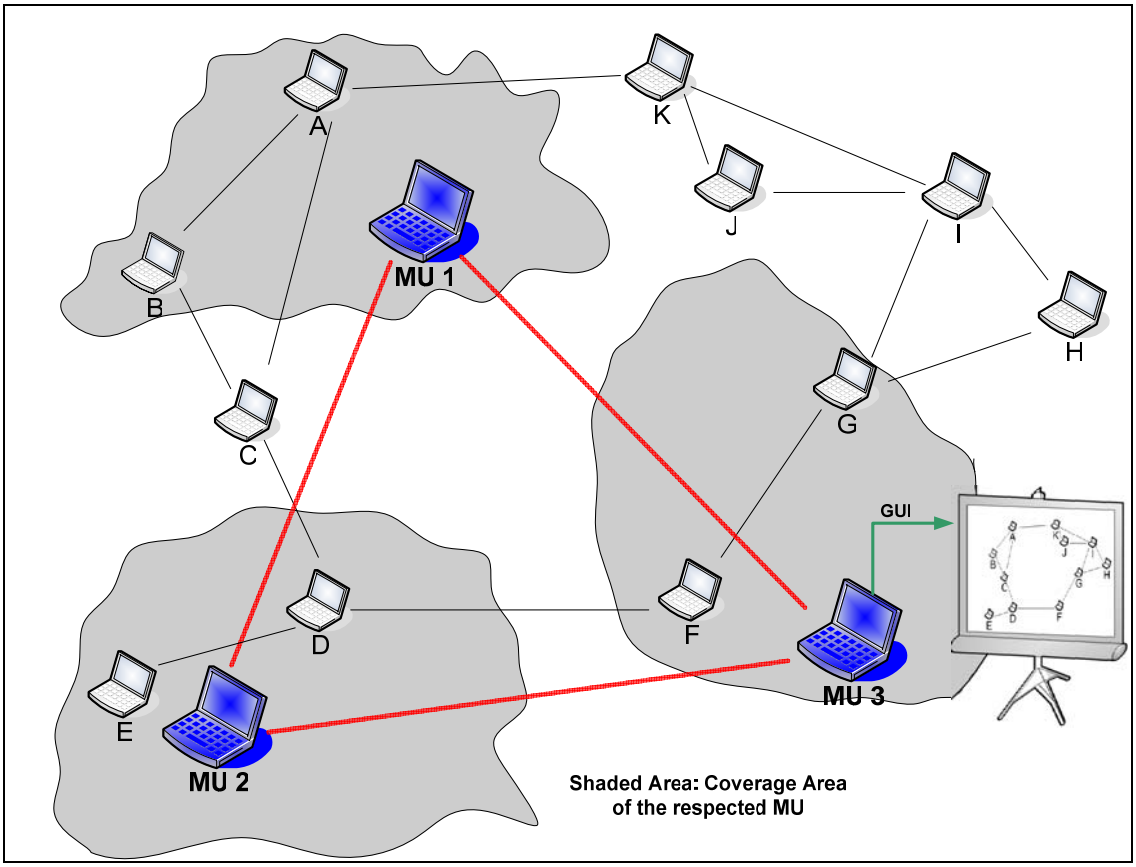

Figure 3-1: Multiple Monitoring Units (MUs) collect information and share their findings. All findings are consolidated to produces a comprehensive network assessment report.

The final picture (shown in Figure 3-1 as the GUI output) produced by the management unit is the result of consolidation and analysis of all incoming information from all Monitoring Units, providing a comprehensive picture of the network-level and node-level information, even in partitioned networks. Moreover, the results are updated dynamically and reflect changes in link connectivity, node cooperation and traffic load at each node with only a few seconds of delay.

Considering the operational environment for this system - large and unpredictable geographical coverage area, incomplete coverage of the MANET and passive nature of monitoring and data collection – the system was designed to have two major independent

components. A Capture component and an Analysis & GUI component are installed on Monitoring Units and the management unit(s), respectively. Use of two independent components provided a distributed system structure, an important feature of our proposed solution.

The Capture component, deployed on Monitoring Units across the network, has the responsibility of sniffing packets, analyzing them and summarizing the findings in output files, called Info Files. The Analysis & GUI component receives the Info Files produced by Monitoring Units, processes them and presents the final outcome on a dynamic GUI.

To manage the delivery of Info Files from the Capture component to Analysis & GUI component, a workflow is designed, called File Delivery unit. This unit, running on all management nodes, manages the sharing and distribution of Info Files between the two components.

3.3 Components

3.3.1 Capture Component

To collect information regarding the behavior of the MANET, we had two options to consider: using polling/trap to collect information directly from each node; or promiscuously capturing packets off the air and analyzing them to obtain the specific desired information.

The Capture component was designed solely for collecting desired information from the network under surveillance. It was our design goal to avoid injecting any extra traffic into the monitored network and therefore the Capture component is designed to capture packets in a passive and promiscuous mode, and to analyze them to extract the desired information regarding the topology, traffic, and cooperation. Moreover, because of possibly high packet rate, the Capture component must have a simple logic so that it will require as little processing power as possible.

The Capture component should aggregate its findings in as little space as possible into output files, called Info Files. Small-sized files need less storage capacity and less

bandwidth when they are being pushed to the Management Units by the File Delivery Unit.

3.3.1.1 Promiscuous Mode Capturing

To avoid injection of any extra traffic into the network, and also to achieve passive monitoring, the Capture component captures network traffic in promiscuous mode. Promiscuous packet capturing allows the system to intercept and log network traffic. In this method, the network interface card must be configured with the targeted network's netmask and does not need to be a participating node of the monitored network.

3.3.1.2 Extracting Information

In any wireless network, there are several types of packets being exchanged at any given time (which in turn can be captured in promiscuous mode). Examples include topology control messages, routing information, regular data packets, ARP requests and responses, ICMP packets, etc.

To extract the required information from the packets being exchanged over the wireless medium, and also to achieve processing efficiency, a simple categorization is the first thing applied to each incoming packet. Then, based on the type of the packet, the packet is processed to extract the intended information from it.

The Capture component identifies individual nodes in the network and discovers the connectivity between them by processing routing protocol control messages.

Also by sniffing all UDP and TCP traffic, the system is able to assess the absolute traffic load on each node as well as its cooperation level in terms of forwarding other nodes' packets.

Our method for determining the level of cooperation of each node is based on two counters: The number of packets that are *expected* to be forwarded and the number of packets that are *actually* forwarded by the node.

The cooperation information, like all other types of information, is dynamic and is periodically updated. We note that, for a node that has fallen outside the coverage area of any of the monitoring units, we will not have traffic load or cooperation information.

3.3.1.3 Summarization of the Findings

After each *Capture-Period* (a configurable value), The Capture component produces an Info File, containing a summary of its findings. The format of these output files must be efficient and known to the Analysis & GUI component. Small sized output files are desired in order to facilitate quick and efficient sharing of them between Monitoring Units and management nodes.

3.3.2 Analysis & GUI Component

The Analysis & GUI component takes in the Info Files that are generated by the Capture components deployed across the network, and after processing them produces a dynamic GUI showing network topology, node connectivity and node traffic information, as well as an assessment of each node's level of forwarding cooperation.

3.3.2.1 Network Level Information

One of the goals of our monitoring system is to present a dynamic picture of the MANET, showing the nodes and the links between them. In the initialization of the GUI, each node is assigned a random location on the canvass and the links between them are drawn based on the connectivity information reported in the Info Files.

However, all this information is designed to be short-lived so that the GUI represents an updated and dynamic picture of the network at all times. In other words, if a node or a connection link is not reported for a period of time (t) which is greater than its validity time (vt), it is erased from the GUI. Therefore, if a node or a connection between two nodes does not recur for a certain time, they must disappear from the GUI. This ensures that the GUI shows the changes in the MANET as they happen.

3.3.2.2 Node Level Information

In addition to the overall picture of the network topology, the GUI also presents node-level information such as node cooperation level and node traffic load. This information also needs to be shown in a dynamic manner. Like network-level information, if the information about a node traffic or behavior is not reported by the Info Files for a certain time (vt), it must be erased from the GUI. This ensures that what is shown on the GUI is an up-to-date representative of what is going on in the network.

3.3.3 File Delivery Unit

As described in High Level Design section of this document, the Info Files that are generated by the Capture component must be fed into the Analysis & GUI component for further analysis, consolidation and production of the final outcome. These two main components generally run on different hosts. Therefore, there is a need for a resilient and efficient module to deliver the files from one component to the other.

A file delivery module, running on all Monitoring Units and using out-of-band communication link, assumes the responsibility of pushing the locally generated Info Files to the desired directory in the Management Unit(s) where the Analysis & GUI component is running.

3.4 Drawing the Topology

The system must keep track of what is currently being displayed on the GUI and what needs to be changed or erased. To achieve this goal, every node and every communication link is assigned a valid time (*vt* = CurrentTime + ValidityTime). Every time this information is found in an Info File, the *vt* gets updated. These *vt* values are constantly checked, and if the information is expired (CurrentTime > *vt*), the expired information must be erased from the GUI canvass.

3.5 Traffic Activity and Cooperation Assessment

To report the traffic activity and represent the cooperation level of each node, each Info File contains four sets of information for each node under the coverage:

1. The number of data packets originated by the node.
2. The number of data packets destined (sinked) to the node.
3. The number of data packet *expected* to be forwarded by the node.
4. The number of data packets that the node *actually* forwarded for other nodes.

This information will be processed in the Analysis & GUI component to assess whether a node has been cooperating in forwarding packets for its neighbors and to report its traffic activity.

Cooperation level at each node is calculated using two numbers: *actually* vs. *expected* forwarded packets:

$$Cooperation\ Percentage = \frac{Number\ of\ \ Actually\ \ forwarded\ \ packets}{Number\ of\ \ packets\ Expected\ \ to\ be\ \ forwarded}$$

Traffic load information is shown as the rate of incoming traffic to a node (sum of the packets that are expected to be forward plus packets that are actually destined to the node) and outgoing traffic (sum of the packets that the node has generated plus the packets it has forwarded for others).

Like the topology related information, only recent traffic and cooperation information is displayed by the GUI, and old information is deleted as it exceeds its validity time.

3.6 Modularity of the Design Structure

As described in this chapter so far, the overall architecture of our system is flat (no explicit hierarchy) and distributed. One component can be launched in one host while the other one may be running in another host.

In other words, although the two components need to work with each other to produce the desired outcome, they are independent from each other. This means that if a Monitoring Unit goes down or loses its connection with management nodes, it does not result in failure of the system. The system functions as long as there is at least one of each components running at one or multiple hosts. Hence the stability of the system does not depend on any particular node.

Also, because of this loosely coupled connectivity between Monitoring Units and Management Nodes, the system is easily scalable. In other words, as the size of the network increases/decreases, it is possible to easily add/remove Monitoring Units and maintain the same overall system performance.

3.7 Summary

In this chapter we detailed the design objectives and architecture of our proposed scheme for monitoring mobile ad hoc networks. In Section 3.2 we presented an overall architecture of our system. Then in the proceeding parts of this chapter we offered more insights to the overall design structure of our system, explaining the role of each component as well as methods to collect and interpret network information.

The next chapter explains the detailed structure design and implementation of our proposed solution, using Java for developing the system components. Optimized Link State Routing (OLSR) protocol was used as the underlying routing protocol for the MANET.

4 Implementation Architecture

4.1 Introduction

The majority of the research in the area of MANET management and security assessment systems relies on simulation. Simulation is an option in the development and validation of wireless network solutions and in many cases the only viable choice. Simulation schemes provide a controlled environment where the conditions can be modified easily. However, there are limitations and shortcomings associated with solutions that are solely designed and validated by simulation. Since in a simulation it is usually infeasible to account for all factors that could be present in a real wireless environment, simulation results are prone to be inexact and sometimes invalid.

Because of the weaknesses associated with simulation-driven solutions, we focused on implementing our proposed solution in a real network environment.

The two main components of the system, the Capture component and the Analysis & GUI component, were implemented using the Java programming language in a Linux environment (Fedora Core 4 and 5, Slackware Linux 10.2).

Use of Java helped us to take advantage of the JPCAP packet capture suite [21], a SourceForge open source project that aims at providing toolsets for capturing network packets.

The Capture component was implemented to match the specification of the Optimized Link State Routing (OLSR) protocol (NRL implementation) [22].

The object-oriented nature of Java helped us to efficiently manage the complex task of information analysis in the Analysis & GUI component. In the Analysis & GUI component, new Java classes were introduced, while a few classes were borrowed from the JPCAP set of classes. Table 4-1 contains the list of Java classes and a brief description of them. The objective of the Analysis & GUI component is to deliver an easy-to-understand GUI which not only is accurate in representing the status of the network and nodes, but is also dynamic and able to reflect changes in the network with

less than a few seconds' delay. Figure 4-1 shows how different Java classes in the Analysis & GUI component collaborate to create and maintain a dynamic GUI.

Global Classes	Description
OLSRLogMsg	New Class. Creates log files, logs error messages, warnings and progress information.
OLSRConfigSpecs	New Class. Loads up the OLSR.conf file that provides certain configuration settings.
Capture Component Classes	**Description**
CaptureComponent	New Class. Invokes the *PacketHandler* class to capture packets, extracts information from captured packets and creates Info Files.
PacketHandler	JPCAP Class. Called by the *CaptureComponent* class to capture a packet and deliver a copy of it to the *CaptureComponent* class.
Analysis&GUI Component Classes	**Description**
AnalysisAndGui	New Class. It is the *Analysis&*GUI daemon. Launches the *FileProcess* class to process Info Files.
FileProcess	New Class. Processes each Info File, merges and consolidates information gathered by multiple Monitoring Units. Creates several hashes and passes them to the *CaptureViewFrame* class.
CaptureViewFrame	Modified JPCAP Class. Takes in several hashes from the *FileProcess* class and, after analysis, passes node-level information to the *HostRenderer* class and network-level information to the *CommRenderer* class.
HostRenderer	Modified JPCAP Class. Takes in data hashes from the *CaptureViewFrame* class that contain information regarding nodes, traffic and cooperation information. Draws/erases the node-level information on the GUI.
CommRenderer	Modified JPCAP Class. Takes in data hashes from the *CaptureViewFrame* class that contain information regarding connection links between nodes. Draws/erases the network-level information on the GUI.
PacketVisualizationCanvas	Modified JPCAP Class. Initially launches the GUI canvass. After the initialization of canvass, it is constantly updated and maintained by the *HostRenderer* and *CommRenderer* classes.
CaptureViewMouseListener	Original JPCAP Class. Enables mouse action on the GUI.

Table 4-1: Java classes used in the implementation of the system.

4.1.1 JPCAP

JPCAP is a set of Java classes which provide an interface for network packet capture and real time traffic monitoring. JPCAP utilizes Libpcap [23], a widely deployed packet capture library. Use of JPCAP for packet capture and analysis allows us to leverage both Java's object oriented programming interface and Libpcap's highly scalable packet capture capabilities. Additionally, JPCAP implementation is platform-independent and well documented.

4.1.2 OLSR

Optimized Link State Routing protocol (OLSR) [24] is a proactive routing protocol that limits the control overhead through its optimized flooding techniques and compact control messages. To provide adequate performance in an ad-hoc mobile environment, OLSR constrains the set of nodes that generate and forward messages containing topology information. Each node that serves as a multipoint relay generates Topology Control (TC) messages in which it declares currently active links. In OLSR, TC messages propagate throughout the network and are delivered to each node for determining the next hop towards each possible destination. All nodes advertise their existence (as well as their known neighbors) through the use of periodic beacons, commonly referred to as HELLO messages. Nodes broadcast these packets to inform one-hop neighbors of their status and to provide state information to their neighbors. However, HELLO messages are not propagated and can only be heard by one-hop neighbors of their originator. Throughout our development and testing stages, we used OLSR as the underlying routing protocol for the monitored MANET network.

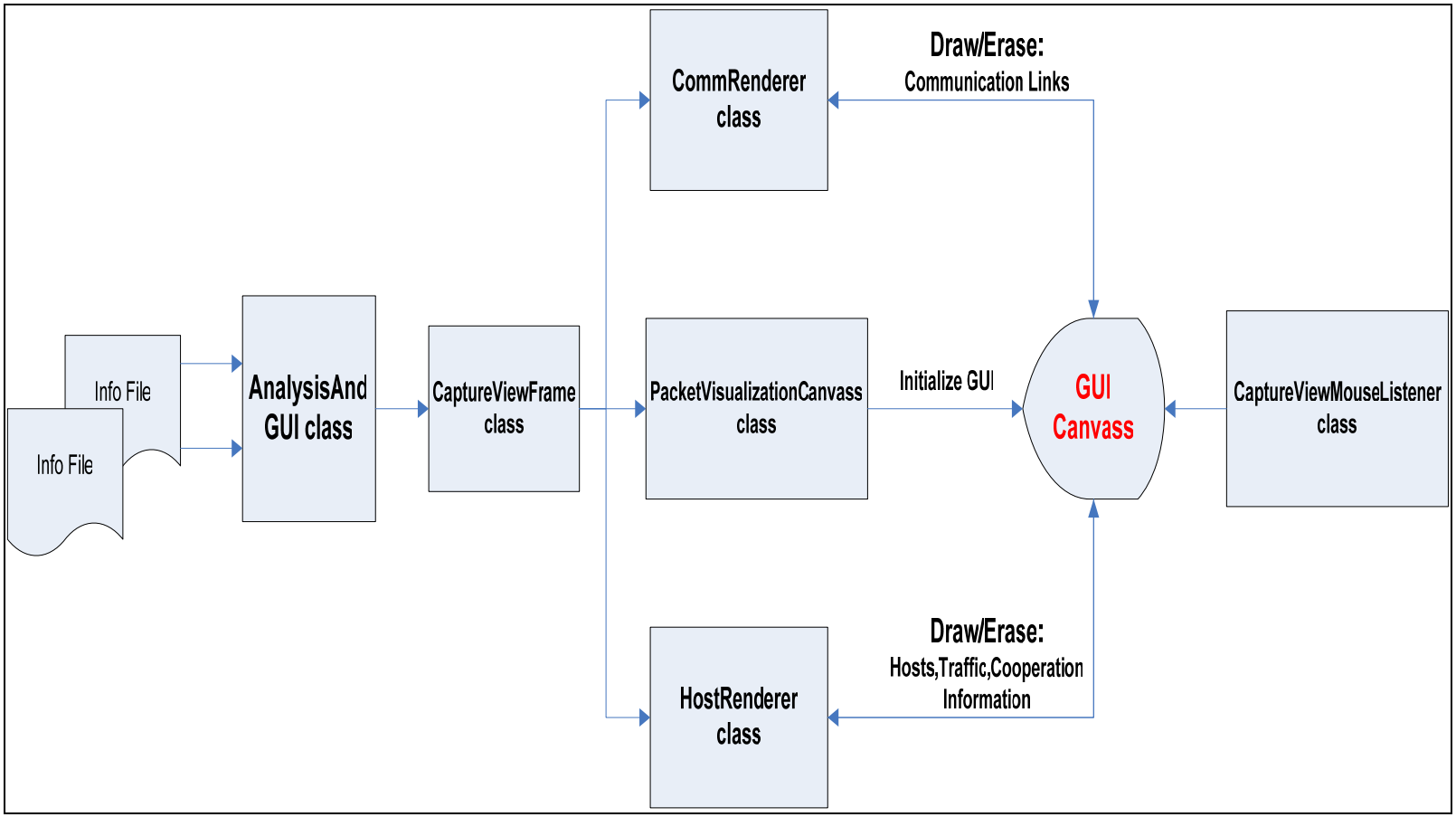

Figure 4-1: Creation and maintenance of GUI by Java classes in the *Analysis & GUI* component.

4.2 Overview

In the overall design architecture of our system, one of our objectives was that the targeted multi-hop network, which is being monitored by our system, should be completely unaware of the presence of our monitoring system. In other words, because our system is unobtrusive and collects its information passively, its existence should have no effect on the performance of the network under surveillance. Also, although our implementation assumes the use of OLSR, the system can easily be modified to work with other MANET routing protocols.

Achieving good performance in the Capture component requires simplicity in its design and care in its implementation. Efficient logic for the Capture component enables the system to capture and analyze a high volume of packets and therefore report accurate statistics of traffic activity even in networks with high traffic activity. Moreover, adept logic for capturing packets results in relatively low CPU consumption, hence more efficiency and less battery power requirement for the system.

4.3 Implementation Architecture

4.3.1 Overall Flow

The system has the following overall flow:

1. The Capture component monitors the targeted network passively and, after analyzing the routing protocol-related messages and IP traffic, it produces Info Files.
2. The File Delivery module periodically delivers the Info Files produced by the Capture component to the Analysis & GUI component.
3. The Analysis & GUI component receives the Info Files generated by one or several Monitoring Units, processes them and consolidates their findings. It then shows the latest findings on the GUI. It continuously updates the GUI based on the latest findings, supplemented by incoming Info Files.

As shown in Figure 4-2, different components of the system may be running on different computers. However, whether both the Capture component and the Analysis & GUI component are running on the same machine or on different nodes, the File Delivery module is responsible for delivering all the Info Files produced by every Capture component to the targeted Analysis & GUI component.

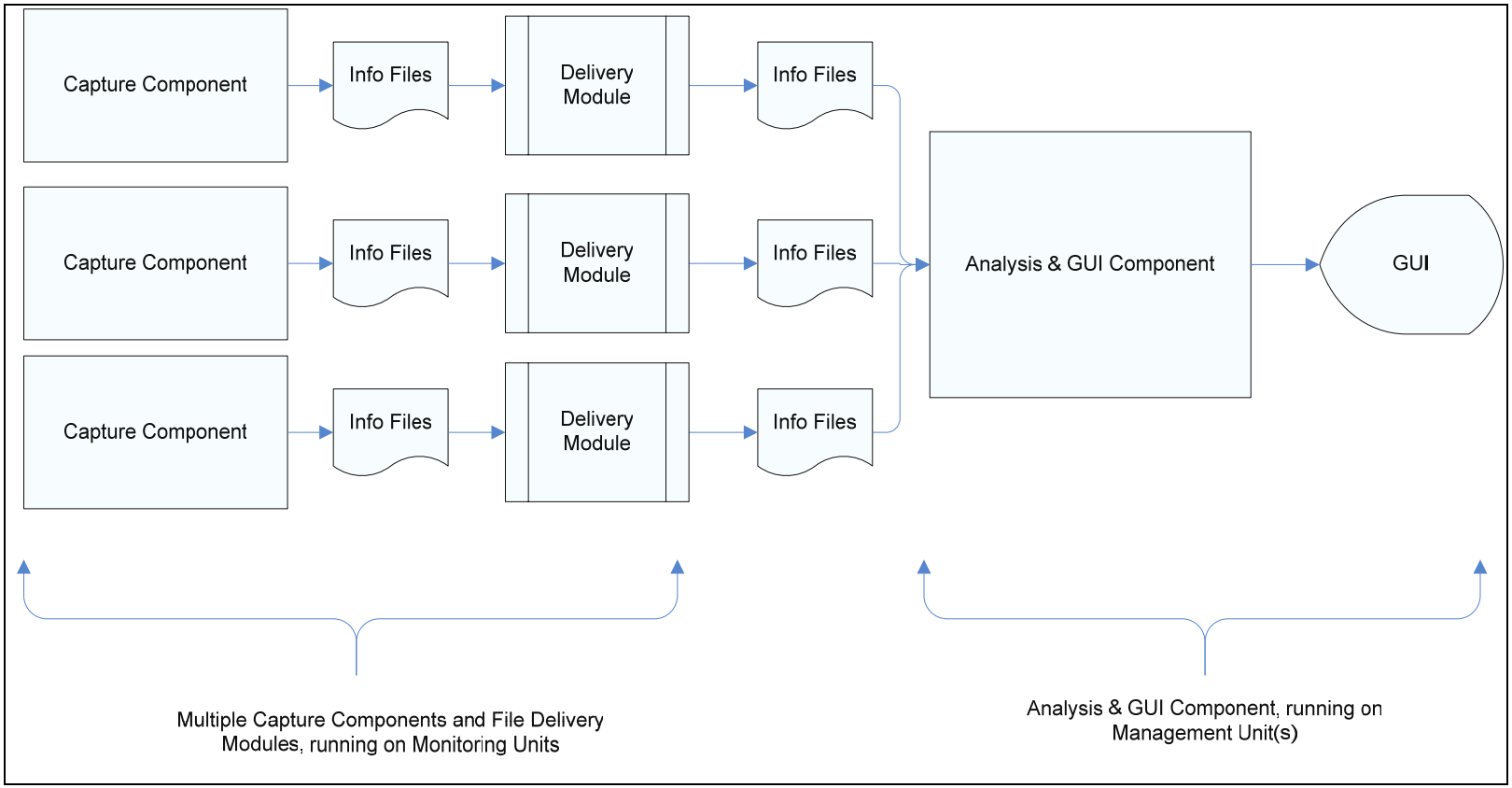

Figure 4-2: The overall flow of the system.

4.4 Capture Component

The Capture component is generally responsible for capturing packets, extracting the required information from them and summarizing its findings in output files, called Info Files.

The Capture component is implemented entirely in Java and uses JPCAP packet capture capabilities for capturing packets in promiscuous mode. In this component, each captured packet goes through a filter and, based on its type, it is passed through particular logic. The detailed flow structure of the Capture component is shown in Figure 4-3.

4.4.1 Collecting Information

Many network management systems use Simple Network Management Protocol (SNMP) [25] trap and polling functionalities to collect information directly from each node in a computer network. In SNMP, an authorized agent is able to directly inquire a node about its health, connectivity level and traffic-related information.

However, there are several limitations in the use of SNMP in a MANET [26, 27]. Use of SNMP introduces overhead and increases the competition for limited available bandwidth [27]. It also requires activation of the SNMP daemon and maintenance of the Managed Information Bases (MIB) on each node. These result in higher storage capacity and CPU processing power requirements for MANET nodes [28]. Moreover, SNMP is designed to work with centrally managed networks. In such networks, the SNMP agent has guaranteed access to all nodes through access points, switches, and routers [27]. However, in multi-hop mobile networks, there is no central node with a direct connection to all other nodes. This problem could be partly mitigated by using different layers of management nodes, like management agents, which are spread across the network and collect information from nodes under their coverage [12]. However, there is no guarantee that all nodes in a MANET would be covered by these agents at all times.

The research to design and implement a customized SNMP for multi-hop ad hoc networks, known as Ad Hoc Network Management Protocol (ANMP) [14], is under way and is yet to be completed and widely deployed.

Our solution however, takes a different approach to collect information from the monitored MANET. In our scheme, we promiscuously capture protocol control traffic and, by analyzing this traffic, we extract network-level information such as the MANET topology picture. Also, by sniffing the wireless medium for data packets, our system is able to determine the cooperation and traffic activity level of nodes under the coverage of Monitoring Units. Our approach does not require complete or constant coverage of the entire network to achieve its goals. Moreover, it does not require additional hardware or software to be installed on the MANET nodes and does not depend on the development of any new global schemes such as ANMP.

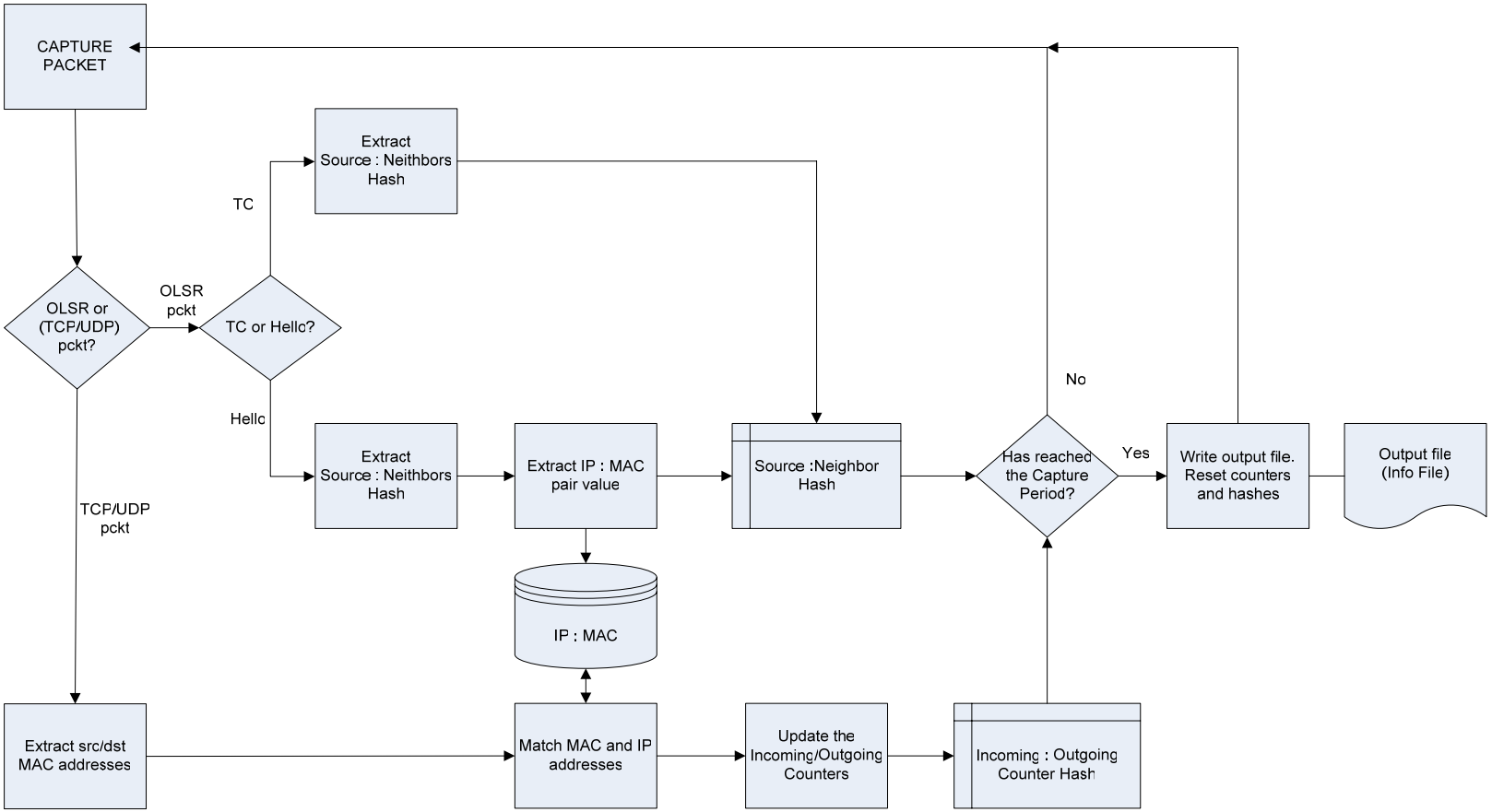

Figure 4-3: Overall flowchart of the Capture component.

4.4.2 Promiscuous Packet Capture

Promiscuous packet capturing is implemented by using the PacketHandler class of the JPCAP suite. An instance of this class is initiated in the Capture component. When PacketHandler is initiated, it launches a PacketListener object that listens for packets in the network interface card of the Monitoring Unit. When a packet arrives at the network interface, a copy of it is sent to the PacketHandler and our Capture component can then process that packet.

4.4.3 Packet Processing

When the Capture component is first launched, it loads up a configuration file that contains configurable values:

CapturePeriod: The time period (in ms) during which the Capture component collects information from the network. After each CapturePeriod, the component writes all its findings to an Info File, and all HashMaps and variables are reset to null.

SharedDirectory: This is the directory where the output files (Info Files) are stored.

4.4.4 Topology Extraction

After the PacketHandler object delivers a copy of a packet to the Capture component, it first determines whether it is an OLSR control packet. OLSR packets are transmitted using port 698, and therefore a simple getSrcPort() method is called on the packet header to determine whether it is an OLSR packet.

If a captured packet is an OLSR packet, the Capture component further analyzes the packet to determine whether it is a Topology Control (TC) message or a HELLO message. Then, based on the packet type, the information regarding the source node and the list of its neighboring nodes is extracted and stored in a HashMap, called nodeNeighbor, with the format of:

< key = Source Node IP , value= neighbor list >

Topology Control (TC) messages and HELLO messages have different formats. Figure 4-4 shows a TC message format and Figure 4-5 shows a HELLO packet structure.

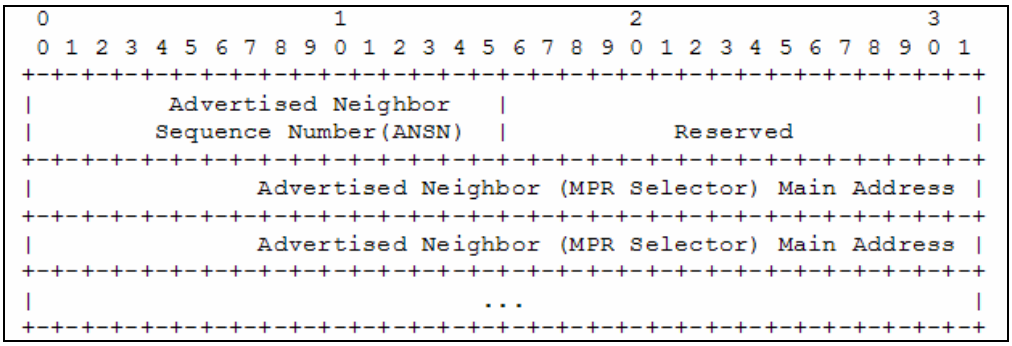

Figure 4-4: OLSR Topology Control (TC) message format.

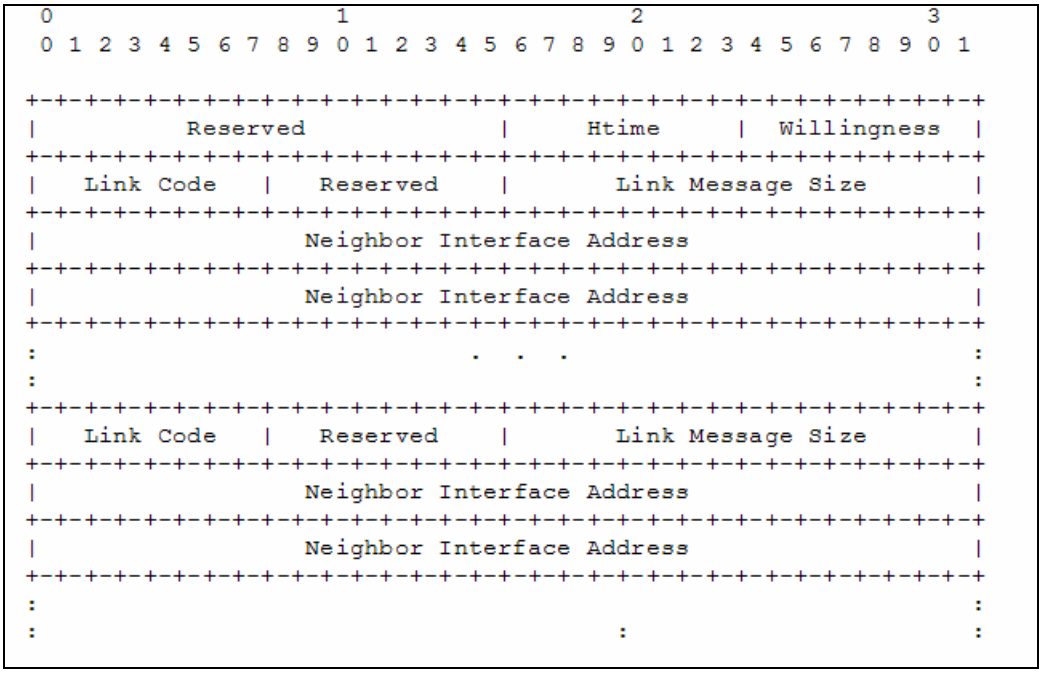

Figure 4-5: OLSR HELLO message format.

The OLSR TC messages are generated only by MPR nodes and are propagated throughout the network. However, if only TC messages are processed by the Capture component, some links may remain undiscovered, as these messages do not contain

information regarding links between non-MPR nodes. For example, in Figure 4-6, processing TC messages generated by nodes 2 and 4 does not reveal links 3-5 and 1-3.

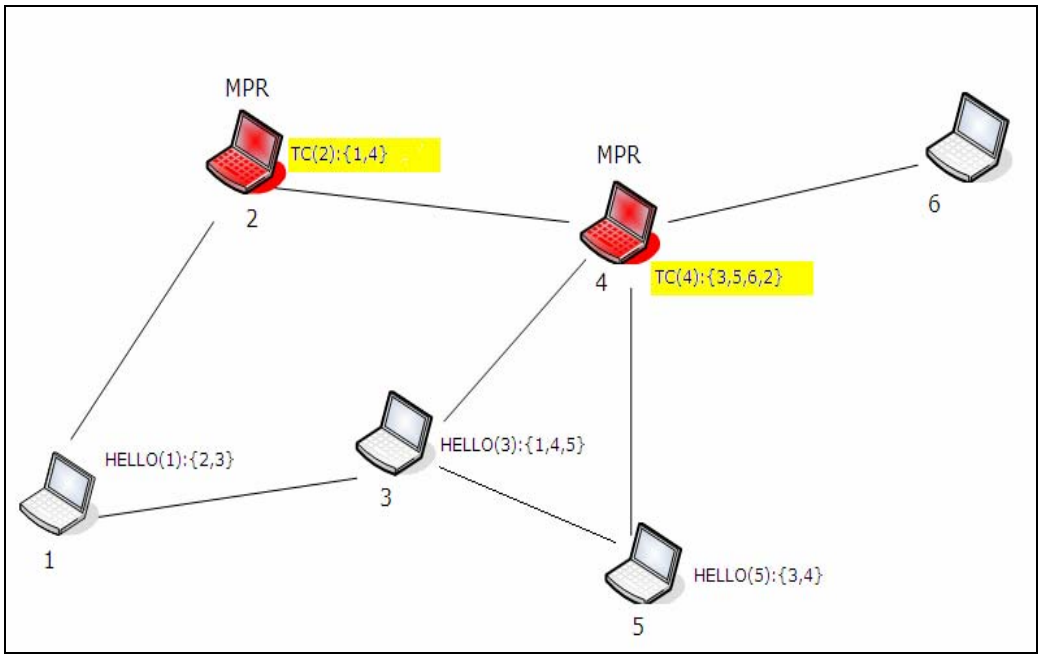

Figure 4-6: An example scenario where processing TC messages does not reveal ALL links.

Hence, to get a more complete picture of the network topology, the Capture component also processes HELLO messages where they are available. For the above example, a HELLO message that is generated by node 3 declares nodes 1 and 5 as its neighbors.

However, HELLO messages are not propagated throughout the network and can only be captured by the nodes which are within range of at least one of the Monitoring Units. As would be expected, this results in more complete network topology information for parts of the network that are under direct coverage of Monitoring Unit(s) than to parts that are not directly within reach of MUs.

Processing HELLO messages also helps in establishing a database for matching MAC addresses with IP addresses. This database helps us in traffic analysis, explained in the next subsection.

Figure 4-3 shows the overall flowchart of the Capture component.

4.4.5 Traffic Activity Reporting

In multi-hop networks each node acts as a router for its peers. The Capture component is designed to report four sets of values for each node:

1. The number of data packets originated by the node, maintained by the *TrafficSource* counter.
2. The number of data packets destined (sinked) to the node, maintained by the *TrafficSink* counter.
3. The number of data packet ***expected*** to be forwarded by the node, maintained by the *ForwardIN* counter.
4. The number of data packets that the node ***actually*** forwarded for other nodes, maintained by the *ForwardOUT* counter.

This information will be processed in the Analysis & GUI component to assess whether a node has been cooperating in forwarding packets for their neighbors and to report its traffic activity.

The counters described above are incremented based on the following:

- If the source MAC address in the frame containing the IP datagram matches the source IP address, the packet has been originated at this MAC address (*TrafficSource* += 1). Otherwise, this packet has been forwarded by the node with this MAC address (*ForwardOUT* += 1).
- If the destination MAC address in the frame containing the IP datagram matches the destination IP address, the packet is destined to this MAC address (*TrafficSink* += 1). Otherwise, this packet is ***expected*** be forwarded by the node with this destination MAC address (*ForwardIN* += 1).

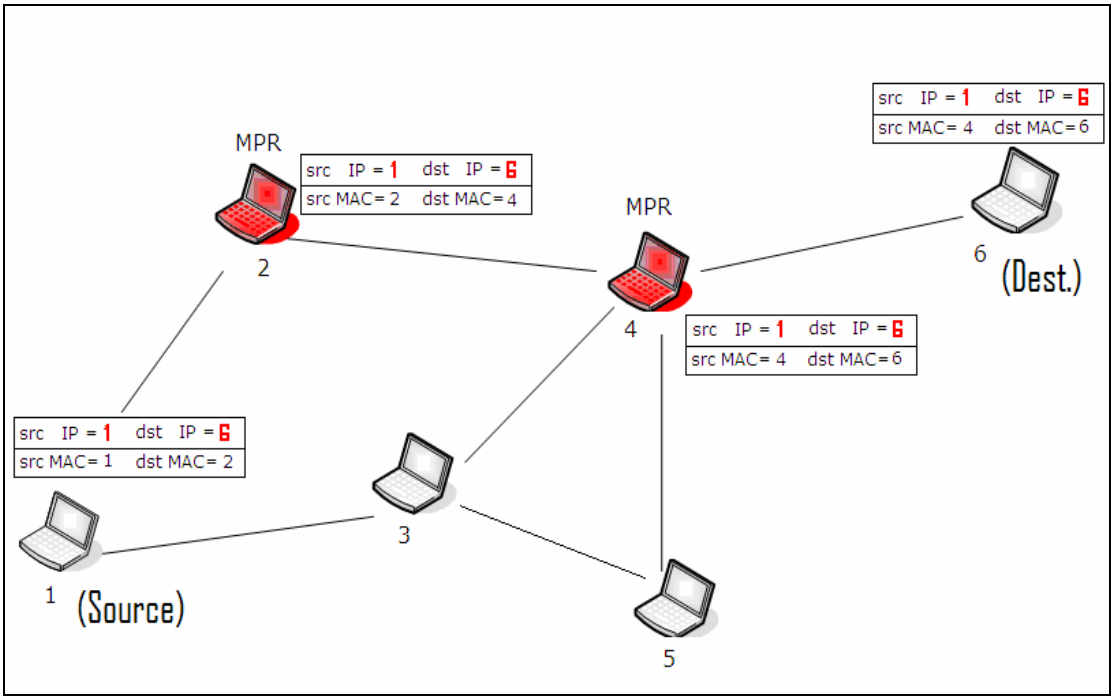

Figure 4-7: Src/Dst IP address of a packet remains unchanged as it gets forwarded.

For the scenario shown in Figure 4-7, when *n* packets have been generated by node 1 and delivered to node 6, the traffic counters will be reported as follows (assuming 100% forwarding cooperation at nodes 2 and 4):

Node 1: *TrafficSource=**n**, TrafficSink=0, ForwardIN=0, ForwardOUT=0*

Node 2: *TrafficSource=0 , TrafficSink=0, ForwardIN=**n**, ForwardOUT=**n***

Node 4: *TrafficSource=0 , TrafficSink=0, ForwardIN=**n**, ForwardOUT=**n***

Node 6: *TrafficSource=0 , TrafficSink=**n**, ForwardIN=0, ForwardOUT=0*

Figure 4-8 shows how the Capture component maintains these four counters.

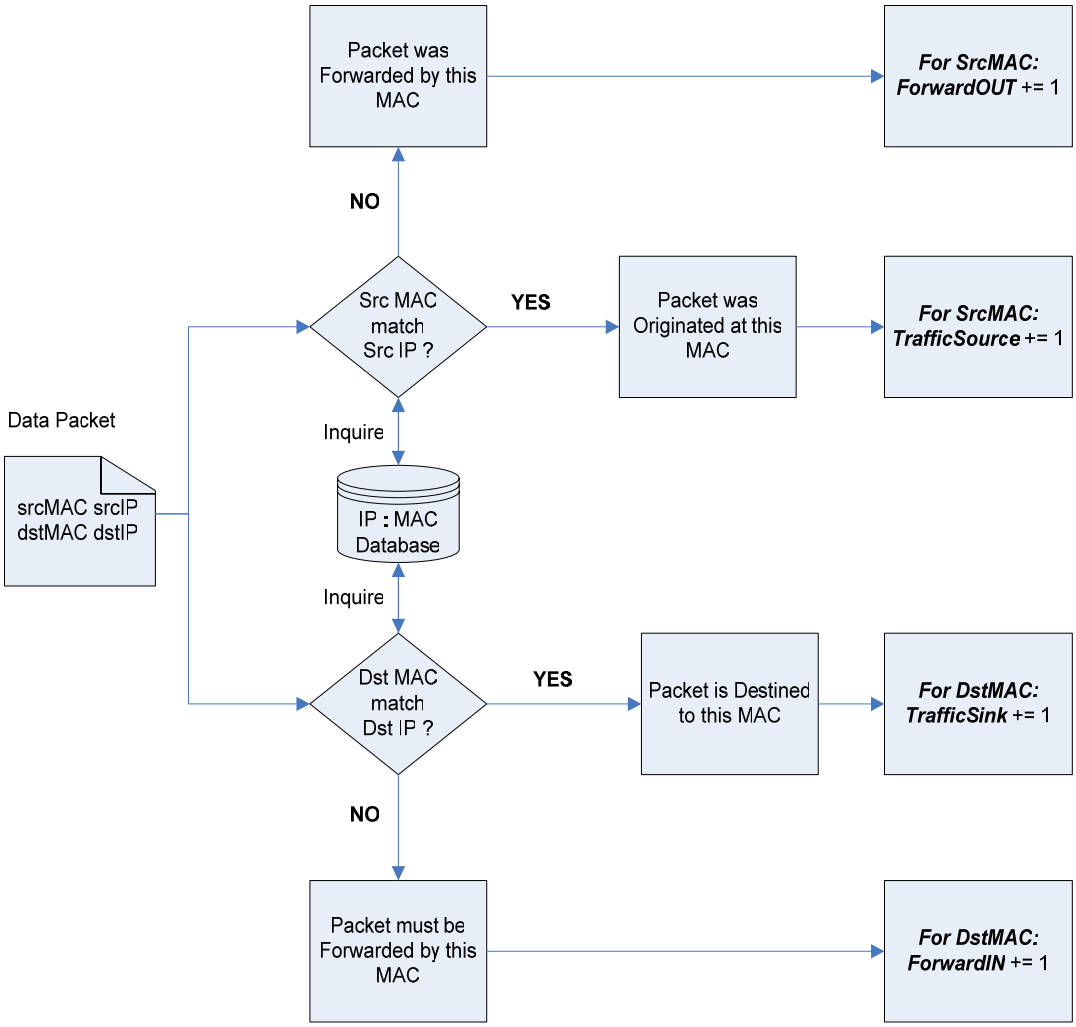

Figure 4-8: Capture component assesses cooperation and traffic activity by using 4 counters.

4.4.6 Reporting the Findings

During each *CapturePeriod*, the Capture component collects data traffic information and topology-related information from the MANET under surveillance. Topology-related information is maintained in the *nodeNeighbor* HashMap and data traffic information for each node is kept in the *nodeTraffic* HashMap.

At the end of each *CapturePeriod*, the *writeOutput()* method of the Capture component is called. This method goes through both the *nodeNeighbor* HashMap and the *nodeTraffic* HashMap and writes them to an Info File. Then the Capture component resets its HashMaps to *null*, resets its timers and starts over.

Because these files get transferred from the Monitoring Units to where the Analysis & GUI component is running, the information in Info Files is aggregated to conserve bandwidth as well as storage.

The format for a line containing topology information is:

Source_IP **:** *type* **:** *neighbor1* **:** *neighbor2* **:**...

The format of a line containing traffic activity information is:

Node_IP **:** *type* **:** *Incoming_Counter* **:** *Outgoing_Counter* **:** *CapturePeriod*

type = 1 indicates that the information comes from a HELLO message generated by a node under coverage of the Monitoring Unit.

type = 2 indicates that information comes from a TC message generated by an MPR node. This will help to mark the MPR nodes on the GUI.

type = 3 indicates that this line contains traffic-related information.

Figure 4-10 shows a sample Info File for the network in Figure 4-9, where node 10.10.0.6 is not under the direct coverage of the Monitoring Unit.

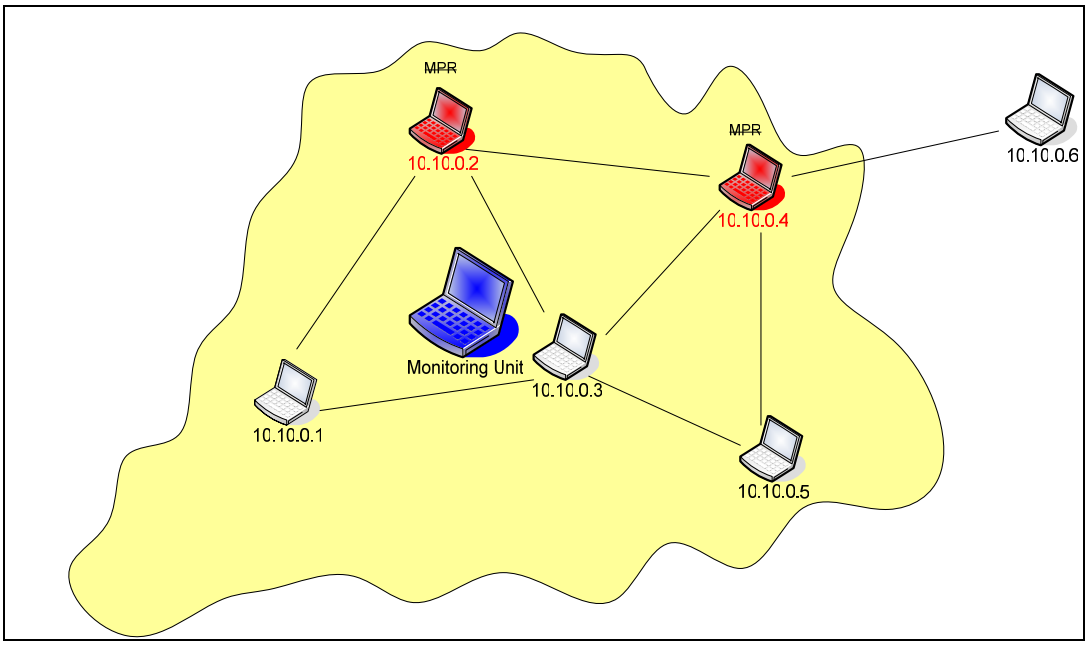

Figure 4-9: A Sample Scenario for a Monitoring Unit (blue). MPR nodes are in red. The yellow area is the coverage area of the MU.

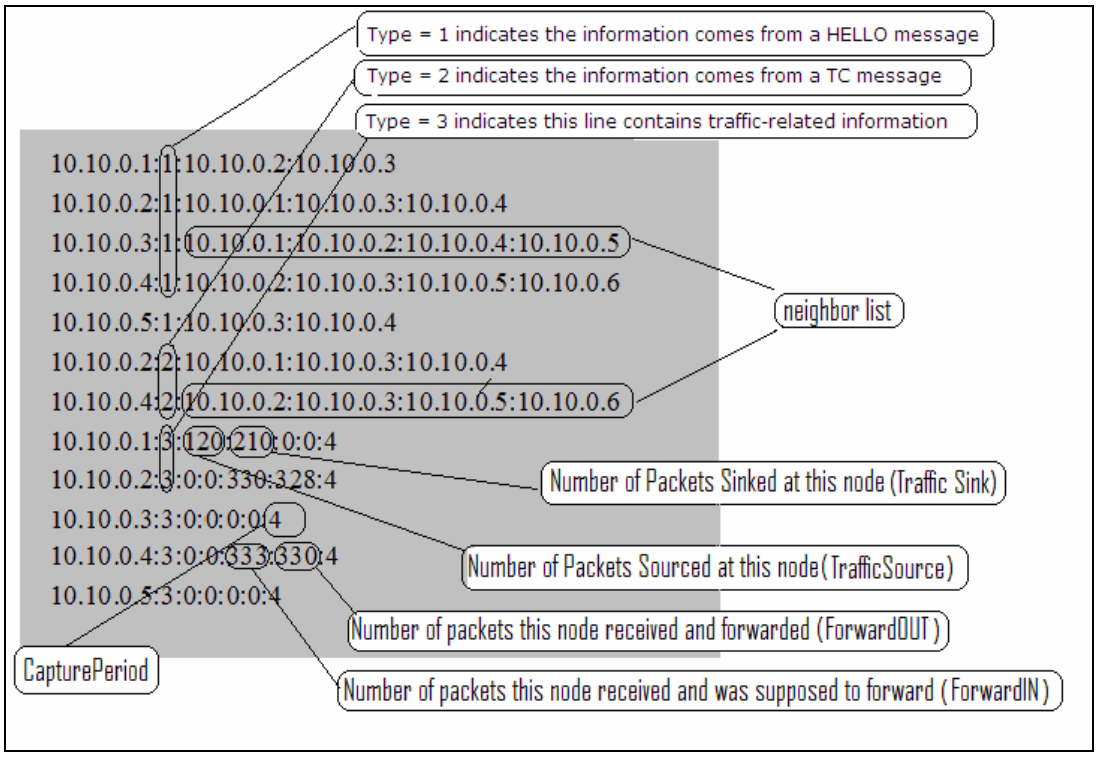

Figure 4-10 : A sample Info File for scenario shown in Figure 4-9.

4.5 Analysis & GUI Component

The Analysis & GUI component processes the Info Files generated by multiple Monitoring Units, merges and consolidate their findings and produces a comprehensive picture of the network topology and information regarding cooperation in forwarding packets and traffic activity of each of the MANET nodes.

The Analysis & GUI component is in fact a Java daemon. It is implemented entirely in Java, taking advantage of its object oriented nature and mature GUI features. In implementing this component, we took advantage of some JPCAP classes, listed in Table 4-1.

4.5.1 Overview of the Analysis & GUI Component

When the Analysis & GUI component launches, it first loads a configuration file which contains the following configuration values:

OLSRDirMonitored: This directory is monitored by the component. The File Delivery module must deliver the Info Files to this directory.

ThreadSleepTime: The sleep time (in ms) for the Analysis & GUI daemon. During this time the daemon sleeps before checking *OLSRDirMonitored* for Info Files again.

OLSRArchiveDir: This is where the info files are archived after being processed by the Analysis & GUI daemon.

The Analysis & GUI component takes in Info Files that are produced by multiple Monitoring Units across the network. It extracts information from individual files and consolidates the reports from all MUs. The component then shows the final results on a dynamic graphical user interface. In all cases, the latest findings are shown on the GUI. A valid time value (*ValidTime*) is associated with each piece of information. If the *ValidTime* of an item is greater than the current time, the item is removed from the GUI.

Figure 4-11 shows the flowchart of the Analysis & GUI component, while Figure 4-12 shows how different classes in this component interact with each other to maintain a

dynamic GUI. Figure 4-13 shows how the *FileProcess* class extracts information from the Info File and ensures that only valid information is shown on the GUI.

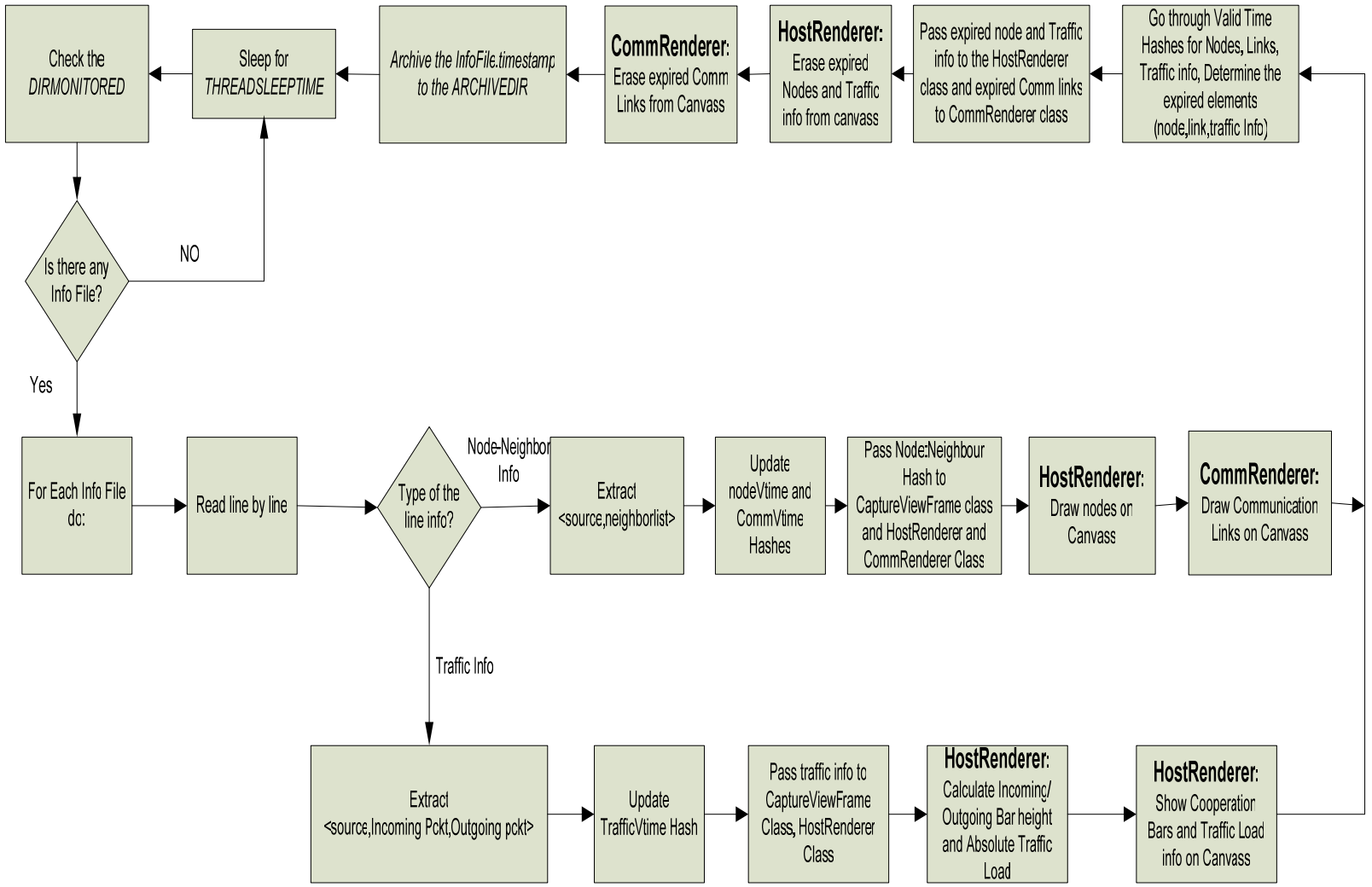

Figure 4-11: The Overall flowchart of the Analysis & GUI component. Java class names are in BOLD letters.

4.5.2 Info File Processing

The Analysis & GUI daemon constantly checks the *OLSRDirMonitored* directory. If there are any Info Files, a list of them is made and each gets processed.

In order to process an Info File, the daemon initiates an instance of the *FileProcess* class. For each Info File, the *FileProcess* object goes through each line, creates multiple hashes for nodes, neighbor lists, expiration time, traffic info, etc. Figure 4-13 shows the detailed flow of the *FileProcess* class.

The *FileProcess* class performs the bulk of the processing and consolidation for the Analysis & GUI component. As shown in Figure 4-12 and Figure 4-13, the *FileProcess* class passes its final findings to the *CaptureViewFrame* class, which in turn calls the *HostRenderer* and *CommRenderer* classes. The *HostRenderer* and *CommRenderer* classes are responsible for displaying or erasing information from the GUI.

Figure 4-12 shows the interaction between the *FileProcess*, *CaptureViewFrame*, *HostRenderer*, and *CommRenderer* classes and how they collaborate with one another to maintain a dynamic GUI.

The *FileProcess* class extracts a *node:neighbourList* hash from the Info Files and passes this hash to the *CaptureViewFrame* class. The *CaptureViewFrame* class then produces *node:neighbor1*, *node:neighbor2, node:neighbor3,* … pairs and passes them to the *CommRenderer* class, where communication links between a node and each of its neighbors are drawn.

The *FileProcess* class also produces a *node:TrafficInfo* hash. This information is passed to the *CaptureViewFrame* class, where it is further analyzed. The *CaptureViewFrame* class then passes two hashes to the *HostRenderer* class: the *node:CooperationInfo* hash and the *node:TrafficLoadInfo* hash.

Then, based on the *node:CooperationInfo* hash, the *HostRenderer* class calculates the following:

$$Cooperation\ Percentage = \frac{Actual\ \ Number\ of\ \ Forwarded\ \ Packets}{Expected\ \ Number\ of\ \ Packets\ to\ be\ Forwarded}$$

To show the cooperation level of a node, the *HostRenderer* class then draws a percentage bar next to the node's icon, representing the node's level of cooperation.

This method of representing the cooperation level helps the network manager detect the selfish nodes with only a glance at the GUI.

To show the traffic activity at each node, the *HostRenderer* class uses the *node:TrafficLoadInfo* hash, which contains the average incoming and outgoing traffic load (in packets/sec) for each node. This information is shown below the node's icon on the GUI.

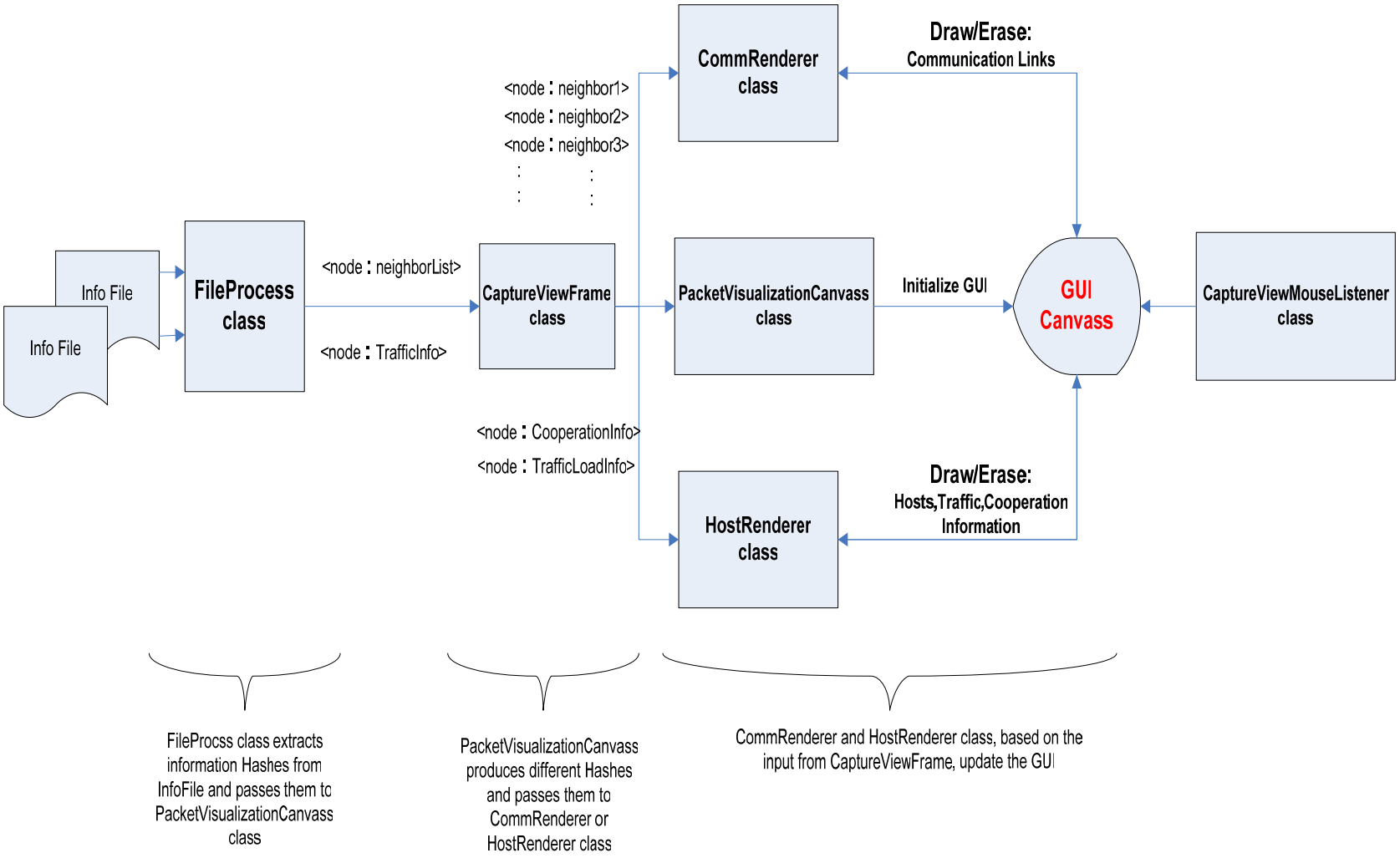

Figure 4-12 : Different Java classes collaborate with each other to initialize and maintain the GUI.

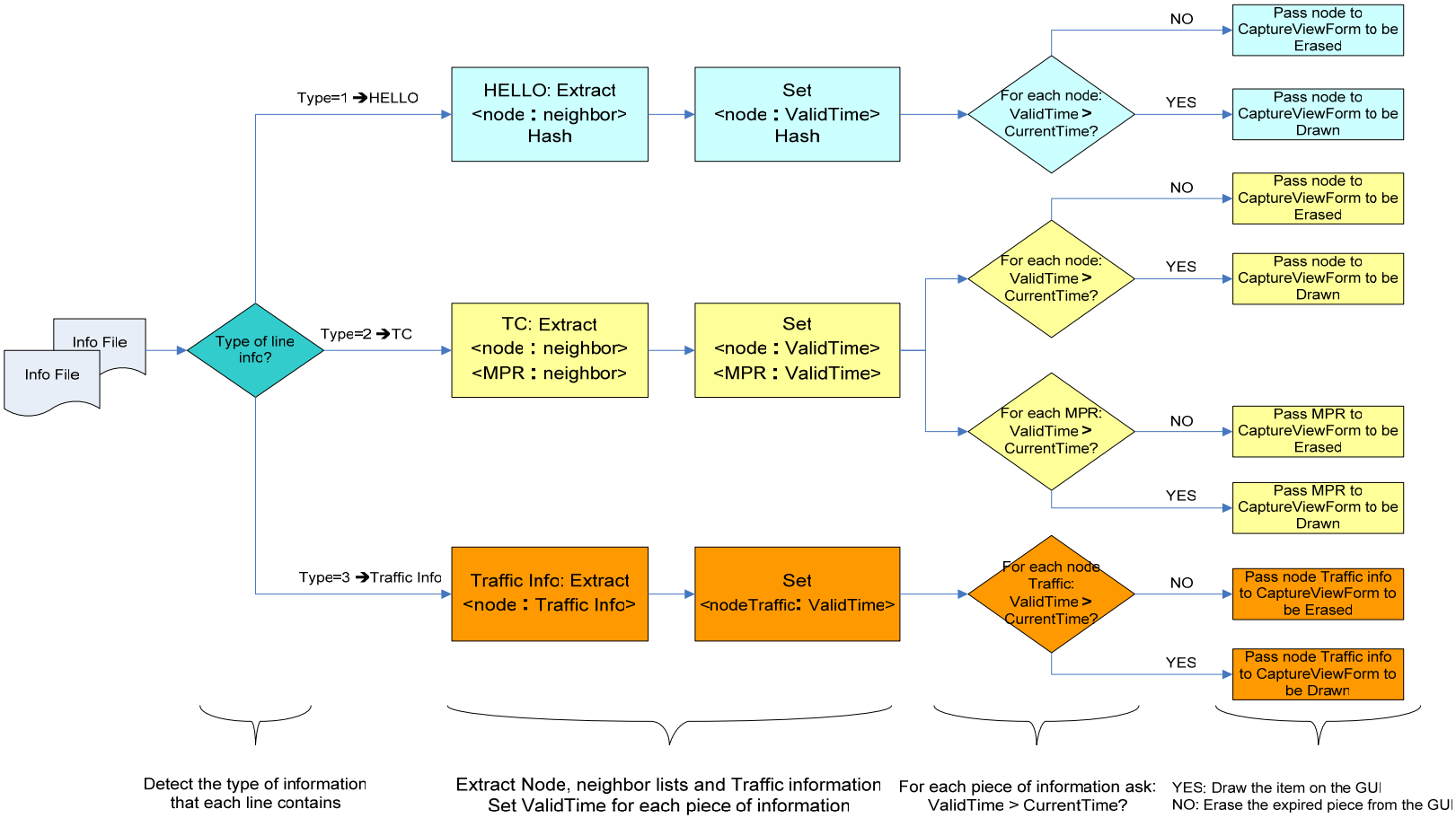

Figure 4-13: Detailed flow of the *FileProcess* class showing how it extracts and validates information.

4.5.3 Graphical User Interface

The graphical user interface is initiated by the *PacketVisualizationCanvass* class. This class is instantiated by the *CaptureViewFrame* class when the Analysis & GUI component is first launched. However, the *HostRenderer* and *CommRenderer* classes maintain the GUI and add or delete information to/from it.

The *CaptureViewMouseListener* class is borrowed from the JPCAP package. An instance of this class helps the user to have mouse action on the GUI and be able to move nodes around the canvass. When a node is moved, all related objects, including its communication links with other nodes and its cooperation and traffic information, are moved with it.

When the *FileProcess* class has determined that a set of information is valid and must be drawn on the GUI, it passes it to the *CaptureViewFrame* class. Based on the type of the information, network-level or node-level, the *CaptureViewFrame* class passes this data to the *CommRenderer* or *HostRenderer* classes, where the information get translated into graphical objects and drawn on the GUI.

Erasing invalid information from the GUI takes the same path. This flow is shown in Figure 4-13.

4.5.3.1 What is shown on the GUI?

The GUI contains two types of information: topology-related information and node traffic-related information.

The following items are shown on the GUI:

1. Each MANET node with its IP address. The MPR nodes are distinguished by a different color.
2. Communication links between nodes.
3. Cooperation information, represented by a graphical bar that shows the percentage of packets that the node has agreed to forward for its neighbors.

4. The average incoming and outgoing load for each node (in packets/sec).

A snapshot of the GUI is shown in Figure 4-14, where the node on the left (10.10.0.22) is the source of the UDP traffic destined to the node on the right (10.10.0.12). In this example, node 10.10.0.1 forwards approximately 90% of the packets that it is asked to forward for its neighbors, while node 10.10.0.11 acts completely selfishly, dropping every packet it is asked to forward.

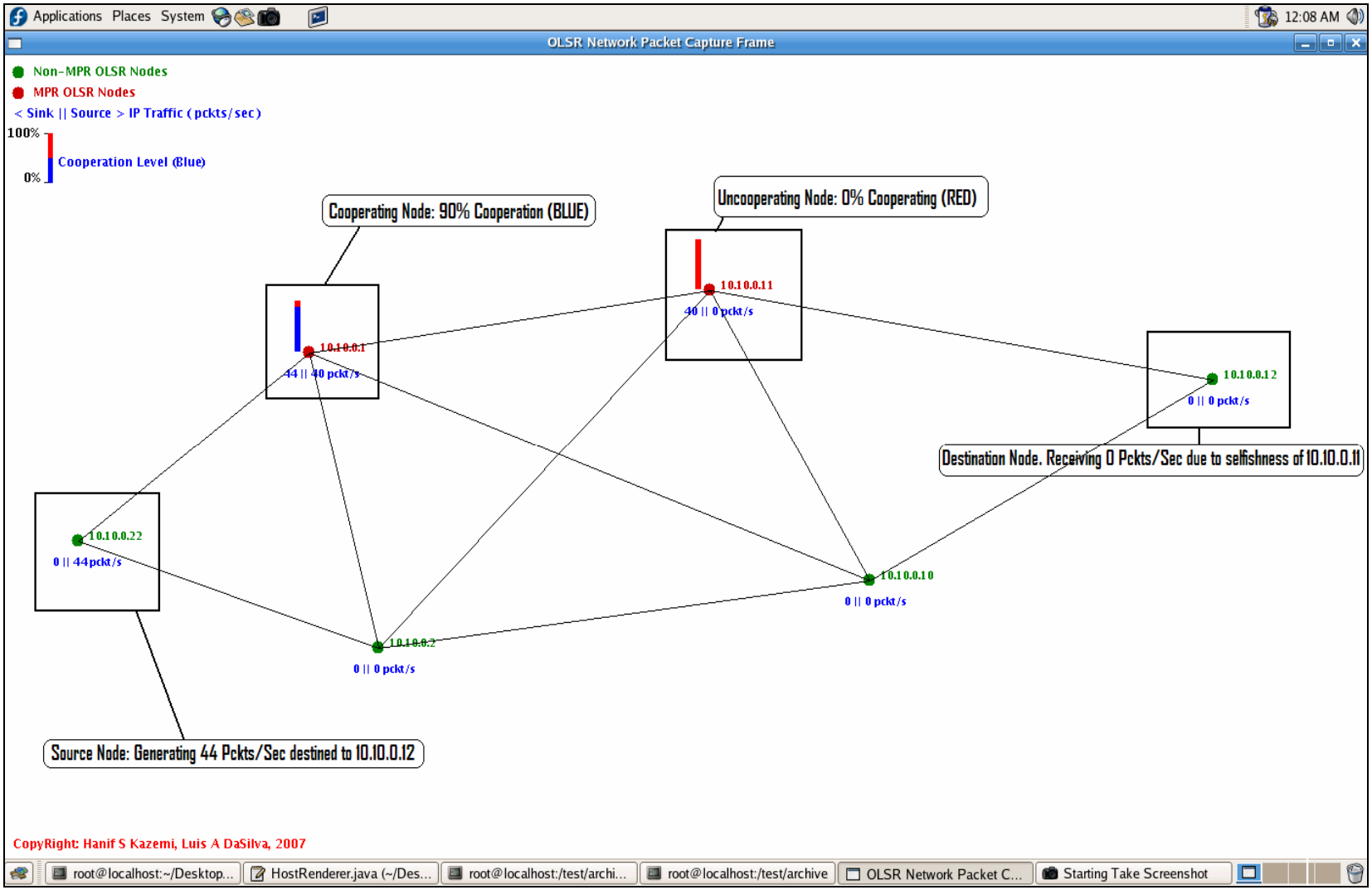

Figure 4-14: A snapshot of the GUI produced by the Analysis & GUI component showing topology, traffic and cooperation information.

4.6 File Delivery Module

The File Delivery module is responsible for delivering the Info Files that are produced by the Capture components, deployed on multiple Monitoring Units across the network, to the Analysis & GUI component. This module uses out-of-band communication for transferring the files.

The File Delivery module is implemented using Korn Shell (KSH) [29] scripting language and uses Secure Shell (SSH) [30] connections for transferring Info Files. In order to guarantee safe and secure delivery of Info Files, all management nodes must be configured for SSH connection. Also, HostBasedAuthentication between these nodes must be enabled and public keys must be exchanged prior to launching the system.

Also, to achieve password-less authentication, hostnames of all management nodes must be added to the /etc/hosts.equiv file on all Monitoring Units.

The File Delivery module should deliver the Info Files that are produced at an MU node to all other management nodes. However, to prevent an infinite cycle of Info Files travelling between MUs and the Analysis & GUI components, the File Delivery module must only push out the Info Files that were produced by its local Capture component - if any - and not transfer Info Files that have been received from other MUs.

To achieve this, we define two directories: a *Shared* directory, where the Capture component saves the Info Files, and a *Current* directory, which is where the Analysis & GUI component expects the Info Files.

In addition to delivering the Info File from an MU to the management node, the File Delivery module is also responsible for feeding the Info Files to the Analysis & GUI component (moving them from the *shared* directory to the *current* directory).

The File Delivery module has the following flow:

1. Get a list of all the files in the *shared* directory;
2. For each file:

If the file is produced by the local Capture component, then SCP it to the *shared* directory of the target host (other MUs and where the Analysis & GUI component is running);

3. Move all the Info Files (locally produced or received from other MUs) to the *Current* directory.

Figure 4-15 shows the overall logic of the File Delivery Module.

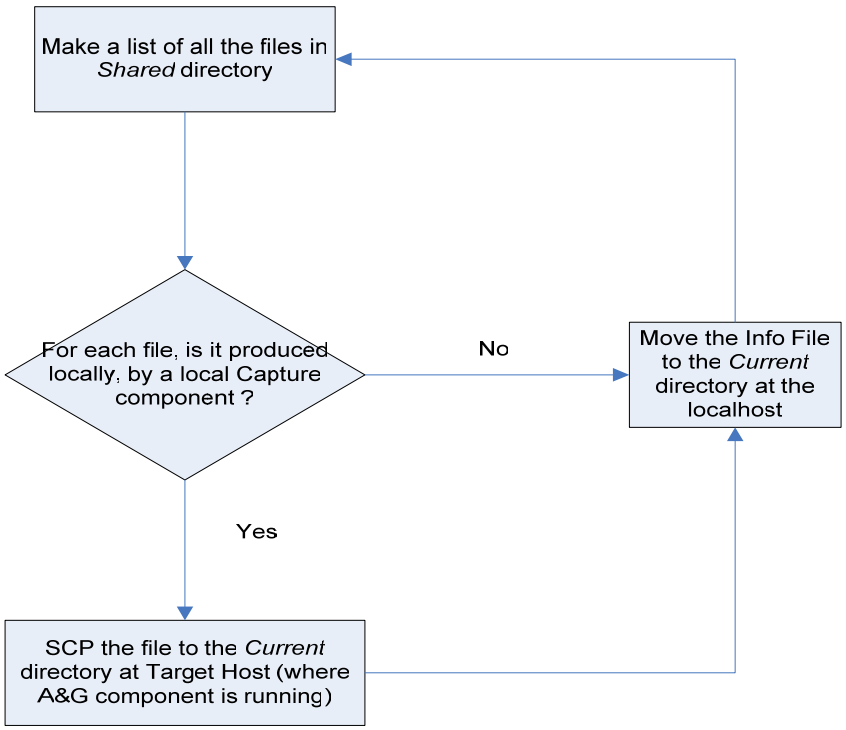

Figure 4-15: The overall logic of the File Delivery module.

4.7 Summary

In this chapter, we detailed the design and implementation of a MANET monitoring system based on our proposed scheme introduced in Chapter 3.

We explained how our proposed solution results in silent collection of network and node-level information. We also detailed how the information gathered by several Monitoring

Units across the network are consolidated and are used to produce a comprehensive picture of network topology, nodes' level of cooperation and traffic load activity.

In the next chapter, we will present the results of experimental performance evaluation of our system.

5 Experimental Evaluation Results

5.1 Introduction

In previous chapters, we presented a framework (Chapter 3) outlining the structure of a lightweight MANET monitoring and performance assessment system. In Chapter 4, we presented the detailed design and implementation of our solution.

In this chapter, we present the results of testing and evaluation of our proposed solution in real ad hoc wireless environments. We examine the robustness and effectiveness of our solution and observe its performance under different circumstances.

The rest of this chapter is structured as follows. In Section 5.2, we describe the methodology of our experimental evaluation. In Section 5.3, we go over the system capabilities and requirements we evaluated in our experiments. Then in Section 5.4, we present the results of several experiments, where our system was deployed to monitor a real ad hoc network. In Section 5.5, we present a summary of our evaluation experiments and system performance.

5.2 Methodology

Results of a MANET monitoring experiment depend heavily on the radio frequency (RF) properties of the test environment. To validate the performance of our system and observe its outcome in a real wireless environment (instead of a testbed or laboratory environment), we conducted our experiments in the following two scenarios:

1. A 10-node MANET deployed across a large residential house, covering an approximate area of 1 acre (~40,000 sq. ft.), with some nodes inside and some outside of the building (Figure 5-1).
2. A 10-node MANET deployed in a typical office building (Figure 5-2).

All of the MANET nodes in both experiments were running OLSR (NRL implementation) as their underlying routing protocol. Also, nodes were running different

versions of the Linux operating systems: one node had *Fedora Core 5*, 4 nodes had *Fedora Core 4* and 5 nodes had *Slackware Linux 10.2* as their operating systems.

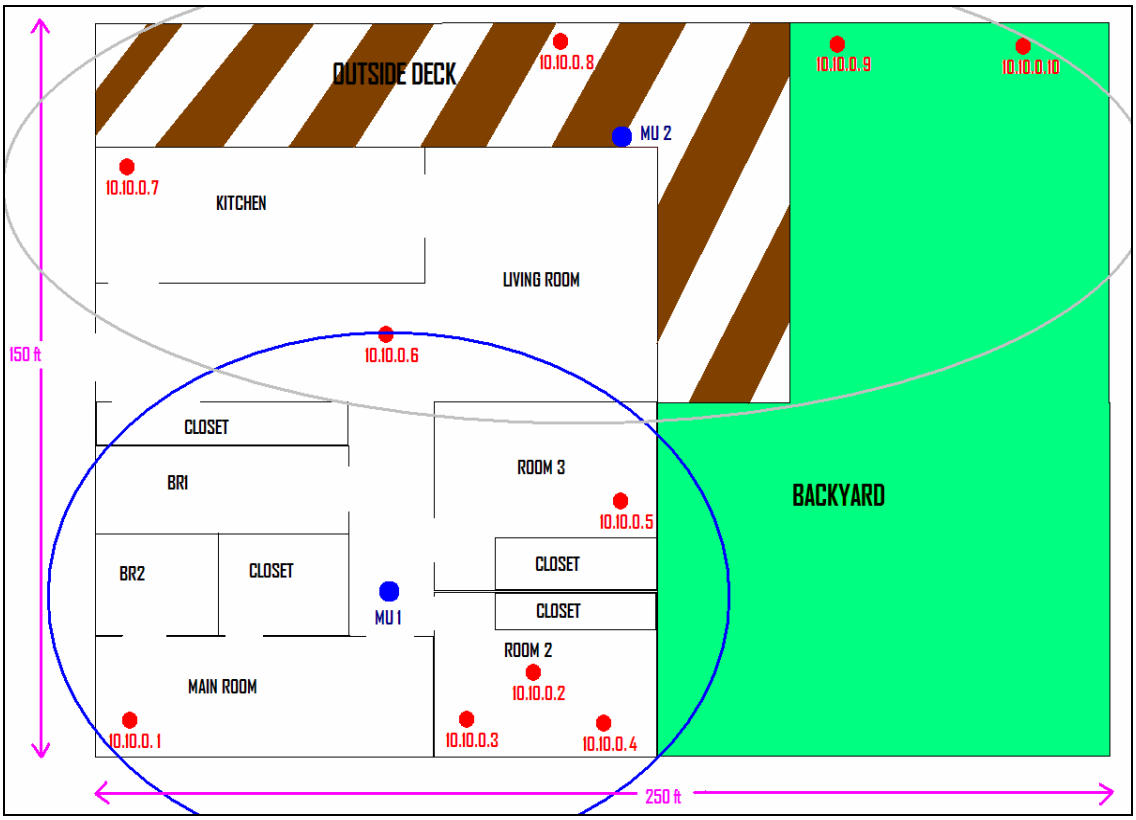

Figure 5-1: A 10-node MANET deployed across a 38,000 sq. ft. house.

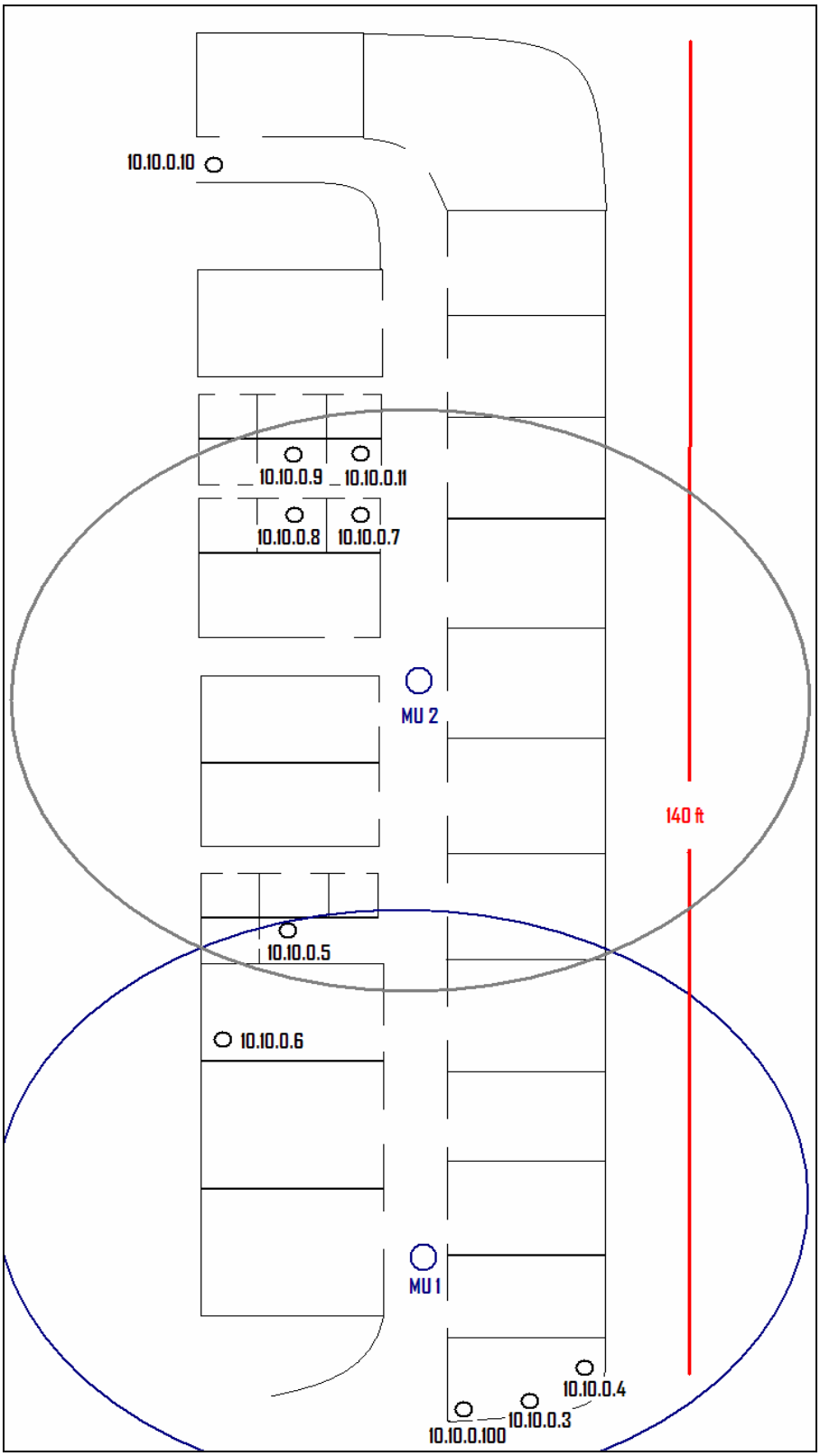

Figure 5-2: A 10-node MANET deployed on a typical office environment.

For both experiments, we evaluated the performance of our system for the following scenarios:

1. The MANET was monitored using only one Monitoring Unit, positioned to have partial coverage of the MANET (please refer to the coverage area of *MU1* in Figure 5-1 and Figure 5-2).
2. The same MANET was monitored by 2 Monitoring Units, where each MU had approximately 50% coverage of the network. Because there was some overlap between monitored areas, the overall coverage of the MANET was about 80%-90%. The coverage areas of *MU1* and *MU2* are shown in Figure 5-1 and Figure 5-2.

5.3 Performance Assessment

In assessing the performance and effectiveness of a MANET monitoring system, there are concerns about the experiment environment. In our experiments, there were regular interference and signal degradation causes such as walls, electrical devices and etc. Experiments were run for 6 periods of 30 minutes each.

The following capabilities and requirements of the system were examined and evaluated:

- Ability to show a dynamic and up-to-date picture of the MANET topology under all circumstances.
- Ability to show traffic details (incoming and outgoing traffic load) for each node under the coverage of the Monitoring Units. This capability was tested for different levels of network traffic load (high and low volume) and different types of traffic flows (TCP/IP and UDP).
- Ability to present an assessment of cooperation level for nodes under the coverage area of the Monitoring Units, with different traffic loads.

- Storage capacity and CPU power requirements for management nodes.

5.4 Experiment Results

The performance of the system was evaluated from different perspectives. In this section, we present the findings of our experiments in different groupings.

5.4.1 Producing an Up-to-date MANET Topology Picture

The overall performance of our system in producing the MANET topology picture has been detailed in the following categories:

A. Incomplete vs. Complete Coverage

To better understand the results of partial versus complete coverage of the MANET by the Monitoring Units, we examined the performance of the system in the network shown in Figure 5-1. Figure 5-3 shows a screenshot of the topology produced by the Analysis & GUI component when one Monitoring Unit (MU 1) is covering half of the MANET nodes. In Figure 5-4 we see that the addition of another well-positioned Monitoring Unit (MU 2) helps the system to produce a complete network topology picture.

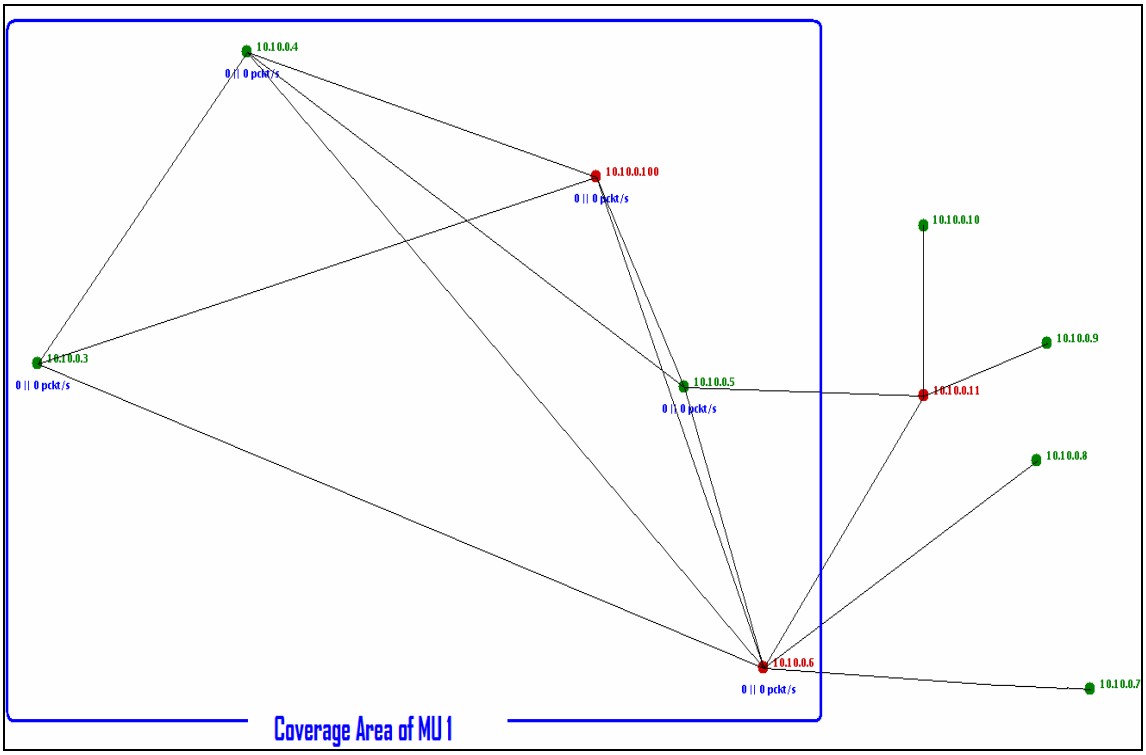

Figure 5-3: More details are provided for areas covered by the Monitoring Unit(s)

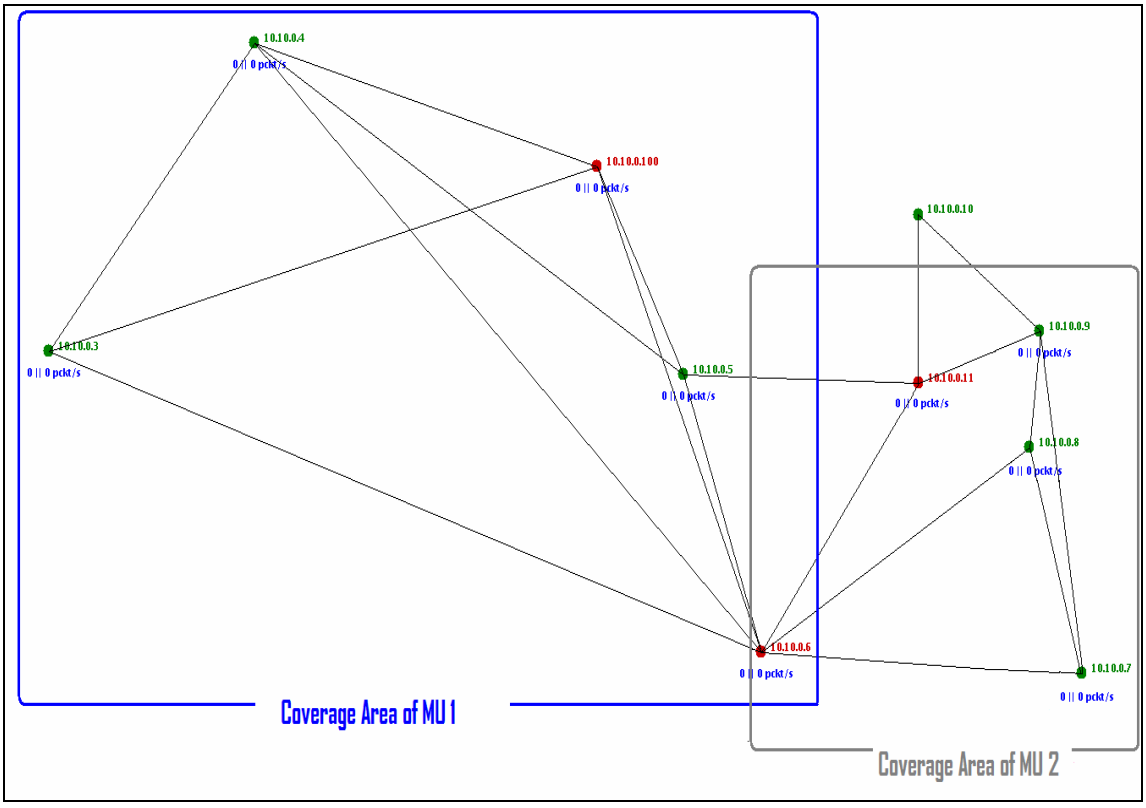

Figure 5-4: Addition of another MU (total coverage of ~90%) helps the system to produce a complete picture of the network topology.

B. Ability to Detect Network Partitions

The distributed nature of our system, in addition to its independence from the monitored MANET (both were critical design objectives), help the system to perform successfully in the face of network partitions, a common feature of MANET that has rarely been addressed in other monitoring solutions.

Figure 5-5 shows the result of the monitoring when the MANET is partitioned due to the loss of a critical communication link.

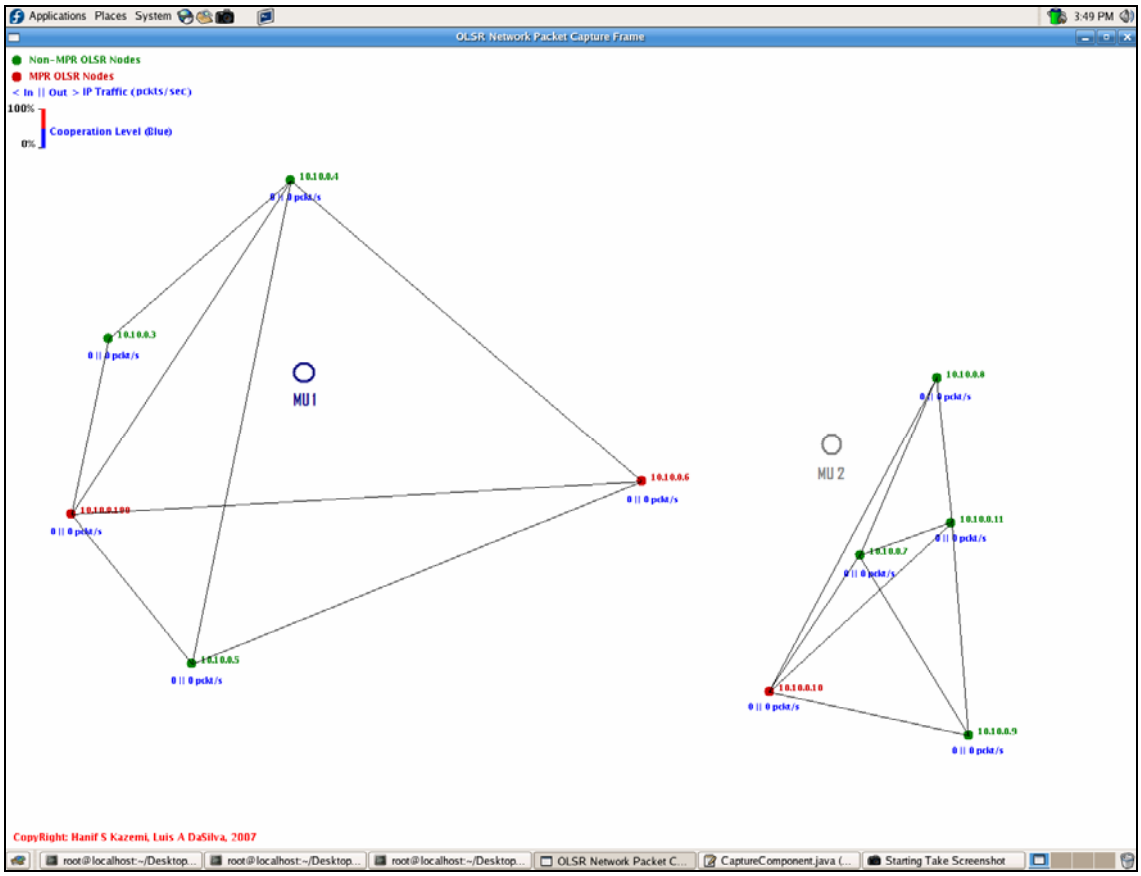

Figure 5-5: Well positioned MUs help the system to report MANET partitioning.

C. Accuracy and Dynamicity

Throughout our experiments, we observed that even in an unstable MANET, where links change rapidly, our system was always able to produce an up-to-date report on the MANET's status. Topology picture and traffic information were dynamic and constantly updated. Also when nodes would fall out of reach of MUs, their node-level information would vanish from the GUI, resulting in a dynamic and accurate picture of the network. We witnessed that every change in the network was reflected on the GUI in about 3-3.5 seconds.

D. Details Regarding an Individual Node's Role in the MANET

The system was able to detect each node's role in the MANET at all times. In all screenshots, nodes with darker color (dark red) are OLSR multipoint relays (MPR) nodes and nodes shown in green are non-MPR nodes. It should be noted that even the nodes that are outside of the coverage area of MUs were marked based on their role; node 10.10.0.11 in Figure 5-3 is out of the coverage area, yet it is marked as an MPR. Figure 5-4 confirms the node's role.

5.4.2 Assessing Traffic Level and Cooperation Grade

Cooperation degree and traffic-related information is provided for nodes that are covered by an MU. Because this information is produced by counting the packets that are transmitted to or from each node, it was important for us to observe the performance of the system in both low-volume and high-volume traffic environments, and when there was complete and partial coverage of the MANET.

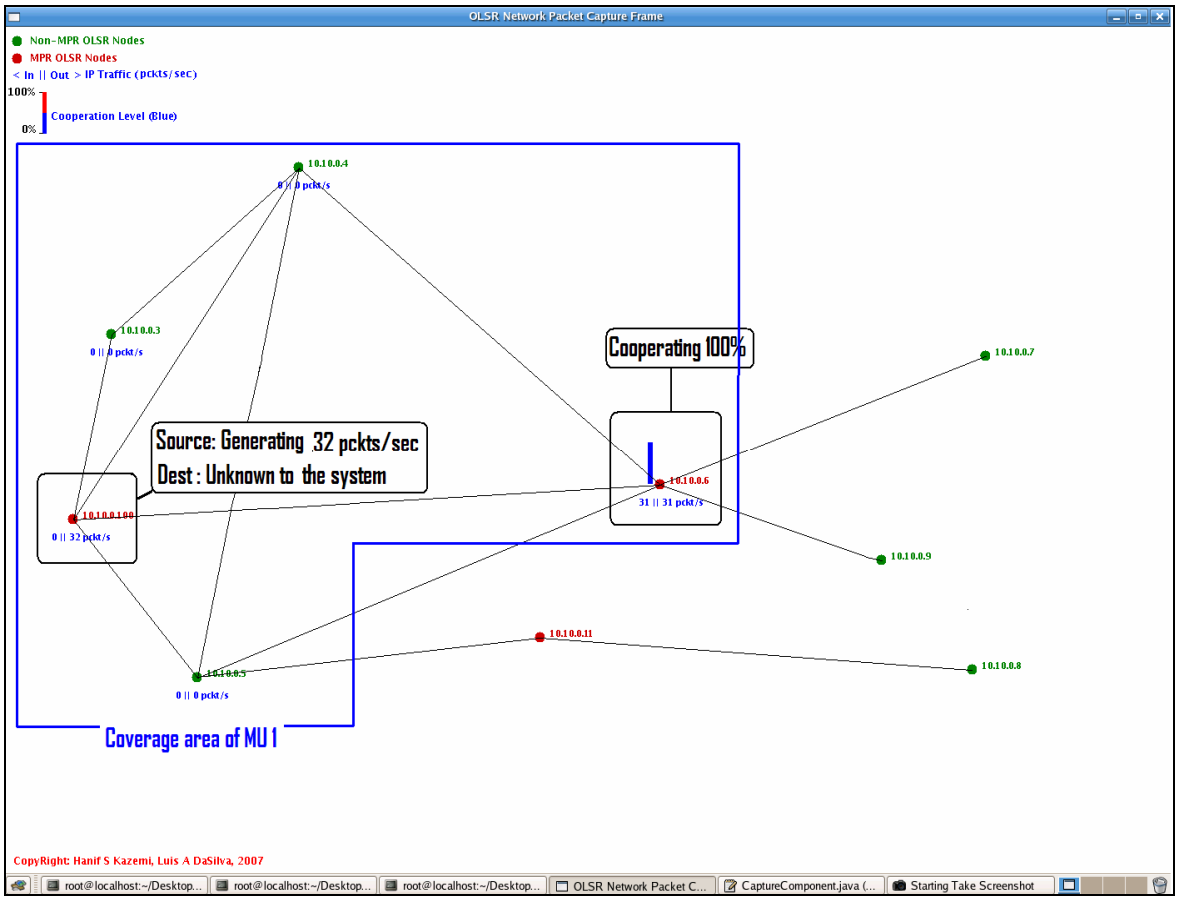

Figure 5-6: Low UDP traffic load (~ 0.25 Mbps) and partial coverage of the MANET by one MU.

Figure 5-6 and Figure 5-8 show that in both cases of high-volume and low-volume traffic, when MUs have a partial coverage over the monitored MANET, traffic and cooperation information is still accurate and precise.

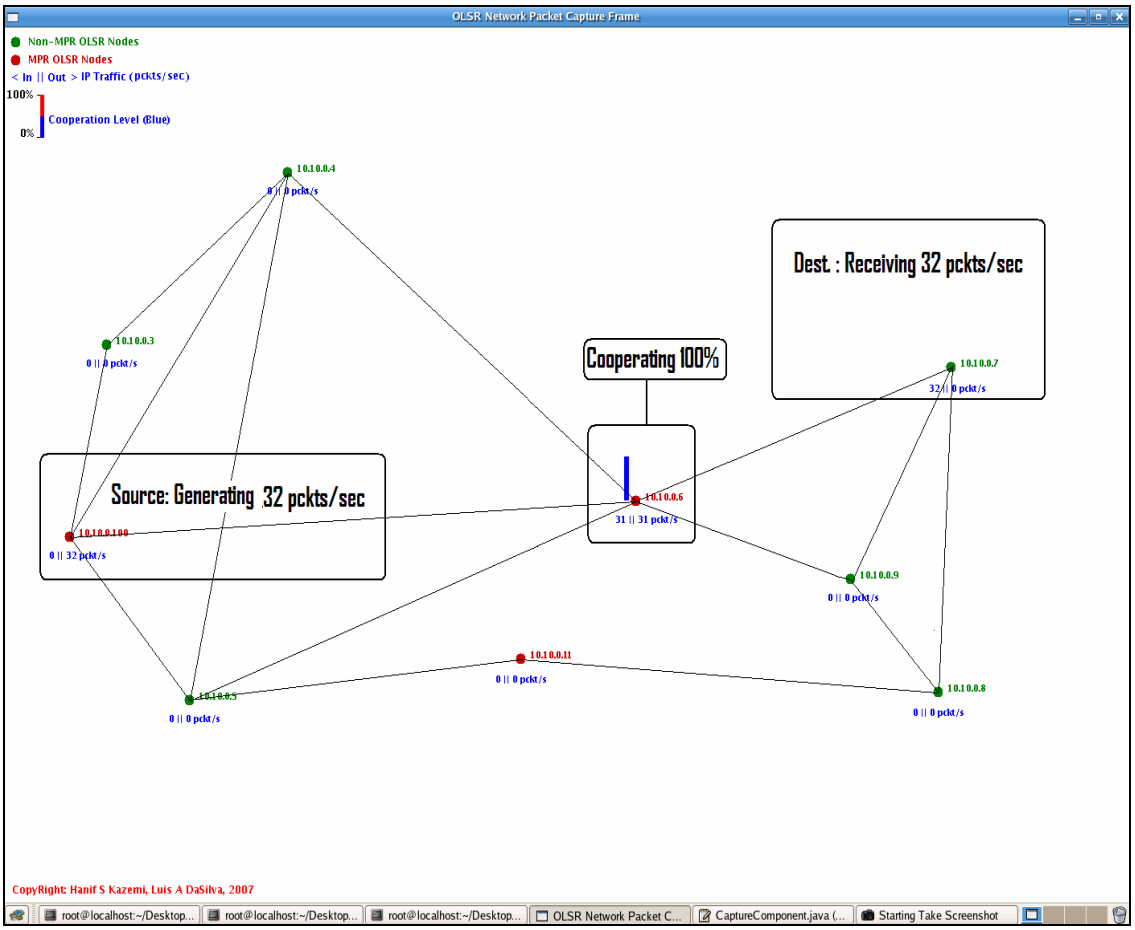

Figure 5-7: Low UDP traffic load (~ 0.25 Mbps) and complete coverage of the MANET by two MUs.

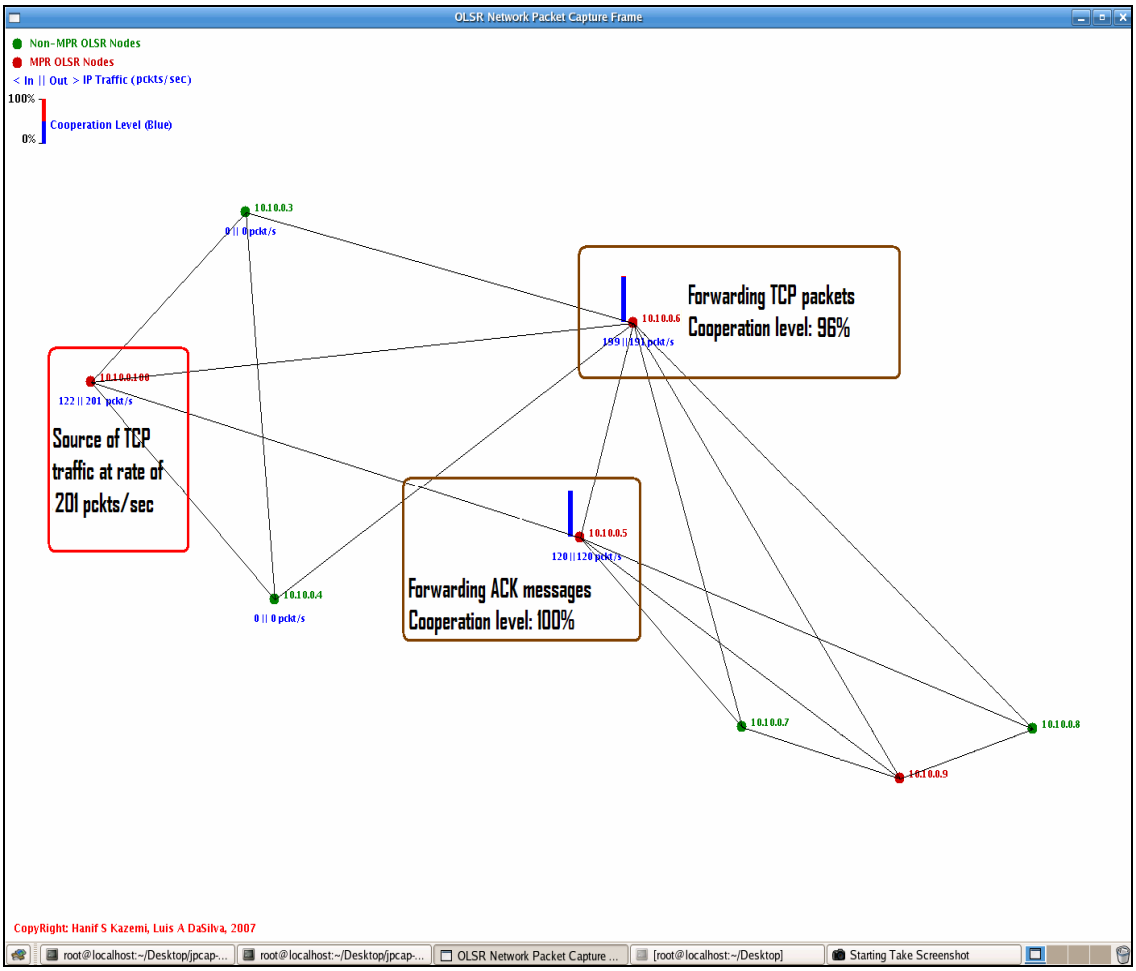

Figure 5-8: High traffic TCP flow (~ 2.7 Mbps) and partial coverage of the MANET.

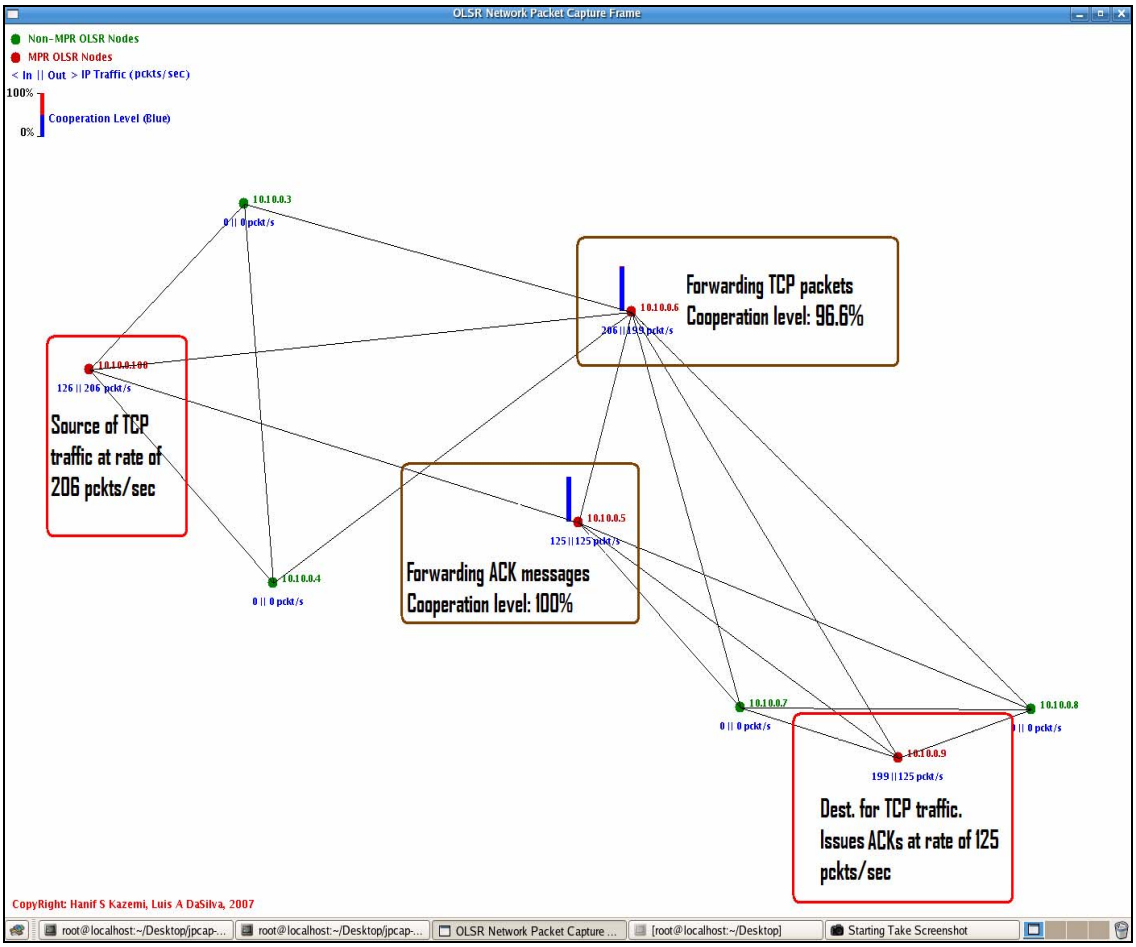

Figure 5-9: High traffic TCP flow (~ 2.7 Mbps) and complete coverage of the MANET.

5.4.3 Studying the Effect of Selfishness on Traffic Flow

When a UDP source generates traffic destined to a particular node, it does not care whether the packets are received by the destination or not. Therefore a node that drops only some of the packets has little negative effect on a UDP flow. However, lack of complete cooperation by a node can significantly affect a TCP flow. Because before sending new packets the TCP source node waits for acknowledgement of sent packets, not receiving ACK messages causes a TCP timeout, a reduction of the sender window size, and long delay before resending the packets. Figure 5-10 shows how a partially-cooperative node who refuses to forward ACK messages can significantly affect a TCP flow.

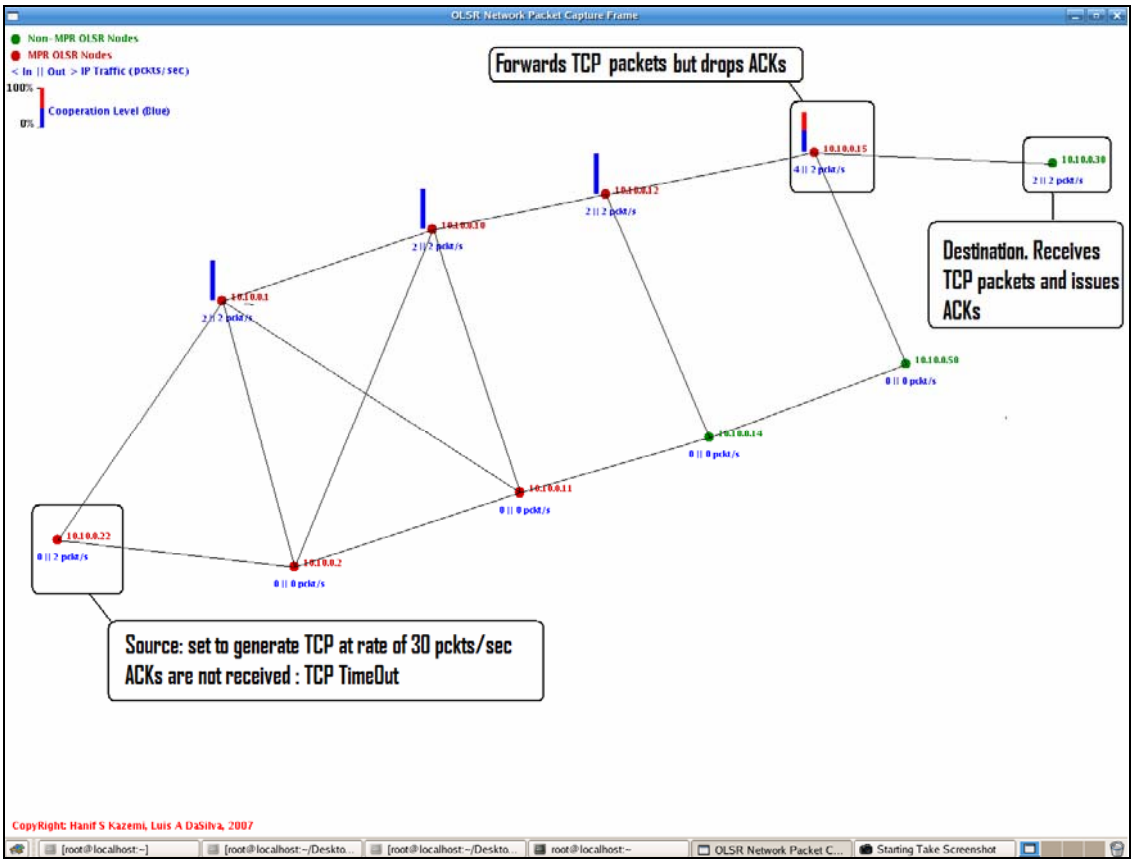

Figure 5-10: A non-cooperative node can significantly affect TCP traffic flows.

5.4.4 System Storage Capacity and CPU Power Requirements

To get an estimate of the CPU requirements on Monitoring Units and management nodes, we took advantage of the *top* utility. We observed that, on average, a Capture component consumes about 3-4% CPU power of a regular laptop (Fedora Core 5, Pentium M 1.6 GHz) and the Analysis & GUI component needs about 3% CPU power of the same computer. When both components were deployed on this laptop, CPU dedication to system never reached 8%, a modest processing requirement.

The Analysis & GUI component is designed to archive all the Info Files that it receives into an *archive* directory. We measured the number and the size of Info Files that are produced by all MUs in our experiments and the total size of Info Files that were transmitted to the Analysis & GUI component, hence helping us estimate the bandwidth

requirement for the out-of-band communication between the MUs and the manager node (used by the Delivery module).

Table 5-1 shows a summary of storage capacity and out-of-band communication link bandwidth requirements for our system.

Parameter	Value
Number of Info Files generated by local Capture component	800 Info Files in 40 minutes
Number of Info Files generated by a remote Capture component	600 Info Files in 30 minutes
Average size of Info Files	630 bytes
Info File size range	330-830 bytes
Bandwidth requirement for transmission of Info Files from the remote Capture component to the Analysis&GUI component	~ 1.6 Kbps
Total storage capacity required on the Analysis&GUI component	< 1 MB

Table 5-1: Storage capacity and BW requirements for a 30 minute experiment

5.5 Summary

In this chapter, we presented the experimental performance evaluation of our MANET monitoring system implementation studied under varied networking conditions – with different network densities, complete and partial coverage of the MANET, altered node cooperation levels and various traffic rates- in a real MANET environment.

In general, our monitoring system was capable of producing the expected results under all circumstances.

We verified that the produced topology picture of the network was always accurate and up-to-date. Also, traffic load information and cooperation grades of monitored nodes were precise and a true reflection of the monitored MANET's behavior.

We illustrated the effectiveness of our solution in detecting selfishness, showing network partitions, and its robust performance under high-volume traffic.

Finally, we presented a summary of the system's modest CPU power and storage capacity requirements.

6 Conclusion and Future Work

6.1 Contributions

In this thesis, we formulated the overall design structure of a monitoring solution that is generally applicable to mobile decentralized networks. Our framework presented a distributed and lightweight scheme for observing MANETs and producing network-level and node-level reports on a user-friendly graphical user interface.

The unobtrusive and distributed nature of our approach to performance assessment of a MANET resulted in an effective and efficient solution where monitoring nodes are not a part of the network and collect information promiscuously. Then they summarize their findings and transmit them to a management unit, where findings of multiple monitoring nodes are consolidated to produce a final report. Since the communication between all management nodes (a monitoring unit and a management node) takes place in an out-of-band communication link, our monitoring solution does not depend on the MANET's performance, hence is robust to network partitioning, link breaks, a node's death and node misbehavior.

We then presented a detailed design and implementation of our framework for a MANET with the Optimized Link State Routing (OLSR) protocol as its underlying routing protocol. The structural design of our system is general to all MANETs, and although our implementation assumes the use of OLSR, the system can easily be modified to work with other MANET routing protocols.

Next, we exhibited the results of experimental evaluation of our system. We deployed our system to monitor two 10-node MANETs; one MANET across a 40,000 sq. ft. house and one MANET spread in a typical office building.

In both experiments we observed that our system performed as expected, with a superior resiliency to topology changes, link breaks and misbehavior. The results were verified in

both low and high volume traffic networks. Furthermore, we witnessed that the system had modest storage capacity requirements (< 1 Mb for 30 minutes of monitoring), CPU consumption on management nodes was low (< 8%) and bandwidth requirements on the out-of-band communication link were not significant (< 1.7 kbps).

6.2 Comments

During our experimental evaluation, we verified that our approach in structuring, and our care in implementing the system, had paid up. We observed that the distributed nature of the system helped it to be robust in the face of network partitioning or incomplete coverage of the MANET, a clear advantage over most other MANET monitoring solutions. Also, we confirmed the efficiency and effectiveness of the system in presenting an up-to-date topology picture of the network with details regarding each node's role, level of traffic load and forwarding cooperation percentage. Other advantages of our system over other proposed schemes are: it is unobtrusive to the monitored MANET, does not require new software to be installed on the MANET nodes, and causes no deterioration on the performance of the MANET.

6.3 Future Work

Management of mobile ad hoc networks is still an active field of research. With expansion of MANET applications in commercial and defense worlds, everyday advances in the use of directional antennas, and development of new MANET routing protocols, it is imperative for researchers to come up with management solutions that are not only capable of detecting flaws and failures, but are also able to analyze the problems and take the appropriate actions. However, because of efficiency and stability concerns associated with mobile ad hoc networks, any management scheme should avoid affecting the performance of the MANET.

For future work, we believe that it is important to implement our proposed approach with other MANET routing protocols and observe its performance. Also, integrating this monitoring solution with a MANET management system, where management decisions

are made based on the information collected by the monitoring system, is another important step toward accomplishing a comprehensive MANET management solution.

References

1. B. Biskupski, J. Dowling and S. Jan, "Properties and Mechanisms of Self-organizing MANET and P2P Systems," *ACM Trans. Auton. Adapt. Syst. Journal*, Jan. 2007. p. 1.

2. I. Almomani, et al., "Architectural Framework for Wireless Mobile Ad hoc Networks (AF WMANETs)," *Computer Communications*, Aug. 2006. pp. 178-191.

3. IETF MANET (Mobile Ad-hoc Networks) Working Group. Nov. 2006; http://www.ietf.org/html.charters/manetcharter.html.

4. N. Negroponte, "One Laptop Per Child: Non-Profit Movement for Spreading Education Across the World," Aug. 2006; http://www.laptop.org/.

5. H. Slay, P. Wentworth and J. Locke, "BingBee, An Information Kiosk for Social Enablement in Marginalized Communities," *Proc. of 2006 S. African Inst. of Computer Scientists in Developing Countries Conference*, Mar. 2006. pp. 107-116.

6. Ozone Communications Inc., Wireless Internet Provider, Paris, France, Mar. 2006; http://www.ozone.net/en/reve.html.

7. P. Sass, "Communications Networks for the Force XXI Digitized Battlefield," *Journal of Mobile Network Applications*, Nov. 1999. pp. 139-155.

8. E. Huang and J. Crowcroft, "Towards Commercial Mobile Ad hoc Network Applications: A Radio Dispatch System," *Proc. of the 6th ACM International Symposium on Mobile Ad hoc Networking and Computing*, Urbana-Champaign, IL, Sep. 2005. pp. 355-365.

9. V. Namboodiri, M. Agarwal and L. Gao, "A Study on the Feasibility of Mobile Gateways for Vehicular Ad-hoc Networks," *Proc. of the 1st ACM International Workshop on Vehicular Ad hoc Networks*, Philadelphia, PA, Jun. 2004. pp. 66-75.

10. L. Qun and R. Peterson, "Distributed Algorithms for Guiding Navigation Across a Sensor Network," *Proc. of the 9th Annual International Conference on Mobile Computing and Networking*, San Diego, CA, Dec. 2003. pp. 313-325.

11. F. Damien, et al., "Towards Ad hoc Contextual Services for Pervasive Computing," *Proc. of the 1st Workshop on Middleware for Service Oriented Computing* (MW4SOC 2006), Melbourne, Australia, Jul. 2006. pp. 36-41.

12. R. Badonnel, R. State and O. Festor, "Management of Mobile Ad hoc Networks: Information Model and Probe-based Architecture," *International Journal of Network Management*, Oct. 2005. pp. 335-347.

13. S. Chien-Chung, et al., "The GUERILLA Management Architecture for Ad hoc Networks," *Proc. Military Communications Conference* (MILCOM), Anaheim, CA, Oct. 2002. pp. 467-472.

14. C. Wenli, J. Nitin and S. Singh, "ANMP: Ad hoc Network Management Protocol," *IEEE Journal on Selected Areas in Communications*, Dec. 1999. pp. 1506-1531.

15. E. M. Belding-Royer, K. N. Ramachandran and K. C. Aimeroth, "DAMON: A Distributed Architecture for Monitoring Multi-hop Mobile Networks," *Proc. Sensor and Ad Hoc Communications and Networks* (IEEE SECON), Santa Clara, CA, Oct. 2004. pp. 601-609.

16. S. Buchegger and J.Y Boudec, "Performance Analysis of the CONFIDANT Protocol," *Proc. of the 3rd ACM International Symposium on Mobile Ad hoc Networking & Computing*, Lausanne, Switzerland, Apr. 2002. pp. 226-236.

17. P. Michiardi and R. Molva, "CORE: COllaborative REputation Mechanism to Enforce Node Cooperation in Mobile Ad hoc Networks," *Proc. of 6th Joint Working Conf. on Comm. and Multimedia Sec.* (IFIP TC6/TC11), Apr. 2002. pp. 107-121.

18. K. Balakrishnan, D. Jing and V. K. Varshney, "TWOACK: Preventing Selfishness in Mobile Ad hoc Networks," *Proc. Wireless Communications and Networking Conference* (IEEE WCNC), Mar. 2005. pp. 2137-2142.

19. M. Baker, K. Giuli and S. Marti, "Mitigating Routing Misbehavior in Mobile Ad hoc Networks," *Proc. 6th Annual International Conference on Mobile Computing and Networking* (MOBICOM), Boston, MA, Aug. 2000. pp. 255-265.

20. J. Chen, S. Zhong and Y. R. Yang, “SPRITE: A Simple, Cheat-proof, Creditbased System for Mobile Ad-hoc Networks,” *Proc. 22nd Annual Joint Conference of the IEEE Computer and Communications Societies* (INFOCOM 2003), San Francisco, CA, Mar. 2003. pp. 1987-1997.

21. JPCAP: A Java API to Libpcap Packet Capture Utility. Nov. 2006; *http://jpcap.sourceforge.net/*.

22. Naval Research Lab (NRL) implementation of Optimized Link State Routing protocol. Oct. 2006; *http://pf.itd.nrl.navy.mil/projects.php?name=olsr*.

23. Libpcap: A Sourceforge Project for Packet Capture Tool. Nov. 2006; *http://sourceforge.net/projects/libpcap/*.

24. P. Jacquet, et al., "Optimized Link State Routing Protocol (OLSR) for Ad hoc Networks," *Proc. Technology for the 21st Century* (IEEE INMIC), Lahore, Pakistan, Feb. 2001. pp. 62-68.

25. P. Drake, "Using SNMP to Manage Networks," *Proc. Designing Resilient Architectures* (IEE Colloquium), Sep. 1991. pp. 21-24.

26. T.S. Randhawa and R.H. Hardy, "Over-the-air Management of Multi-mode Mobile Hosts Using SNMP," *Proc. IP Operations and Management*, Jun. 2002. pp. 189-193.

27. T. Nanya and E. P. Duarte, "An SNMP-based Implementation of the Adaptive Distributed System-level Diagnosis Algorithm for LAN Fault Management," *Proc. Network Operations and Management Symposium* (IEEE CS Press), Apr. 1999. pp. 530-539.

28. J. Kantorovitch and P. Mahonen, "Case Studies and Experiments of SNMP in Wireless Networks," *Proc. IP Operations and Management*, Jun. 2002. pp. 179-183.

29. D. Korn, J. Korn and C. Northrup, "The New Korn Shell," *Linux Journal*, Dec. 1996. pp. 4.

30. D. V. Bhatt, J.F. Blignaut and G.P. Hancke, "Securing a Transmission Channel Between Two Remote Computers with Secure Shell," *Proc. 7th AFRICON Conference*, Apr. 2004. pp. 377-382.

Vita

Hanif Kazemi was born on Aug. 1st, 1980 in the city of Mashad, Iran. He earned his bachelor degree, cum laude, in Electrical Engineering from the Virginia Polytechnic Institute and State University (Virginia Tech) in December 2005. Upon his graduation, he joined S.W.I.F.T as a fulltime system engineer and network developer while he continued his studies in Bradley Department of Electrical and Computer Engineering at Virginia Tech, Arlington Research Institute. In the course of the next 16 months, he continued to work fulltime, take classes toward his degree, and conduct his research with the help of his mentor, professor and advisor; Dr. Luiz A DaSilva.

Hanif hopes to continue his work and research in the area of network management, fault detection systems and next generation communication systems.