ABSTRACT

Title of dissertation:	IMPROVING NETWORK PERFORMANCE, SECURITY AND ROBUSTNESS IN HYBRID WIRELESS NETWORKS USING A SATELLITE OVERLAY Ayan Roy-Chowdhury Doctor of Philosophy, 2008
Dissertation directed by:	Professor John S. Baras Department of Electrical and Computer Engineering

In this thesis we propose that the addition of a satellite overlay to large or dense wireless networks will result in improvement in application performance and network reliability, and also enable efficient security solutions that are well-suited for wireless nodes with limited resources. We term the combined network as a hybrid wireless network. Through analysis, network modeling and simulation, we quantify the improvement in end-to-end performance in such networks, compared to flat wireless networks.

We also propose a new analytical method for modeling and estimating the performance of hybrid wireless networks. We create a loss network model for hybrid networks using the hierarchical reduced loss network model, adapted for packet-switched networks. Applying a fixed point approximation method on the set of relations modeling the hierarchical loss network, we derive a solution that converges to a fixed point for the parameter set. We analyze the sensitivity of the performance metric to variations in the network parameters by applying Automatic Differentiation to the performance model. We thus develop a method for parameter optimization and sensitivity analysis of protocols for designing hybrid networks.

We investigate how the satellite overlay can help to implement better solutions for secure group communications in hybrid wireless networks. We propose a source authentication protocol for multicast communications that makes intelligent use of the satellite overlay, by modifying and extending TESLA certificates. We also propose a probabilistic non-repudiation technique that uses the satellite as a proxy node. We describe how the authentication protocol can be integrated with a topology-aware hierarchical multicast routing protocol to design a secure multicast routing protocol that is robust to active attacks.

Lastly, we examine how the end-to-end delay is adversely affected when IP Security protocol (IPSEC) and Secure Socket Layer protocol (SSL) are applied to unicast communications in hybrid networks. For network-layer security with low delay, we propose the use of the Layered IPSEC protocol, with a modified Internet Key Exchange protocol. For secure web browsing with low delay, we propose the Dual-mode SSL protocol. We present simulation results to quantify the performance improvement with our proposed protocols, compared to the traditional solutions.

IMPROVING NETWORK PERFORMANCE, SECURITY AND ROBUSTNESS IN HYBRID WIRELESS NETWORKS USING A SATELLITE OVERLAY

by

Ayan Roy-Chowdhury

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2008

Dedication

To my family -

Richa, Rishi, Baba, Ma,

Chordibhai, Joyjitda and

Eric.

Acknowledgments

I would like to thank my advisor, Professor John S. Baras for his support and encouragement over the years and for giving me the opportunity to work on several challenging projects and problems. Working with him has been a privilege and it has greatly enhanced my knowledge and technical abilities.

I would also like to thank Professor Alexander Barg, Professor Manoj Franklin, Professor Raymond Miller and Professor Gang Qu for kindly agreeing to serve on the dissertation committee and for reviewing my dissertation.

My graduate work has been supported by the National Aeronautics and Space Administration under award No.NCC8235 and I am grateful for that support.

I would like to acknowledge the invaluable help and support that I have received from the staff at ISR and the ECE Department. Kimberley Edwards, Diane Hicks and Althia Kirlew have over the years taken care of all non-research-related, but nonetheless critical, aspects of graduate studies. Peggy Jayant and Carlos Luceno have been very helpful with my numerous requests for technical support. Maria Hoo has smoothed all interactions with Graduate School. I am very grateful to them.

Special thanks are due to my colleagues and friends in ISR with whom I have worked on projects and discussed various technical ideas that have contributed to this dissertation. I would particularly like to mention Dr. Michael Hadjitheodosiou, Dr. Majid Raissi-Dehkordi, Nicolas Rentz, Dr. Gun Akkor, Dr. Vahid Tabatabaee, Karthikeyan Chandrashekhar, Hui Zengh and Georgios Papageorgiou.

The successful completion of my Ph.D. dissertation is the culmination of a long journey which started when I was a young kid curious about the world, interested in knowing more and hoping to grow up a learned person. I have been encouraged and supported in this journey by many in my family. Some are no longer around to share in my success - *Sejodadu* M.N. Roy Choudhury, *Jethu* Samir Roy Choudhury, *Puta* Dipika Roy Choudhury, *Pishimoni* Monika Basu, *Sundarpisho* Amitabha Sarkar and *Jaharkaku* Capt. J.L. Roy, but I am remembering them today with love and gratitude. Thankfully there are many who are around today to know how grateful I am to them for everything they have done for me. Foremost are my parents Probir and Sabita Roy Choudhury and my sister Preetha and brother-in-law Joyjit Nath, who have been the pillars of support in my life. Their love and encouragement sustain me in my endeavors. My thanks also to my aunts and uncles - especially *Mejomesho* Samudra Prasad Sinha and *Chhotomashi* Sumi Dutta.

I am also thankful to my wonderfully supportive friends who have helped me navigate the difficult stretches in my graduate student life. They include Dr. Brinda Ganesh, Dr. Ayush Gupta, Nipamanjari Dutta, Raj Sarkar, Dr. Madhubanti Mukherjee and Anirvan and Deboja Das. Joyce Lehrer and Jay Wolvovsky also deserve my heartfelt thanks for considering me a part of their family.

Finally, my undying gratitude and love to Eric Wolvovsky for everything he has done and continues to do for me and for loving me and believing in me and helping me cross the finish line.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Wireless ad hoc networks hold the promise of ubiquitous connectivity, but their widespread deployment has not become a reality due to serious limitations on their performance. Information-theoretic studies have shown that the per-node available throughput of the wireless channel can degrade severely with increase in the number of nodes in a flat wireless network [1]. The lack of robustness of the wireless networks is also a major issue. Many envisioned applications of wireless networks are in adverse environments like military operations or disaster relief, situations where the possibility of node failure is high, or the terrestrial wireless channel might be disrupted and connectivity between source-destination pairs might not be available.

Security of the communication in a wireless environment is also very important. Due to the open nature of wireless transmission, any outsider can eavesdrop on the communication, or try to disrupt the communication by injecting, modifying or deleting packets. Traditional methods for data encryption, authentication or message integrity were designed for wired networks where nodes are mostly stationary and connected to power supplies. The solutions could thus be powerful in terms of security without having to consider any constraints on the available energy of the network nodes. However, this approach designing powerful security algorithms that are not energy-conscious, make them unsuitable in the wireless setting, since the wireless nodes usually have a limited amount of energy and also their processing power is less due to the energy constraints. The wired security solutions can actually be security threats in the wireless environment as their execution can lead to a rapid drainage of the wireless node energy and thus render the wireless nodes inoperative.

In this research work, we investigate whether the addition of a satellite overlay network can improve the performance, security and robustness of large wireless networks. We define a satellite overlay network as comprised of high-power terrestrial gateway nodes placed amongst the wireless user nodes and interconnected by high-bandwidth satellite links. We postulate that the addition of such a satellite overlay network can offer significant improvement in performance, network reliability and availability in large wireless networks, and also enable efficient security solutions that are well-suited for wireless nodes with limited resources. Thus, we define a hybrid wireless network as composed of terrestrial wireless LANs with a satellite overlay network, and we try to answer the following questions:

1. how does the satellite overlay network improve the communication performance and network reliability in a hybrid wireless network,
2. can we make use of the satellite overlay to design efficient security protocols that are well-suited for the resource-limited user nodes, and
3. what are the disadvantages, if any, introduced by the satellite overlay and how do we address these disadvantages efficiently.

We address the above questions in our research work in chapters 2 to 6 of this dissertation. A outline of the work contained in chapters 2 to 6 is as follows [1].

1. Through network modeling and simulation, we investigate whether the end-to-end performance for high data-rate traffic is improved in a hybrid wireless network when a satellite overlay is added, compared to a flat wireless network.

2. We develop analytical models to estimate the performance of hybrid networks and demonstrate the use of mathematical tools to allow us to design optimal hybrid network topologies.

3. We analyze the security requirements for group communication in hybrid wireless networks and examine how the presence of the satellite overlay can help to implement better security solutions. We focus on source authentication in group communication, and we design a protocol for multicast source authentication that makes intelligent use of the satellite overlay.

4. We investigate the requirements of wireless multicast routing protocols for group communication in the hybrid network and propose an efficient, topology-aware multicast routing protocol. We integrate the multicast routing protocol with the source authentication protocol to propose a design for secure multicast routing in hybrid networks that is robust to attacks by malicious adversaries.

5. We discuss the various performance enhancement solutions that have been proposed to overcome the high propagation delay of satellite links, and how these per-

[1]In the rest of this dissertation, we use the terms ”hybrid network”, ”hybrid wireless network” and ”hybrid satellite/wireless network” interchangeably.

formance improvements are adversely affected by traditional unicast security protocols like the IP Security Protocol (IPSEC) at the network layer or Secure Socket Layer (SSL) protocol for secure HTTP (HTTPS) at the application layer. Through modeling and simulation, we investigate whether the proposed *layered IPSEC* protocol can overcome the performance problems associated with traditional IPSEC. For secure HTTP, we propose the Dual-SSL protocol as an alternative to SSL and investigate its performance in comparison to SSL.

1.1 Organization of the Dissertation

The rest of the dissertation is organized as follows.

In chapter 2, we describe the hybrid network topologies that we consider and some of their important applications. We also describe the network models that we have built for simulations and give the results of the simulations to compare the performance of the hybrid network topologies with that of flat topologies.

We describe analytical methods to estimate the performance of hybrid networks in chapter 3. Here we show how we can apply loss network models to the hybrid network, and perform fixed point iterations on the loss network model to estimate performance metrics for various network parameters. We also show how we can combine the analytical model with mathematical computational techniques to fine-tune the network parameters and thus do network design.

In chapter 4, we propose a source authentication protocol for the wireless nodes for securing group communication. The source authentication protocol takes advantage of the presence of the satellite overlay network. We demonstrate the major savings in energy and processing delay that are possible with the proposed protocol, compared to commonly-used authentication methods.

In chapter 5, we discuss why secure multicast communication is important in hybrid satellite networks and propose a multicast routing protocol that makes efficient use of the physical hierarchy present in the hybrid network. We also propose a design for secure multicast routing that combines the multicast protocol with the source authentication protocol described in chapter 4.

Chapter 6 deals with the issue of secure unicast communication in satellite networks. We discuss why the introduction of the satellite overlay has a detrimental effect on the performance of unicast communication between a pair of network nodes, and the TCP and HTTP proxy enhancements that are used by satellite networks to mitigate the adverse effects. We show how the network enhancements are broken by the introduction of IPSEC and SSL protocols for unicast security, and we describe novel methods that we propose to allow both the performance enhancements and the security protocols to function simultaneously.

We conclude the dissertation in chapter 7, highlighting the major contributions of our research efforts and by giving a roadmap for future research endeavors related to the work presented in this dissertation.

Chapter 2

Achieving Performance Improvement in a Wireless Network with a Satellite Overlay

2.1 Overview

Deployment of large-scale ad hoc wireless networks suffer from several major problems. Recent studies have shown that the per-node wireless channel throughput in a wireless network is inversely proportional to the square root of the number of nodes in the network [1]. Therefore, as the network increases in size, the individual throughput of the nodes decreases rapidly. Even if the network is connected, the ad hoc routing protocol might fail to find routes between the source and the destination when they are widely separated. For many traffic profiles, a fully wireless network might not be able to satisfy the quality of service (QoS) requirements of the traffic. For example, the end-to-end delay for voice traffic might be unacceptably high. Also, wireless ad hoc networks are ideally suited for applications like military battlefield and disaster relief due to the lack of infrastructure requirement and rapid deployment capability. Such applications are typically in hostile environments where there is a high probability of failure of the wireless nodes. In the event of node failures, the network might get partitioned and the path between sources and destinations might become unavailable. Wireless networks therefore are not robust to node failures.

Based on the above issues with wireless networks, we address the question of feasibility of large-scale wireless networks. We propose that the addition of a satellite overlay network can effectively solve the problems with performance and robustness of wireless networks and make it possible to implement wireless networks with a large number of nodes.

2.2 Methodology

We define a satellite overlay network as consisting of one or more satellite nodes and multiple terrestrial “gateway” nodes that have interfaces for both satellite links and terrestrial wireless communications. For our network model, we consider one satellite in geostationary orbit. We divide the terrestrial network into multiple “clusters” of wireless nodes, with each cluster being served by a gateway node. Ordinary terrestrial user nodes reach the gateway node through multi-hop terrestrial paths. This satellite overlay network provides multiple advantages to the wireless network. The advantages are outlined below.

- The forwarding over the satellite links is single-hop, compared to multi-hop forwarding path on the ground. Thus the overlay provides shortcut paths between a terrestrial source and a destination when the source and destination are separated by several hops.

- The bandwidth of the satellite links is higher than the terrestrial wireless channels, and thus the overlay network provides high-bandwidth alternate paths for the application traffic.

- The satellite is always on, and the characteristics of the satellite links are well-known. In the event of forwarding node failure on the terrestrial path between a source and destination, the satellite overlay provides alternate, forwarding paths. The satellite overlay thus provides reliable communication paths to the terrestrial nodes.

2.2.1 Proposed Network Model

We consider the terrestrial network to be composed of wireless nodes that are grouped into either separate local area networks (LANs), or they are spread in a single large wireless network covering a large area. The terrestrial network has no wired infrastructure. The wireless nodes have limited energy and processing power, and they are also limited in their transmission range. In this chapter, we consider unicast communications between source-destination pairs, and we treat the case of group communications involving multiple sources and multiple destinations in chapter 5. The source and destination can be in different physical LANs, or they might be widely separated in the same wireless network. At a given time, there can be multiple source-destination pairs communicating with one another. Communications between the source and destination pairs are via multi-hop routing paths established with intermediate nodes acting as forwarding routers. The bandwidth available is limited by the maximum bandwidth of the wireless channel, which is shared by all the nodes. The communications can involve a wide range of applications with different QoS requirements:

- Video stream: high bandwidth, low jitter
- Voice: low bandwidth, low jitter

- Data traffic: high reliability, medium delay

Based on the above traffic types and communications capabilities of the user nodes, we propose to add a satellite overlay network for improved performance and network availability in the event of node failures. The satellite overlay network consists of a geostationary (GEO) satellite and multiple terrestrial "gateway" nodes that have interfaces for both satellite links and terrestrial wireless communications. In our present model, we assume each ground cluster is served by one gateway node. A cluster can be a physical wireless LAN, or clusters can be logical subdivisions of a large wireless network, created using different wireless base station set (BSS) identifiers. We assume that the GEO satellite has multiple spot-beams. It is capable of on-board processing and switching between the different spot-beams. The GEO satellite has a large footprint and therefore it can interconnect all the terrestrial LANs in a large area. The satellite is managed from a remote Network Operations Center (NOC) through a dedicated high-bandwidth channel. The NOC has wired broadband link to the Internet. The satellite supports high bandwidth for downlink (approximately 90Mbps) and moderate bandwidth for uplink (approximately 1.5Mbps). These assumptions about the GEO satellite are consistent with the features of several next-generation satellites in development or already deployed (for example, [2]).

The wireless terrestrial network and the satellite overlay together form a hierarchical hybrid network with varying node capabilities and different channel characteristics. Every node in the network, including the satellite, is IP-addressable and can support IP-based protocols. The wireless user nodes in a cluster or LAN communicate with one another and the local gateway node using multi-hop ad hoc routing protocols over the terrestrial wireless channels. The gateway nodes have multiple communication paths to other gateway nodes: either through terrestrial multi-hop wireless paths using ad hoc routing protocols, or in a single-hop over the satellite channel. The satellite overlay thus provides space diversity to the network. Communication between a source and a destination located in different LANs can take one of multiple forwarding paths: multi-hop ad hoc paths through forwarding user nodes (assuming the LANs are connected terrestrially), or multi-hop ad hoc paths to the local gateway, which forwards to the destination gateway either terrestrially or over the satellite links, and from there to the destination node. If the gateway nodes can route terrestrially or through the satellite links, the path selection is based on the end-to-end delay and the throughput required for the data traffic. Each terrestrial wireless LAN creates a mesh network with the local gateway node acting as the core of the mesh network. In the overlay level, the terrestrial paths between the gateway nodes create a mesh architecture, while the satellite defines a star network where the bandwidth can be dynamically distributed amongst the underlying gateway nodes.

2.2.2 Example Hierarchical Topologies

Based on the network model outlined in section 2.2.1, figure 2.1 shows a representative architecture that is suitable for civilian or commercial use. The network comprises wired LANs, wireless LANs with fixed and mobile access points (APs), and mobile ad hoc networks (MANETs) that are connected to one another, and to the wired Internet, through a GEO satellite. Mobile APs serve networks where the infrastructure is not readily available, or a network in a moving vehicle. A subset of the nodes in the MANET

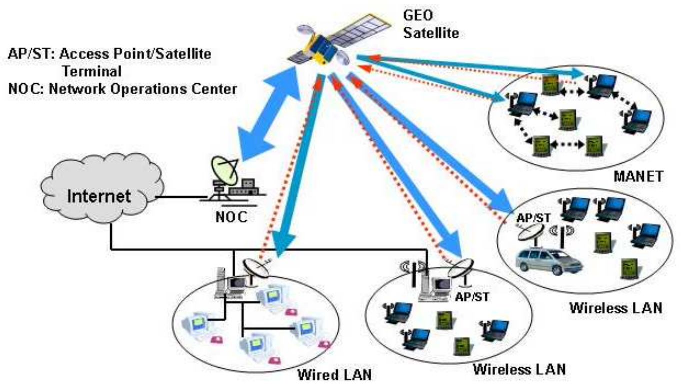
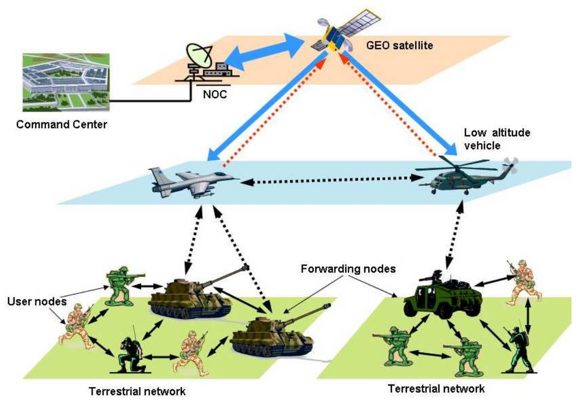
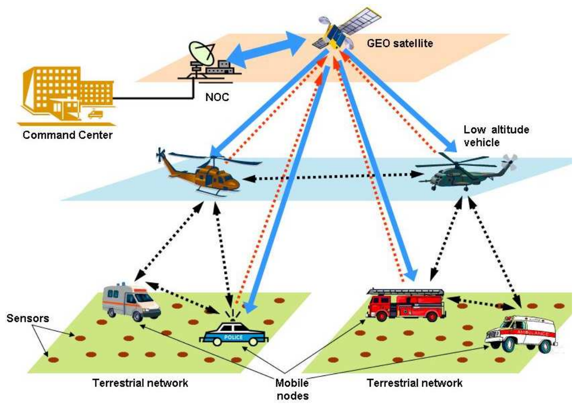


Figure 2.1: Hybrid network topology with a satellite overlay for civilian/commercial applications.

have satellite uplink/downlink capability and can therefore connect to the rest of the hybrid network.



(a) Hybrid network topology for military operations



(b) Hybrid network topology for emergency operations/disaster relief

Figure 2.2: Hybrid wireless network topologies with a satellite overlay for military operations and disaster relief.

Figure 2.2(a) illustrates a hybrid wireless network for military use. The terrestrial segment is composed of MANETs with wireless mobile nodes (e.g., ground soldiers). Each MANET has one or more *forwarding nodes* (FN) with higher capabilities (e.g., armored vehicles). The FNs have wireless communication with both the basic mobile nodes, and with spacecraft flying at low altitudes. The spacecraft are Unmanned Air Vehicles (UAVs) or manned aircraft like helicopters etc, or a combination of both. The basic mobile nodes can communicate with the spacecraft through the FNs. The UAVs and other low-altitude spacecraft have satellite uplink/downlink to a GEO satellite. The different spacecraft can communicate with one another either through horizontal communication links or through the satellite. The satellite also links to a command and control center in a different location, and therefore connects the MANETs and the low altitude spacecraft to the command and control center.

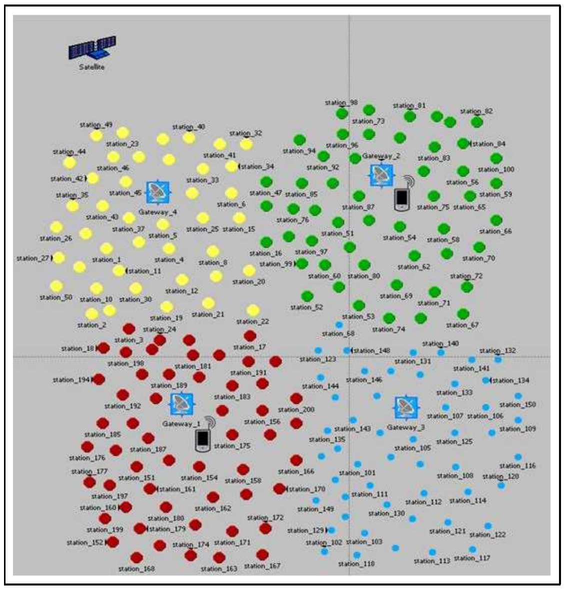
Figure 2.2(b) illustrates a hybrid wireless network suitable for emergency operations like disaster relief. The terrestrial segment is comprised of low-power sensor nodes grouped into clusters. Each sensor node cluster has one or more mobile base station nodes with higher capabilities. The mobile nodes communicate with the sensors, with low altitude spacecraft, and also with a GEO satellite. The base stations also receive command and control messages from the satellite and broadcast the messages to the sensors. The low altitude spacecraft (UAVs and/or helicopters) have satellite uplink/downlink. The satellite connects the low altitude spacecraft and the mobile nodes on the ground to a command and control center. The sensor nodes collect data about conditions on the ground. The mobile nodes, which can be emergency vehicles, process the data collected from the sensors and relays the data via the spacecraft and satellite to the command center. The command center processes the collected information and sends operational commands to the spacecraft and the mobile vehicles to facilitate the disaster relief operations.

2.3 Simulation Model and Results

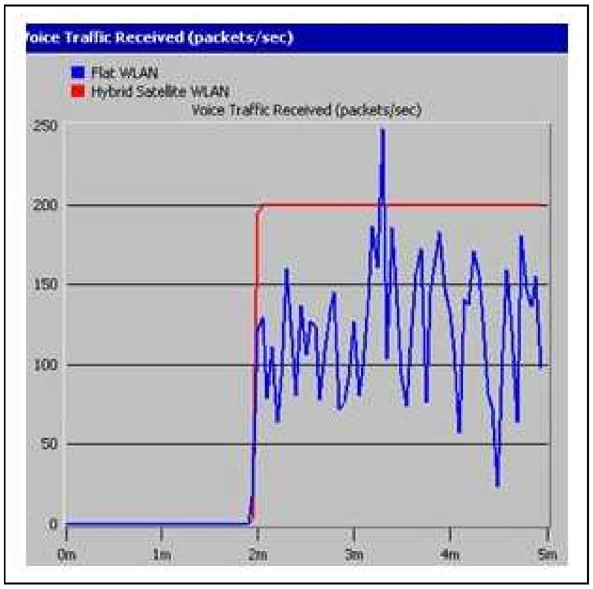
2.3.1 Network Models with Stationary Nodes

We have created models of hybrid wireless networks in Opnet Modeler [3] and obtained performance results for voice and video traffic through simulations. In the network models, we have used IEEE 802.11 11Mbps as the MAC layer for the wireless ground segment. We use AODV as the ad hoc routing protocol for the ground segment. For the satellite overlay network, the ground wireless segment is divided into four clusters, with one gateway in each cluster. The gateways are located within one-hop transmission range of the source and destination nodes. Over the satellite links, we used point-to-point static routing. The satellite is located in geostationary orbit. For the simulation runs, we consider the nodes in the network to be stationary.

We ran simulations for network sizes of 200 nodes and 1065 nodes. Figure 2.3(a) shows the simulation model for the hybrid wireless network with 200 nodes. For this network, there are two source-destination pairs sending medium data rate voice traffic using multi-hop forwarding paths. The voice traffic received by the destinations in the hybrid wireless network with satellite overlay, compared to a similar flat network without the overlay, is shown in figure 2.3(b). For the hybrid network, all the voice traffic is received, while a large percentage is dropped by the flat network. A comparison of the voice traffic parameters is given in figures 2.4 and 2.5. The end-to-end delay for the voice traffic is substantially less in case of the hybrid wireless network, and is mostly due to the propagation delay over the satellite links (figure 2.4(a)). Similarly, the voice traffic delay variation (figure 2.4(b)) and the voice jitter (figure 2.5) are much less when the satellite overlay is used for forwarding the traffic. For the ad hoc routing protocol AODV, the time taken to discover routes to the destination, and the number of hops in the route are shown in figure 2.6. As shown in figure 2.6(a), the satellite overlay provides a reliable forwarding path that is available for the duration of the traffic flow, while the terrestrial forwarding paths change due to route timeouts, channel contention, etc, and thus the forwarding path has to be re-established multiple times. This also contributes to the delay and the data drop. The number of hops is also limited in the case of the overlay forwarding, compared to terrestrial paths (figure 2.6(b)).



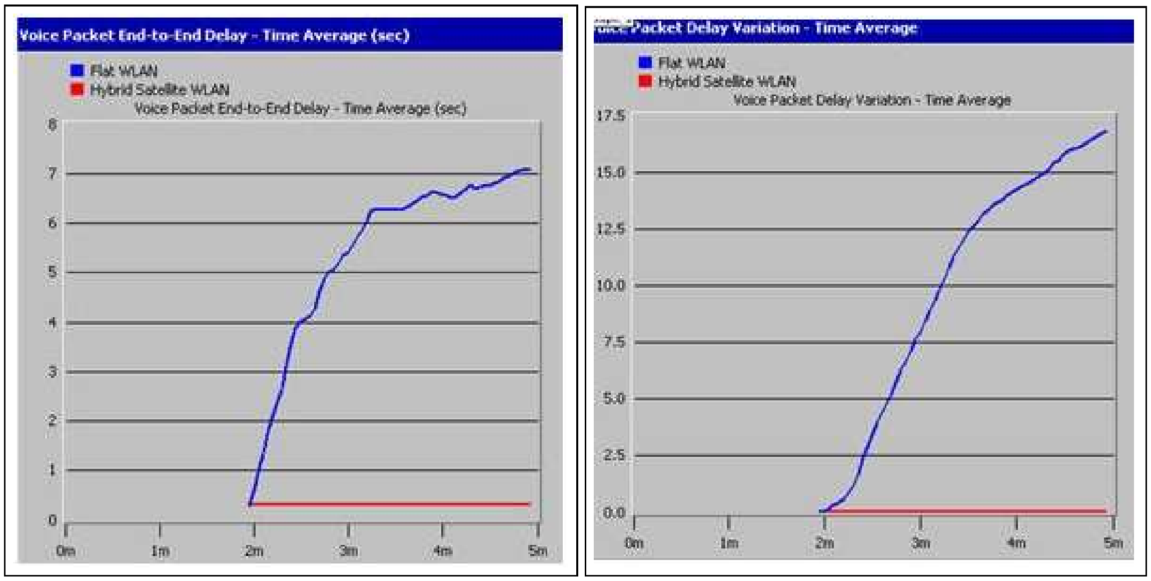
(a) Simulation model for 200-node hybrid wireless network with satellite overlay



(b) Comparison of voice traffic received (X-axis is the simulation duration in minutes)

Figure 2.3: Performance evaluation of 200-node flat wireless network and hybrid wireless network with satellite overlay.

The statistics for the wireless channel for the hybrid overlay network and the flat wireless network are compared in figures 2.7 and 2.8. Figure 2.7(a) shows that even with limited application traffic, the wireless channel for the flat network drops a large amount



(a) End-to-end delay for voice traffic (b) End-to-end delay variation for voice traffic

Figure 2.4: Comparison of voice traffic delay characteristics in 200-node flat wireless network and hybrid wireless network with satellite overlay (X-axis is the simulation duration in minutes).

of the data, while the satellite overlay has no data drops due to the very high bandwidth and the reliable forwarding path it provides. The wireless channel delay is much higher for the flat network as all the nodes contend for the shared medium, whereas for the hybrid network the delay is almost negligible because once the overlay forwarding path is established, all the traffic flows through it and there are no additional resource reservation requests from the sources for the wireless channel (figure 2.7(b)). For similar reason, the overall throughput for the wireless channel for the hybrid network is much less compared to the flat network, since most of the data is transmitted over the satellite links and the wireless channel is not utilized much.

In the network model with 1065 nodes, there are three source-destination pairs. One pair of the sources are located one-hop away from each other, and their respective

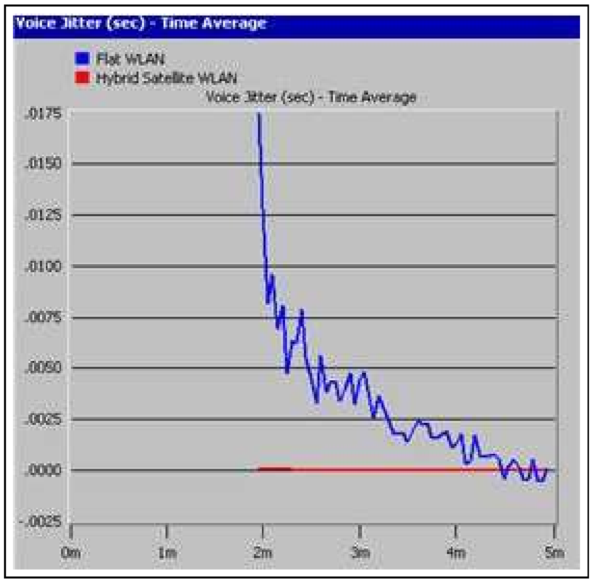
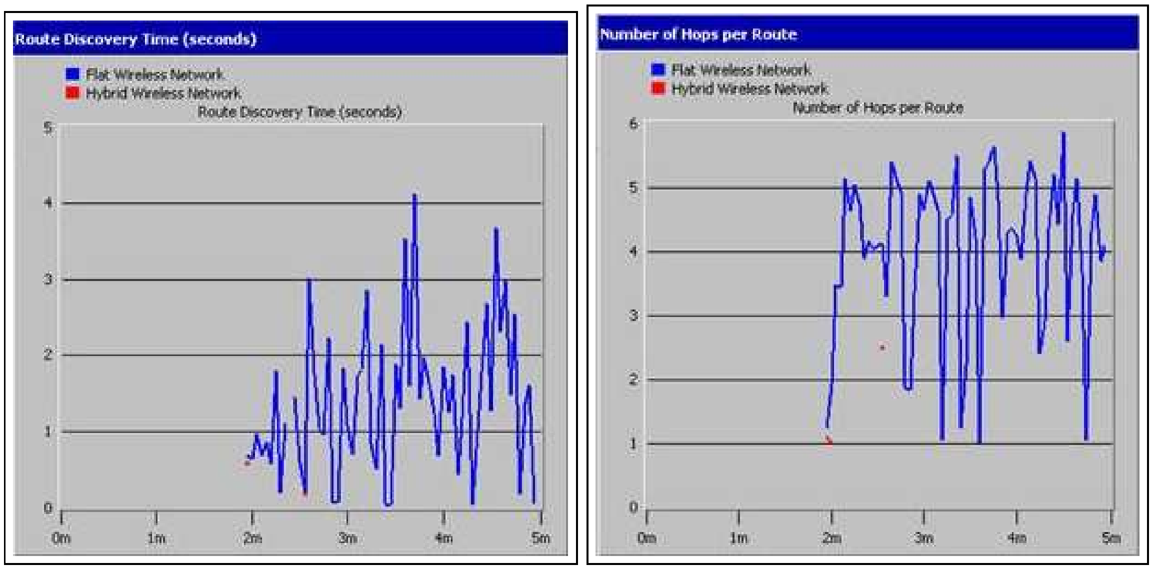


Figure 2.5: Comparison of voice traffic jitter in 200-node flat wireless network and hybrid wireless network with satellite overlay (X-axis is the simulation duration in minutes).

destinations are also one-hop neighbors. These sources and destinations are at the extreme ends of the network, which covers an area of 30km x 30km. The other source-destination pairs are located close to the center of the network. A comparison of the voice traffic sent and received when the network has 1065 nodes is shown in figure 2.9. The graph shows that when the one-hop neighboring sources are transmitting simultaneously, there is a significant traffic drop due to collision in the wireless channel and also due to buffer overflow in the forwarding nodes. When the overlay network is used, the high bandwidth provided by the satellite links ensures that traffic does not get backed up in the gateway, and also there is no collision in the wireless channel. The delay statistics for the voice traffic are given in figures 2.10 and 2.11 and their analyses are similar to the case for the 200-node network.



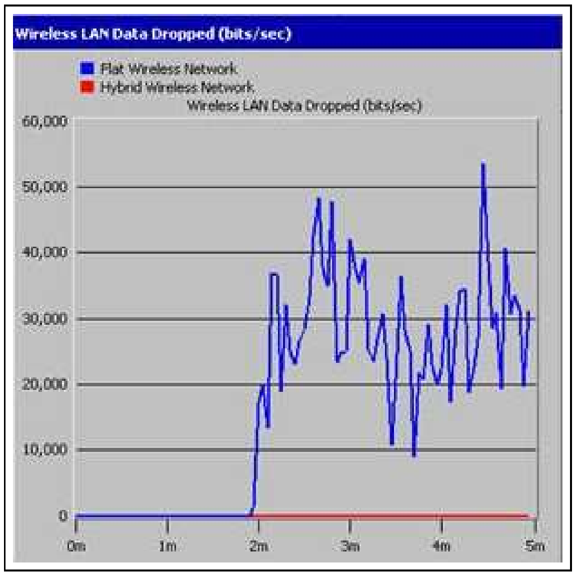
(a) AODV route discovery time (seconds)

(b) AODV number of hops per route

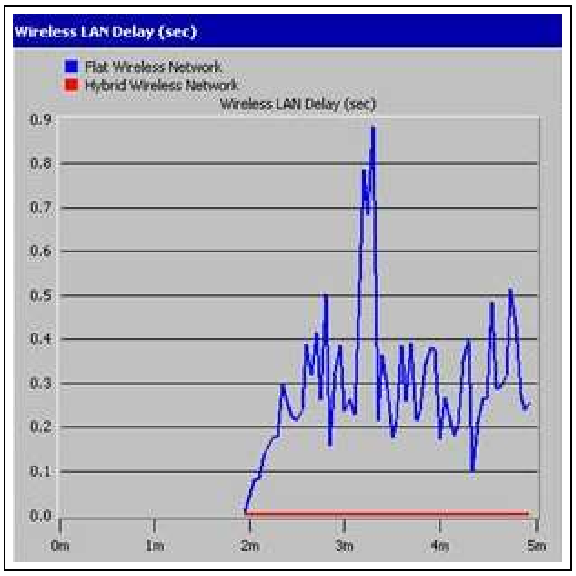
Figure 2.6: Comparison of AODV parameters in 200-node flat wireless network and hybrid wireless network (X-axis is the simulation duration in minutes).

2.3.2 Network Model for Urban Rescue Scenario with Mobile Nodes

We have created a hybrid network topology model for an urban rescue operation scenario (figure 2.12(a)) in Opnet Modeler and ran simulations to evaluate its performance. In this scenario, there are wireless sensor nodes deployed to different locations on the ground to collect and transmit sensory information. There are low altitude spacecraft, Organic Air Vehicles (OAVs), flying over the scenario and generating sensory information. UAVs flying at medium altitude function both as sensors and relays for the data collected by the ground sensors and the OAVs. The UAVs are covered by a GEO satellite that relay the data from the UAVs to a remote command center. Data from the remote command center is also relayed by the satellite and the UAVs to ground vehicles which directs their trajectory based on the information from the sensors. The UAV relays there-



(a) Data dropped by the wireless channel



(b) Wireless channel delay

Figure 2.7: Comparison of the wireless channel statistics in 200-node flat wireless network and hybrid wireless network (X-axis is the simulation duration in minutes).

fore act as the gateway nodes for the ground sensors and the OAVs. The ground sensors, OAVs and the ground vehicles constitute a mobile ad hoc mesh network that increases the possibility of finding paths from the ground sensors to the UAV relays. Each relay defines a local star network where the bandwidth can be dynamically distributed between the underlying nodes. The UAV relays follow the movement of the ground nodes to provide constant coverage. The number of UAV relays and the location of the UAV relays, is determined by the heuristic placement algorithm proposed in [4]. The ground sensors have only a single terrestrial wireless interface, while the OAVs and the vehicles have two interfaces - one for terrestrial wireless communication, while the other for communicating with the UAV relays. The UAV relays have two interfaces each - one for talking to the underlying spacecraft and vehicles, while the other for communicating with the satellite.

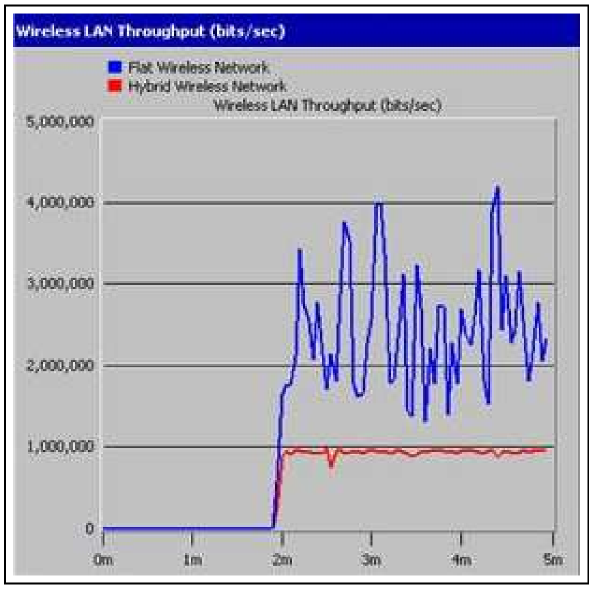
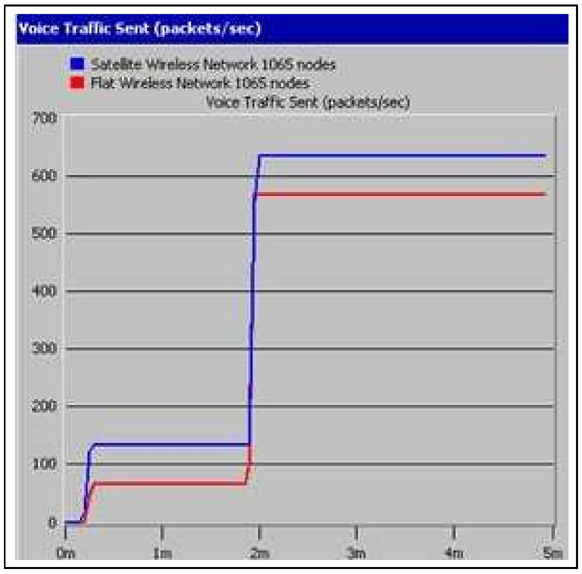
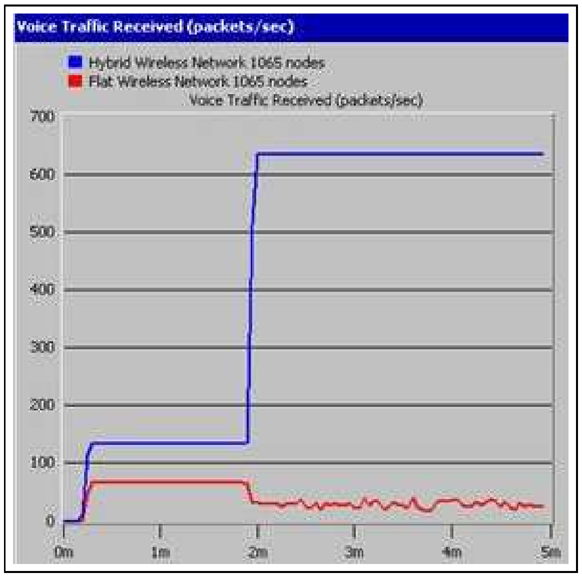


Figure 2.8: Comparison of the wireless channel throughput in 200-node flat wireless network and hybrid wireless network (X-axis is the simulation duration in minutes).

The simulation model in Opnet is shown in figure 2.12(b). The number of ground nodes, their locations and movement patterns are obtained from US military data. The network has 118 ground sensors, 122 OAVs and 28 vehicles. The UAV placement algorithm [4] generated 4 UAV relays for the network at specific locations. The altitude of the UAV relays is 4.5km, while the communication range with each OAV is set at 6km. DSR is used as the ground routing protocol, while the ground MAC is IEEE 802.11 1Mbps. The bandwidth for the UAV relay to OAV link is 2.5Mbps each. The UAVs do static routing of the data. The UAV MAC protocol and the satellite access are both reservation-TDMA. For the simulations, the nodes are configured with custom traffic. On average, 20 OAVs generate streaming traffic simultaneously and send to the UAV relays for further transmission to the satellite. 8 ground sensors generate detection frames in random times



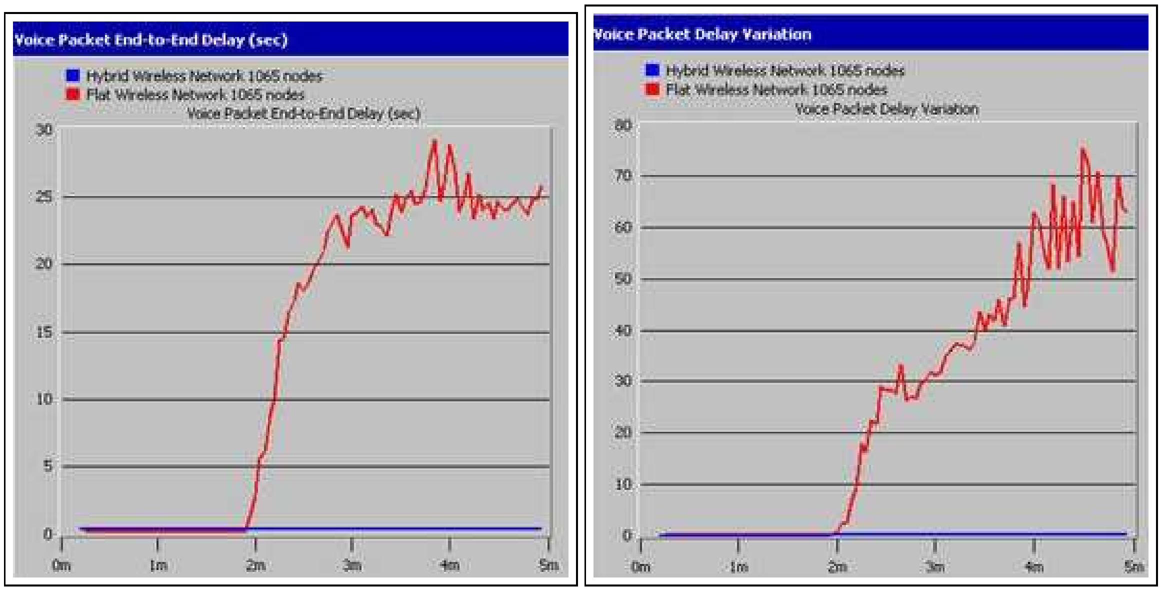
(a) Voice traffic sent



(b) Voice traffic received

Figure 2.9: Comparison of voice traffic sent and received in a 1065-node flat wireless network and a hybrid wireless network with satellite overlay (X-axis is the simulation time in minutes).

throughout the simulation and send to the UAV relays through the OAVs. The traffic generated by the ground nodes is shown in figure 2.13(a). The ground sensors generate about 25% of the total traffic, while the rest is generated by the OAVs. The delay is mostly due to the ground sensor traffic when a sensor node cannot connect to a OAV. The node continues to generate route requests until an OAV comes within transmission range. Figure 2.13(b) shows that high traffic generation by the ground sensors loads the wireless LAN. The throughput is sometimes greater than the link capacity of 1Mbps. This is because not all the ground nodes interfere with one another and sometimes two or more sensors can transmit simultaneously by sending traffic through separate OAVs (space diversity). The larger delays (figure 2.13(c)) are mainly due to the delays of the ground sensor frames for multi-hop paths and also due to the storage delay when a path is not available initially but



(a) Voice packet end-to-end delay (b) Voice packet delay variation

Figure 2.10: Voice traffic delay statistics in a 1065-node flat wireless network and a hybrid wireless network with satellite overlay (X-axis is the simulation time in minutes).

is established later. The delays due to the OAV frames are much smaller in comparison.

2.4 Summary

In this chapter, we have examined whether adding a satellite overlay network to interconnect large terrestrial wireless networks can lead to improvements in the network throughput. We have defined the network topology thus created as a hybrid satellite/wireless network. We have given examples of such networks and highlighted some of their important applications. We have built simulation models of the proposed hybrid network architecture, both for stationary wireless nodes and for mobile wireless nodes. The results of discrete event simulations clearly demonstrate that there is major improvement in end-to-end traffic delivery and delay characteristics when the overlay network is used,

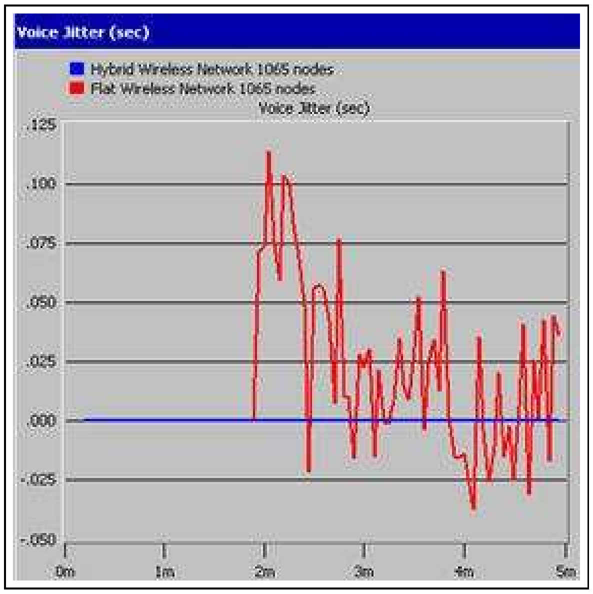
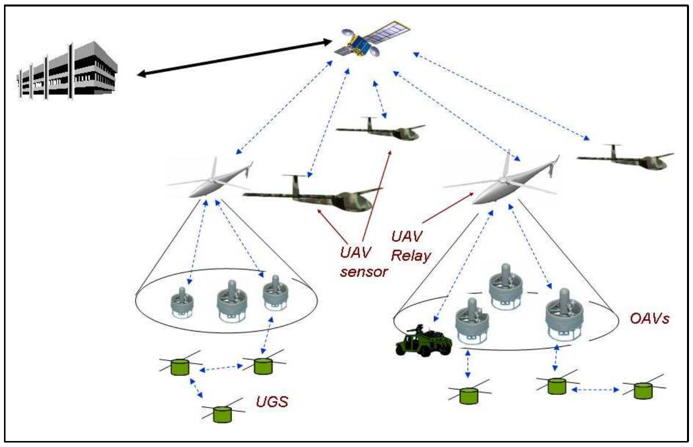


Figure 2.11: Voice traffic jitter in a 1065-node flat wireless network and a hybrid wireless network with satellite overlay (X-axis is the simulation time in minutes).

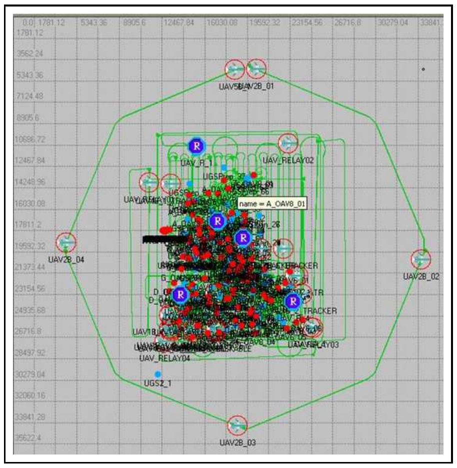
in comparison to flat wireless networks.

2.5 Related Work

Gupta and Kumar have obtained theoretical results that show that the capacity of wireless networks improve by a factor of 1.5 when the networks are 3-dimensional, for example, hierarchical networks with a satellite backbone [5]. In [6], the authors have considered a hybrid terrestrial cellular network with a satellite backbone and have formulated a multi-faceted cost function composed of call-blocking and dropping probabilities, to determine the optimal channel partitioning between the cellular and the satellite systems and to decide on the optimal call assignment policy. They have obtained optimal solutions for sub-problems of the original complex optimization problem, whereby they



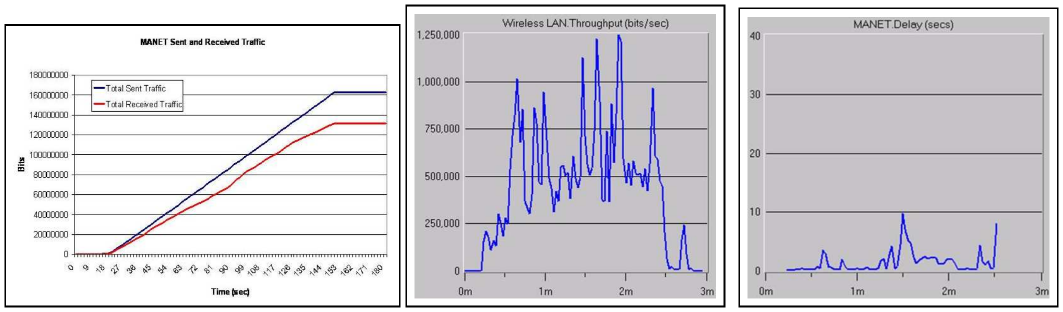
(a) Urban rescue operation network model



(b) Network simulation model for the hierarchical architecture

Figure 2.12: A hybrid wireless architecture for urban rescue operations with satellite-UAV overlay

conclude that double coverage through both cellular and satellite systems, results in substantial improvement over pure terrestrial or pure satellite systems. Dousse et al [7] study the connectivity properties of large-scale ad hoc and hybrid wireless networks where fixed base stations can be reached in multiple hops. The authors find that the introduction of a sparse network of base stations increases the connectivity to a large extent. They also show that bottlenecks become unavoidable at low spatial densities. Ryu et al [8] have proposed an architecture for multi-tier mobile ad hoc networks and developed simulation tools for this hybrid network. To address the challenges of connectivity asymmetry and node heterogeneity, the paper proposes a cross-tier MAC protocol for the access layer, the multi-virtual backbone protocol for ad hoc routing using the hierarchical backbone architecture, and a multi-modal TCP protocol. Wu et al [9] have modeled a hierarchical low earth orbit/medium earth orbit (LEO/MEO) satellite network using generalized stochastic Petri nets and obtained performance results of the Petri nets model through Opnet sim-



(a) MANET Sent and Received Traffic (b) Wireless LAN throughput (c) End-to-end traffic delay

Figure 2.13: Simulation results for urban rescue scenario hybrid wireless network with satellite-UAV overlay

ulations. The paper demonstrates that the double-layered satellite network outperforms single-layered ones for heavy traffic loads. [10] describes the European CAPANINA project, which is investigating the viability of integrating difficult-to-reach areas into the broadband Internet using different types of aerial platforms. The paper describes the work that is being done on possible applications and services for this hybrid architecture, on solutions to integrate the aerial platforms into the network architecture, and on methods to use the aerial platforms to deliver broadband content to high-speed moving vehicles. The role of satellite networks in future telecommunications networks and service provisioning is examined in [11]. The authors argue that the future of satellite systems is to complement terrestrial networks to provide multimedia services to fixed and mobile systems in an integrated architecture. Jetcheva et al [12] have designed *Ad Hoc City*, a multi-tier mobile ad hoc network architecture for wide-area communication, where the backbone network is a mobile multi-hop network composed of wireless devices mounted on moving vehicles. The design integrates cellular networks with ad hoc networking and uses a modified version of DSR for unicast routing.

Chapter 3

Performance Modeling of Hybrid Satellite/Wireless Networks Using Fixed Point Approximation and Sensitivity Analysis for Network Design

3.1 Overview

There is a lack of systematic methodologies and toolkits for the design and dimensioning of hybrid satellite/wireless networks such that there can be predictable bounds on performance, as measured by key performance metrics. Due to the variability in performance of the wireless links at the terrestrial level, and the interdependence of the link data rates at the terrestrial level and the satellite overlay, performance analysis and network design for hybrid networks is a complicated task.

Our objective is to develop analytical and numerical models, or a combination thereof, which are simple to use but can efficiently approximate, to a great degree of accuracy, hybrid network performance. These models have several applications in the design and analysis of hybrid networks - for example, in the evaluation of protocol performance and robustness, and component-based design and parameter tuning for optimal performance.

We propose a novel approach based on the fixed point approximation method and loss network models for performance evaluation and design optimization for hybrid networks. The primary challenges in developing loss network models for hybrid satellite/wireless networks are (i) the coupling between wireless links in the terrestrial segments due to the transmission interference between neighboring nodes, and (ii) the coupling of loss and data rates between the terrestrial segments and the satellite overlay. In the terrestrial segments, we propose to approximate interference and contention as inter-link traffic dependent loss factors by using probabilistic physical and MAC layer models developed in [13]. This approximate model provides a system of equations describing the relations between reduced link rates. For the satellite overlay, we use the Gilbert-Elliot loss model [14, 15] to derive a relationship between traffic arrival rate and the link loss on the satellite channel. We then use the hierarchical reduced load network model from [16], to create an overall network model for the hybrid network that couples the arrival rates in each layer with the associated loss rates, and interconnects the arrival rates and loss rates between the layers. We then use a fixed point approximation method for this set of relations to derive a solution that converges to a fixed point for the set of parameters arrival rate and link loss, while satisfying all the equations in the set. The result is an implicit model of the selected performance metric (for example, throughput), as a function of the network design variables, for example, path routing probabilities, node power levels or link bandwidths.

In addition to the performance model, we are interested in developing a methodology for design of hybrid networks, through sensitivity analysis of performance metrics for parameter optimization and robustness evaluation. We use Automatic Differentiation (AD) [17] for the sensitivity analysis of performance metrics. AD is a powerful method to numerically compute the derivatives of a software-defined function (i.e. a computer program implementation of the function). The analysis model that we generate based on the loss network models and fixed point iterations, is the input function to the AD and the output of the AD is the partial derivative of the performance metric (for example, throughput) with respect to defined input parameters (which in our case are the network design variables). As an example, we show how we can use this methodology to find the optimal load distribution amongst multiple paths between source-destination pairs to maximize throughput. In this example the gradient projection algorithm is used to find the optimal load distribution, and AD is used to compute the gradient of the network throughput with respect to the load distribution parameters.

The rest of this chapter is organized as follows. In section 3.2 we briefly describe the hierarchical loss network model from [16] that we apply to the hybrid satellite/wireless network. Section 3.3 describes the analytical models that we use for the wireless terrestrial section of the hybrid network. The description in section 3.3 is originally from [18]. Section 3.4 describes the analytical models that we use for the satellite overlay segment of the hybrid network. We detail the fixed point approach to the problem in section 3.5, which integrates the models developed in the two preceding sections. Once the hierarchical reduced load model has been developed, we discuss in section 3.6 how to use

3.2 Generalized Loss Network Model for Hierarchical Networks

We briefly describe the hierarchical loss network model for fixed point approximation that was proposed in [16], since we use this model to build our network model.

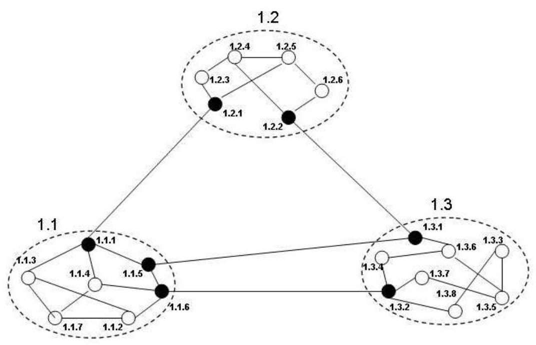


Figure 3.1: Hybrid network with three clusters - lower layer abstraction

3.2.1 Network Abstraction

For the loss network model design we consider large networks that have either physical or routing hierarchies in the architecture, as shown in the abstract network example in figure 3.1. The three dashed regions in the example each represent a cluster in the lower layer, which are labeled $1.x$, with $x \in \{1, 2, 3\}$. 1.1 signifies Layer 1, Peer Group 1. Each node has an address too - for example, 1.2.5 represents node 5 in peer group 2 of layer 1. Each peer group or cluster has one or more border nodes, which are shown in black, and multiple non-border nodes, which are shown in white. Node aggregation and network abstraction are done as follows:

- All border nodes are kept in the higher layer (layer 2 in this example).

- Border nodes in the same cluster are fully connected via “logical links”.

This results in the higher layer abstraction as shown in figure 3.2.

Mapping the hybrid satellite/wireless network to this model is straightforward. The terrestrial wireless networks are analogous to the layer one clusters in the example above. The gateway nodes in each wireless network are the border nodes; the satellite overlay comprising the gateway nodes and the satellite form the higher layer.

3.2.2 Hierarchical Model for Fixed Routing

3.2.2.1 Notations

$G(1.n)$: n^{th} cluster/peer group in layer 1, where $n = 1, ..., N_1$ and N_1 is the total number of clusters.

$1.n.x_i$: node x in cluster $G(1.n)$, where $i = 1, ..., X_n$ and X_n is the total number of nodes in $G(1.n)$.

$1.n.y_i$: border nodes in cluster $G(1.n)$, where $i = 1, ..., Y_n$ and Y_n is the total number of border nodes in $G(1.n)$.

$1.n_1.x_1 \longrightarrow 1.n_2.x_2$: directional link from node $1.n_1.x_1$ to node $1.n_2.x_2$.

$\lambda_s(1.n_1.x_1 \longrightarrow 1.n_2.x_2)$: offered load for class s traffic from source $1.n_1.x_1$ to destination $1.n_2.x_2$, where $s = 1, ..., S$ and S is the total number of traffic classes. It is also written as λ_{ps} with p as the p^{th} source-destination pair.

$P : (1.n_1.x_1 \longrightarrow 1.n_2.x_2)$: the route between node pair $\langle 1.n_1.x_1, 1.n_2.x_2 \rangle$. P_p is the route between the p^{th} node pair.

3.2.2.2 Route Segments

The route between a source-destination pair is broken down into route segments whenever the source and destination are in different clusters. This is done to segregate local computation within a cluster from the higher layer computation between clusters.

Each cluster that the route traverses has its own segment. Therefore, a route P :

$(1.n_1.x_1 \longrightarrow 1.n_2.x_2)$ has the following segments assuming that $n_1 \neq n_2$ and that neither $1.n_1.x_1$ nor $1.n_2.x_2$ is a border node:

$$\begin{aligned}
&P^1 : (1.n_1.x_1 \longrightarrow 1.n_1.y_2) \\
&P^2 : (1.n_1.y_2 \longrightarrow 1.n_i.y_1) \\
&P^3 : (1.n_i.y_1 \longrightarrow 1.n_i.y_2) \\
&... \\
&P^{k-1} : (1.n_j.y_2 \longrightarrow 1.n_2.y_1) \\
&P^k : (1.n_2.y_1 \longrightarrow 1.n_2.x_2)
\end{aligned} \tag{3.1}$$

where y_1 represents a border node through which traffic enters a cluster, while y_2 represents a egress border node for traffic. The set of route segments for the p^{th} source-destination pair is denoted by $P_p^{'} = \{P_p^1, ..., P_p^k\}$.

3.2.2.3 Initial Offered Load and Local Relaxation

Let $\lambda_s^0(1.n_1.x_1 \longrightarrow 1.n_2.x_2)$ be the offered load for class s traffic of the p^{th} node pair $\langle 1.n_1.x_1, 1.n_2.x_2 \rangle$. Therefore for each of the route segments in $P_p^{'} = \{P_p^1, ..., P_p^k\}$, the initial offered load is:

$$\begin{array}{lr}
\lambda^0_{ps}(1.n_1.x_1 \longrightarrow 1.n_1.y_2) & \textit{source cluster traffic} \\
\lambda^0_{ps}(1.n_1.y_2 \longrightarrow 1.n_i.y_1) & \textit{inter-cluster traffic} \\
\lambda^0_{ps}(1.n_i.y_1 \longrightarrow 1.n_i.y_2) & \textit{inter-cluster traffic} \\
\ldots & \\
\lambda^0_{ps}(1.n_j.y_2 \longrightarrow 1.n_2.y_1) & \textit{inter-cluster traffic} \\
\lambda^0_{ps}(1.n_2.y_1 \longrightarrow 1.n_2.x_2) & \textit{destination cluster traffic}
\end{array} \tag{3.2}$$

Each term above takes the value of the initial offered load $\lambda^0_s(1.n_1.x_1 \longrightarrow 1.n_2.x_2)$. Thus the same initial load is assigned to all segments.

For the i^{th} cluster, the aggregate initial offered load between a node pair, for traffic class s, is the sum of the class s traffic that passes through this node pair for all source-destination pairs:

$$\lambda^1_s(1.n_i.x_1 \longrightarrow 1.n_i.y_2) = \sum_{\{p:(1.n_i.x_1 \longrightarrow 1.n_i.y_2) \in P'_p\}} \lambda^0_{ps}(1.n_i.x_1 \longrightarrow 1.n_i.y_2) \tag{3.3}$$

where we assume, without loss of generality, that the destination node is a border node. In equation (3.3), we also assume the initial condition of zero blocking in remote clusters and between clusters.

The reduced load model for fixed routing is then applied to every cluster individu-

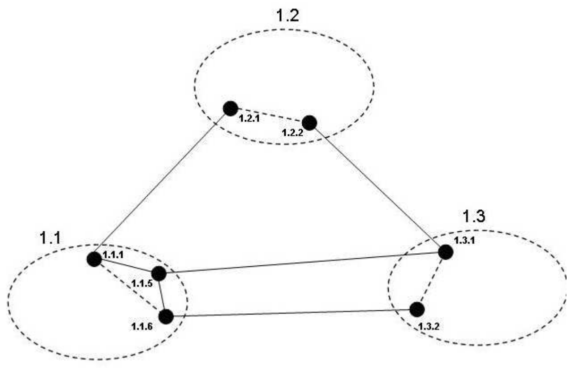


Figure 3.2: Hybrid network with three clusters - higher layer abstraction.

ally using the above offered loads to calculate group-wide blocking probabilities:

$$\begin{aligned}
&B_s(P^1) = B_s(1.n_1.x_1 \longrightarrow 1.n_1.y_2) \\
&\ldots \\
&B_s(P^3) = B_s(1.n_i.y_1 \longrightarrow 1.n_i.y_2) \\
&\ldots \\
&B_s(P^k) = B_s(1.n_2.y_1 \longrightarrow 1.n_2.x_2)
\end{aligned} \tag{3.4}$$

3.2.2.4 Reduced Load and Higher Layer Relaxation

The network consists of only the border nodes in the higher layer. As shown in figure 3.2, the abstraction in the higher layer is a new network composed of the border nodes, the inter-group links and the logical links within the clusters. In this layer, the route segments from equation (3.1) are consolidated into three parts:

- the source cluster segment P^1,

- a new segment that comprises all the clusters between the source and destination clusters: $P^0 = P^2 \bigcup ... \bigcup P^{k-1}$,
- the destination cluster segment P^k.

The aggregate offered load in the higher layer (between the egress gateway node in the source cluster and the entry gateway node in the destination cluster) is the initial offered load thinned by blocking in both the source and destination clusters:

$$\begin{aligned}
&\lambda_s^1(1.n_1.y_2 \longrightarrow 1.n_2.y_1) = \\
&\lambda_s^0(1.n_1.y_2 \longrightarrow 1.n_2.y_1) + \\
&\sum_{\{p:(1.n_1.y_2 \longrightarrow 1.n_2.y_1) \in P_p'\}} \lambda_{ps}^0(1.n_1.y_2 \longrightarrow 1.n_2.y_1). \\
&B_s(1.n_1.x_1 \longrightarrow 1.n_1.y_2).B_s(1.n_2.y_1 \longrightarrow 1.n_2.x_2)
\end{aligned} \tag{3.5}$$

that is,

$$\lambda_s^1(P^0) = \lambda_s^0(P^0) + \sum_{\{p:P^0 \in P_p'\}} \lambda_{ps}^0(P_p^0).B_s(P_p^1).B_s(P_p^k) \tag{3.6}$$

Equation (3.6) gives the complete traffic input in the higher layer. Now running the reduced load approximation algorithm at this layer gives the blocking probability between the remote nodes:

$$\begin{aligned}
&B_s(1.n_i.y_1 \longrightarrow 1.n_i.y_2) && \textit{blocking in intermediate cluster} \\
&B_s(1.n_i.y_2 \longrightarrow 1.n_j.y_i) && \textit{blocking between clusters}
\end{aligned} \tag{3.7}$$

3.2.2.5 Updating Lower Layer Load With Higher Layer Loss

The higher layer blocking probabilities obtained in equation (3.7) are fed back to the lower layer offered load in (3.3) to accurately reflect the fact that the original offered load is thinned by blocking in the remote clusters and the inter-cluster links:

$$\begin{aligned} &\lambda_s^1(1.n_i.x_1 \longrightarrow 1.n_i.y_2) = \\ &\sum_{\{p:(1.n_i.x_1 \longrightarrow 1.n_i.y_2) \in P'_p\}} \lambda_{ps}^0(1.n_i.x_1 \longrightarrow 1.n_i.y_2). \\ &\prod_{\{k:P^k \neq (1.n_i.x_1 \longrightarrow 1.n_i.y_2)\}} B_s(P_p^k) \end{aligned} \tag{3.8}$$

Equation (3.8) becomes the new input for local relaxation at the lower layer. Fixed point approximations at the local and higher layers are then repeated till the difference between the results from successive iterations are within a certain criteria.

3.2.3 Summary Outline of the Fixed Point Approximation Algorithm

Based on the performance model described in section 3.2.2, the algorithm for fixed point approximation of hierarchical networks can be summarized in the following steps.

For each source-destination pair:

1. Divide the traffic route between a source-destination pair into segments.

2. Assign the same initial load to all the segments.

3. For any intermediate segment, aggregate the offered load on the path between boundary nodes as the sum of the initial load and the offered loads for all source-

destination pairs whose traffic route includes the path in question (for a given class of traffic).

4. Run the fixed point approximation in each segment separately to compute the blocking probability (or equivalently, loss factor) in each segment.

5. Abstract the higher layer into three segments - the source segment, destination segment, and one intermediate segment which is the union of all intermediate clusters and inter-cluster links.

6. For the aggregate intermediate segment in the higher layer, the aggregate load on any path is the sum of the initial offered load on that path and the aggregate of the offered load for each source-destination pair whose traffic route includes the path in question, thinned by blocking (or equivalently, loss) in both source and destination clusters for each source-destination pair.

7. Run the fixed point approximation in the intermediate segment of the higher layer to obtain the blocking probability between border nodes with each cluster and between the clusters.

8. Update the initial aggregate offered load in each lower layer cluster with the blocking probabilities computed from the higher layer approximation.

9. Iterate the computations in the lower and higher layers till the results converge.

3.3 Physical and Link Layer Model for the Terrestrial Wireless Segment

In the wireless segment, we assume that the MAC layer protocol is IEEE 802.11. In [13] Bianchi describes a simple but accurate analytical model to compute the IEEE 802.11 throughput, under the assumptions of finite number of terminals and ideal channel conditions. The analysis applies to both packet transmission schemes used in 802.11, namely, the basic access mechanism and the RTS/CTS access mechanism. We use Bianchi's model to approximate the MAC layer induced loss factor that we use in our performance model. In the following we briefly describe Bianchi's model, as outlined in [19].

Consider a fixed number n of contending connections. In Bianchi's model every connection is in saturation condition, i.e. it always has a packet available for transmission. This is appropriate for our problem and method, since we are interested in design and performance bounds. Since all packets are consecutive and there is no empty queue, each packet needs to wait for a random backoff time before the transmission attempt. Let W be the minimum contention window size and m be the maximum backoff stage, then the maximum contention window size is $2^m W$.

Let $b(t)$ be the stochastic process representing the backoff time counter for a given connection and $q(t)$ the stochastic process representing the backoff stage of the connection at time t. Let τ be the stationary probability that the connection transmits a packet in a generic time slot. It is also assumed that at each transmission attempt, regardless of the previous retransmissions numbers, each packet collides with constant and independent probability p. The random process $(q(t), b(t))$ is modelled as a two-dimensional discrete-time Markov process. Hence, we can compute the state transition probabilities and the stationary distribution of the corresponding Markov chain. We can also obtain the following relations [13] between the key parameters p and τ:

$$\begin{aligned} p &= 1-(1-\tau)^{n-1} \\ \tau &= \frac{2(1-2p)}{(1-2p)(W+1)+pW(1-(2p)^m)} \end{aligned} \tag{3.9}$$

Note that the above two equations represent a nonlinear system in the two unknowns, which can be solved using numerical techniques. The system has a unique solution. To reproduce Bianchi's result, we use a fixed point method to solve those two equations. For each value of n we solve these equations once and use the results in our fixed point model.

Let S be the normalized system throughput, defined as the fraction of *time* the channel is used to successfully transmit payload bits. Next step is to compute P_{tr}, the probability of at least one transmission in one time slot and P_s the probability that a transmission occurring in the channel is successful.

$$\begin{aligned} P_{tr} &= 1-(1-\tau)^n \\ P_s &= \frac{n\tau(1-\tau)^{n-1}}{1-(1-\tau)^n} \end{aligned} \tag{3.10}$$

The throughput S becomes,

$$S = \frac{P_s P_{tr}}{(1-P_{tr})\sigma + P_{tr}P_sT_s + P_{tr}(1-P_s)T_c} \tag{3.11}$$

Here, T_s and T_r are the average times the channel is busy due to successful transmission and collision respectively. The parameter σ is the duration of an empty slot and $E[P]$ is the expected packet length. To specifically compute the throughput for a given Distributed Coordination Function (DCF) scheme it suffices to specify T_s and T_c.

For a system completely based on the basic access mechanism, with $H = PHY_{hdr} + MAC_{hdr}$ being the packet header, δ the propagation delay, and with fixed packet size P, we have,

$$\begin{aligned} T_s^{bas} =& H + P + SIFS + \delta + ACK \\ & + DIFS + \delta \\ T_c^{bas} =& H + P + DIFS + \delta \end{aligned} \tag{3.12}$$

For a system based on the RTS/CTS access mechanism, collision can only occur for RTS frames, hence,

$$\begin{aligned} T_s^{rts} =& RTS + SIFS + \delta + CTS + SIFS \\ & + \delta + H + P + SIFS + \delta + ACK \\ & + DIFS + \delta \\ T_c^{rts} =& RTS + DIFS + \delta \end{aligned} \tag{3.13}$$

Note that the slot time σ is also given in Table 1 of [13].

The saturation throughput of IEEE 802.11 depends on the number of contention nodes in the same neighborhood.

3.4 Analytical Model for the Satellite Channel

In this section we describe the channel loss model for the satellite link that we consider for the fixed point algorithm in the higher layer. We consider that for transmission the gateway nodes access the satellite channel using a multiple access protocol and the loss in the channel is due to physical conditions only. We also describe the multiple access protocol that we use for computing the channel throughput for the fixed point algorithm in the higher layer.

3.4.1 Satellite Channel Loss Model

We model the satellite channel by a threshold-based 2-state Markov chain. In this model the channel is either in "GOOD" state, if the transmitted signal experiences less than Γ dB attenuation, or it is in "BAD" state, if the signal fade is more than Γ dB, where Γ is the fade attenuation threshold [14, 15]. We assume that if the channel is in "GOOD" state, channel coding is capable of correcting all bit errors and for simulation purposes the probability of bit error at the output of the channel decoder is zero. In the "BAD" state, the channel behaves as a Binary Symmetric Channel (BSC) with bit error probability equal to ϵ at the output of the channel decoder. The channel state is characterized by the transition probability matrix given by:

$$M = \begin{bmatrix} g & (1-g) \\ (1-b) & b \end{bmatrix} \tag{3.14}$$

where g and $1-b$ are the probabilities that the channel will be in the "GOOD" state at the k^{th} symbol duration, given that the channel was in "GOOD" and "BAD" states, respectively, at the $(k-1)^{st}$ symbol duration. Using the results of the ACTS Propagation Experiments [20], for a fade attenuation threshold of Γ = 10 dB., we have g = 0.9999813, and $1-b$ = 0.00172.

In order to develop a loss model in section 3.5, we need to calculate the bit error probability at the output of the channel decoder. In order to estimate the value of ϵ, we use the link budget calculations of a commercial satellite system proposed in [2]. The calculations confirm that for signal attenuation of less than 10 dB, link budget margins and channel coding are capable of keeping the bit error probability around 10^{-9} at the output of the channel decoder. For signal attenuation of more than 10 dB, on the other hand, bit errors occur with probability $\epsilon \geq 0.1$ at the decoder output of a typical concatenated channel encoder/decoder pair which employs a RS(204,188) Reed-Solomon outer code and rate-punctured inner convolutional codes (capable of supporting rates 1/2, 2/3, 3/4, 5/6, 7/8 to compensate for fading) [21, 22]. Using this result, we evaluate the time progress of the channel state for every bit transmission and assume that a bit is in error with probability $\epsilon = 0.1$ when the channel is in “BAD” state and no errors occur when the channel is in “GOOD” state. Since these errors are at the output of the channel decoder, we further assume that a packet is corrupted if at least one bit is in error. We use this packet loss pattern in our fixed point algorithm.

3.4.2 Multiple-Access Throughput Model for the Satellite Channel

To compute the throughput of the satellite channel on the uplink, we assume that multiple gateway nodes contend for the channel during transmission using the *slotted*

ALOHA multiple access scheme [23]. Therefore if λ_n is the total traffic for n gateway nodes transmitting, the satellite channel throughput is given by:

$$S = \lambda_n e^{-\lambda_n} \tag{3.15}$$

The multiple-access capacity of the satellite channel is then given by [24]:

$$C_a = \lambda_n e^{-\lambda_n}.W.log(1 + \frac{P}{\lambda_n N}) \tag{3.16}$$

where W is the channel bandwidth in hertz and P/N is the average signal-to-noise power ratio of the channel.

We use the multiple-access capacity C_a in computing the channel loss factor from the traffic rate in section 3.5.2.2, equation (3.27).

3.5 The Fixed Point Algorithm for the Hybrid Network

We follow the methodology described in section 3.2 for the performance model of the hybrid network. We assume unicast communication with the source and destination nodes being in separate terrestrial clusters. For example, as shown in the network representation in figure 3.3, we have a source S in cluster 1.1 and a destination D in cluster 1.3.

3.5.1 Lower Layer Loss Model and Fixed Point Method

We assume IEEE 802.11 RTS/CTS as the link layer protocol in each terrestrial cluster and thus model the relation between the traffic rate and the link losses using the

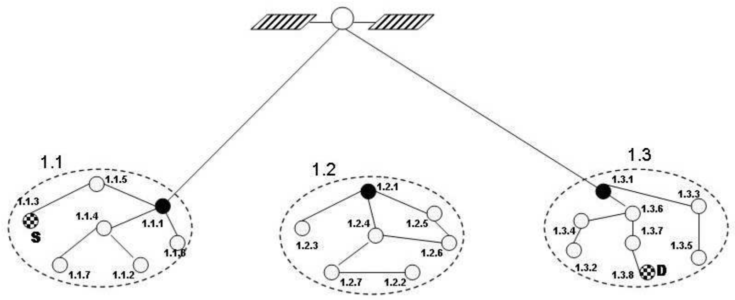


Figure 3.3: Hybrid network with three wireless clusters and satellite - lower layer

set of equations described in 3.3. In our model, this is the instantiation of the lower layer model as described in 3.2.2.3.

3.5.1.1 Notations

We first introduce notation and definitions that we use in deriving the set of implicit equations, where we use the fixed point algorithm. The network topology graph consisting of N nodes is given. Each node has the ability to transmit packets at the rate of Λ bits/second to the nodes that are connected to it, i.e., for simplicity, we assume that the physical layer capacity of all links is fixed and equal to Λ. The extension to different rates for every pair of adjacent links is straightforward; it only complicates notation and thus the main thrust of the argument may be hidden. All nodes use omni-directional antennas, and all neighbor nodes of the transmitting node receive the signal. The connectivity and interfering property between each node pair is decided by the Signal-to-Noise ratio (SNR) (transmission power, distance, modulation, etc). Note that two nodes are neighbors and connected if they can directly communicate, and two nodes are interfering if one of them cannot receive data, while the other one is transmitting data to a third node.

Let $c = 1, \cdots, C$ be the set of *commodities* in the network. Each commodity is specified by its source-destination pair $(I(c), O(c))$, and traffic demand rate r_c between them. Network links (connections between neighbor nodes) are specified either by their index $l = 1, \cdots, L$ or their source-destination pair (i, j). Note that r_c is the average traffic rate generated at the source node (the incoming rate), which may not be equal to the traffic received at the destination node (the outgoing rate) if there is packet loss in the network. We assume that η_l, the PHY layer link loss probability is known and fixed.

The routing is known and fixed and it is defined by the set of end-to-end paths (i.e. the paths between the origin and destination of each commodity) and the fraction (probability) of the incoming traffic that is transmitted on each of these paths. Let Π_c be the set of the paths that are used for commodity c. Consider a path $\pi_{c,k} \in \Pi_c$, then $\alpha_{\pi_{c,k}}$ is the fraction (probability) of commodity c traffic transmitted over path $\pi_{c,k}$ at $I(c)$, the source node of commodity c. We have

$$\sum_{\pi_{c,k} \in \Pi_c} \alpha_{\pi_{c,k}} = 1, \quad for \quad each \quad c = 1, \cdots, C \tag{3.17}$$

Our goal is to find a consistent set of link loss parameters and traffic rate parameters that satisfy two sets of equations. The first set is derived from the network loss model and computes the link outgoing traffic rates $\lambda_l,\ l = 1, \cdots, L$ from the MAC layer effective loss parameters $\epsilon_l,\ l = 1, \cdots, L$. The second set is based on Bianchi's model and computes the loss parameters ϵ_l from the rates λ_l. The fixed point method applies these two mappings iteratively, until convergence to a consistent solution $(\epsilon^\star, \lambda^\star)$ that satisfies both mappings is achieved. The existence of a consistent solution follows from the facts that both mappings are continuous and bounded and map a compact subset of R^{2L} into itself, via an application of a fixed point theorem [25]. We derive the two mappings in the following sections 3.5.1.2 and 3.5.1.3.

3.5.1.2 First Mapping: From ϵ to λ

We assume that the packet loss probabilities due to MAC layer contention are given. We compute the λ_l's, which are the *effective* (outgoing) data rates of the network links. Let $(l^\pi(1), \cdots, l^\pi(k_\pi))$ be the set of the links in path π, which are ordered from the first to the last hop in the path. Let s_π be the commodity that path π is serving. Then, $\lambda^\pi_{l^\pi(i)}$, the outgoing data rate of the i-th link of path π is,

$$\lambda^\pi_{l^\pi(i)} = r_{s_\pi} \alpha_\pi \prod_{j=1}^{i} (1 - \eta_{l^\pi(j)})(1 - \epsilon_{l^\pi(j)}). \tag{3.18}$$

Note that r_{s_π} is the corresponding source incoming traffic rate for commodity s_π, α_π is the fraction of that traffic routed on path π. The two terms in parentheses specify the percentage of the traffic that is successfully transmitted over the first i links of the path π. Let $E(l)$ be the set of paths that share link l. Then the total traffic rate on link l is,

$$\lambda_l = \sum_{\pi : \pi \in E(l)} \lambda^\pi_{l^\pi(j_\pi)} \tag{3.19}$$

The notation convention in the last equation is that there exist an index j_π for path π such that the link l whose total traffic rate we calculate is identical with the link $l^\pi(j_\pi)$ of path π. Equations (3.18), (3.19), taken together, provide the desired mapping from the vector of ϵ's to the vector of λ's. These computations are executed throughout the network synchronously.

3.5.1.3 Second Mapping: From λ to ϵ

This mapping is based on Bianchi's results [13], which we reviewed in section 3.3. Let H_i be the set of interfering nodes with node i. The set of nodes that interfere in the transmission from node i to j is the union of node i's and j's sets of interfering nodes. The total traffic exiting nodes in the set $\{H_i \bigcup H_j \bigcup\{i\} \bigcup\{j\}\}$ contends with each other and share the same multi-access channel. The total traffic demand for this channel is,

$$X_{ij} = \sum_{n=1}^{N} \sum_{m\in\{H_i \bigcup H_j \bigcup\{i\} \bigcup\{j\}\}} \lambda_{mn} \tag{3.20}$$

where N is the total number of nodes in the network.

For now assume that we have calculated S for this channel. Recall that S is the fraction of time that the channel is successfully used to transmit data packets. We assume that the contending links have equal capacity Λ. Therefore, the channel capacity is $S\Lambda$. We assume that the channel capacity is divided proportionally between all contending connections. Therefore, the link (i, j) effective data rate is,

$$u_{ij} = \begin{cases} \lambda_{ij} & \text{if } X_{ij} \leq S\Lambda \\ \\ \frac{S\Lambda}{X_{ij}}\lambda_{ij} & \text{if } X_{ij} > S\Lambda \end{cases} \tag{3.21}$$

The intermediate MAC layer loss factor for link (i, j) is,

$$\epsilon'(i,j) = \frac{\lambda_{ij} - u_{ij}}{\lambda_{ij}} \tag{3.22}$$

Assume now that we are at iteration $k+1$, the new value for MAC layer loss factor of link (i, j) is the weighted average of the previous iteration value and the intermediate value computed in (3.22):

$$\epsilon_{(i,j)}^{k+1} = \beta\epsilon_{(i,j)}^{k} + (1-\beta)\epsilon'(i,j) \tag{3.23}$$

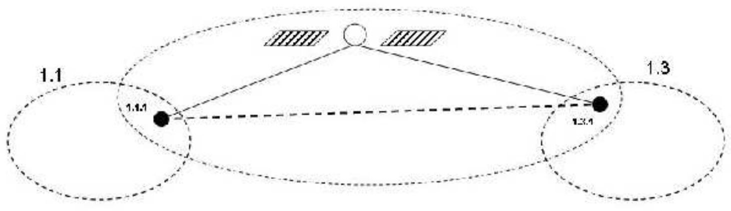


Figure 3.4: Hybrid network higher layer - source, destination and intermediate segments

where β is the weight, $0 \leq \beta \leq 1$. The weighted average is introduced to avoid rapid changes and oscillations in the computation of ϵ and λ in the fixed point iterations.

In summary, for each connection (link) in the network we have defined and computed the channel capacity. The channel capacity is based on the Bianchi's saturation model, and hence a function of the number of interfering nodes with the corresponding link. After convergence, the total effective (outgoing) data rate of each channel is less than its throughput. Equations (3.20), (3.21), (3.22), (3.23) taken together, provide the desired mapping from the vector of λ's to the vector of ϵ's. These computations are executed throughout the network synchronously.

3.5.2 Higher Layer Loss Model and Fixed Point Method

For the higher layer relaxation, we consider the overlay comprising the satellite and the two gateway nodes 1.1.1 and 1.3.1 logically connected to one another via satellite links. This form the intermediate segment at the higher layer, similar to section 3.2.2.4, and as shown in figure 3.4. We assume symmetric links for the satellite uplink and downlink.

In the higher layer, our goal is to find a consistent set of traffic rate parameters and satellite channel loss parameters that satisfy two sets of equations. The first set is derived from the loss model in the satellite layer and computes the outgoing traffic rate as a function of the channel loss and the incoming traffic rate. The second set is derived from the satellite channel capacity model described in section 3.4.2 and computes the satellite channel loss from the traffic rate. The fixed point method applies the two mappings iteratively until the convergence to a consistent solution that satisfies both set of equations. We subsequently feed the convergence values to the offered load in the destination cluster in the lower layer, namely to equation (3.19) and iterate local and higher layer computations till the difference between successive iterations are within certain criteria.

3.5.2.1 Mapping from Loss Parameter to Traffic Rate

The aggregate offered load (for class s of traffic) between the gateway nodes i and j in the overlay is:

$$\lambda^1_{ij_s} = \lambda^0_{ij_s} + \sum_{p \in P} \lambda^0_{ps}.(1 - \eta^{src}_p)(1 - \epsilon^{src}_p) \tag{3.24}$$

where p refers to any source-destination pair whose traffic passes through the link connecting the gateway nodes i, j while P is the total number of source-destination paths that pass through i, j. The first term on the RHS of equation (3.24) is the initial load between the gateway nodes, while the second term is the initial load for each source-destination pair whose path passes through gateway nodes i, j, thinned by physical layer losses η^{src}_p and MAC layer losses ϵ^{src}_p in the source cluster.

Let ϵ be the satellite link loss due to physical and MAC layer conditions, computed using the loss model described in section 3.4.1. Therefore, the reduced load of the i, j

overlay connection is:

$$\lambda_{ij_s}^{1}{}' = \lambda_{ij_s}^{1}(1-\epsilon)(1-\epsilon_{ij}^{'}) \tag{3.25}$$

where $\epsilon_{ij}^{'}$ is the loss due to data drop in the satellite link between i, j, computed from (3.28). In the initial run of the algorithm, we take its value to be zero.

3.5.2.2 Mapping from Traffic Rate to Loss Parameter

From equation (3.24), $\lambda_{ij_s}^{1}$ is the offered load on the connection between one pair of gateway nodes i, j. Let us express this as λ_k^1. Assuming there are L such connections between pairs of gateway nodes, the total traffic demand for the satellite channel is:

$$\lambda_L = \sum_{k=1}^{L} \lambda_k^1 \tag{3.26}$$

We assume that the multiple-access channel capacity C_a is divided proportionally between all L contending connections. Then the effective data rate on the connection between i and j is:

$$\nu_{ij_s}^{1} = \begin{cases} \lambda_{ij_s}^{1}{}' & \text{if } \lambda_L \leq C_a \\ \\ \frac{\lambda_{ij_s}^{1}{}'}{\lambda_L} C_a & \text{otherwise} \end{cases} \tag{3.27}$$

Therefore, the intermediate satellite link loss factor on the connection between i, j is:

$$\epsilon_{ij}^{'} = \frac{\lambda_{ij_s}^{1}{}' - \nu_{ij_s}^{1}}{\lambda_{ij_s}^{1}{}'} \tag{3.28}$$

In order to avoid rapid changes and oscillations in the computation of the fixed point, we update the value of the loss factor as the weighted average of its value in the previous iteration and the intermediate value computed in (3.28). Therefore the value of the loss factor at iteration $k+1$ is:

$$\epsilon_{ij}^{k+1} = \gamma \epsilon_{ij}^{k} + (1-\gamma)\epsilon_{ij}^{'} \tag{3.29}$$

where the value of the dampening factor γ, $0 \leq \gamma \leq 1$, is chosen such that the algorithm converges to a fixed point.

The fixed point values $\langle \epsilon^{\star}, \lambda^{\star} \rangle$ are fed back to the incoming channel traffic rate in (3.19) and the lower layer and higher layer computations are repeated till the difference in the throughput between successive iterations is within pre-determined limits.

Upon convergence of the fixed point iterations, denoting $\lambda_{first,p}$ and $\lambda_{last,p}$ to be respectively the arrival rate of packets of the source or destination of path p, we define the throughput of a source-destination pair c to be:

$$T_c = \frac{\sum\limits_{p \in P_c} \lambda_{last,p}}{\sum\limits_{p \in P_c} \lambda_{first,p}} \tag{3.30}$$

3.6 Network Design by Automatic Differentiation

The fixed point method provides a computational scheme that, when it converges (that is, at the fixed point), gives the value of the performance metric (for example, throughput) for certain values of the network design parameters (for example, routing parameters). We do not get any explicit analytical expression of the performance metric as a function of the network parameter. However, we are also interested in developing a methodology for network design by optimizing the network parameters - for example, finding the set of values of the routing parameters that optimize the network throughput.

This can be achieved from the set of equations used in the fixed point model by computing the sensitivities of the performance metrics with respect to the network parameters. Sensitivity analysis of the performance metrics can be done by the gradient projection method. The gradient projection method requires iterative computation of the throughput gradient. Since no explicit functional description exists for the performance metric, we need to rely on computational methods that numerically approximate the gradients. For this, we use Automatic Differentiation (AD).

AD is a numerical method to compute the derivatives of a program [17]. Using the fact that a computer program is in fact a sequence of primary operations, automatic differentiation records the relationships between them and using the chain rule, it is able to provide the derivative of a function in a short amount of time [19].

In this section, we present the methodology employed to optimize the overall throughput in the network by changing the path probability distribution of each connection on the network. The description of the methodology is from [19].

We denote by P_c the set of paths used in connection c and by C the set of all active connections in the network. The total network throughput T is:

$$T = \frac{\sum\limits_{c \in C}(\sum\limits_{p \in P_c} \lambda_{last,p})}{\sum\limits_{c \in C}(\sum\limits_{p \in P_c} \lambda_{first,p})} \tag{3.31}$$

Then, assuming that there are $m = |C|$ active connections in the network, n_c paths used in the connection c and denoting by $\pi_{i,c}$ the probability associated with using path i in connection c, the total throughput is a function of these input probabilities, namely:

$$T = T(\pi_{1,c_1}, \cdots, \pi_{n_{c_1},c_1}, \cdots, \pi_{n_{c_m},c_m}) \tag{3.32}$$

Thus, we can write our optimization problem in the following way:

$$\begin{aligned} & max\, T = T(\pi_{1,c_1}, \cdots, \pi_{n_{c_1},c_1}, \cdots, \pi_{n_{c_m},c_m}) \\ \text{subject to} \quad & \sum_{i \in P_c} \pi_{i,c} = 1, \forall c \in C \\ & \pi_{i,c} \geqslant 0, \forall (i,c) \in P_c \times C \end{aligned} \tag{3.33}$$

This optimization problem is solved by gradient projection. Let $\overline{\nabla}_c$ be the average gradient obtained for connection c, the value is subtracted from each of the gradients obtained for the paths in P_c to ensure that the constraint $\sum_{i \in P_c} \pi_{i,c} = 1$ is met. In other words, at each iteration the set of routing probabilities is updated using the following formula:

$$\pi_{i,c_k} = max\ (0, \pi_{i,c_k} + \beta(\frac{\delta T}{\delta \pi_{i,c_k}} - \overline{\nabla}_{c_k})), \forall k \in \{0, \cdots, m\} \tag{3.34}$$

β is a parameter used to control the size of the steps taken during the update process of each iteration so that none of the parameters become negative. This iteration is continued until for each connection, every path with non-zero probability of being used has equal gradient. Namely, the iteration stops when:

$$\forall \pi_{i,c} \neq 0 \in P_c, \frac{\delta T}{\delta \pi_{i,c}} = \overline{\nabla}_c, \forall c \in C \tag{3.35}$$

Once the algorithm converges, we output a new set of results for the network configuration containing: a) the optimized throughput of each active connection in the network and b) the set of routing probabilities for each connection needed in order to achieve such throughput.

In order to compute the gradient values, we use the ADIC package [26] that is a source translator augmenting ANSI-C programs with statements for the computation of derivatives using the AD method. In our case, the input to ADIC is the fixed point algorithm that we developed in ANSI C code. ADIC generates a new version of the program that computes both the original result, which is throughput, and its derivatives with respect to the input parameters, which are the path routing probabilities. We then use the computed gradients to find the optimal routing parameters which maximize the throughput.

We have implemented the Dreyfus K-shortest path algorithm [27] for path selection. For a given set of link weights and integer value k and source-destination pair, this algorithm finds k loop free paths with minimum total weight. We set all link weights to one, but it is possible to use other weights based on the distance, bandwidth, interference or other performance related criteria.

3.7 Numerical Results

We have implemented the loss network model for the hybrid satellite network in C code alongwith the proposed fixed point approximation of the loss network model to derive an estimate of the network performance.

For our simulations, we built a simple two-cluster hierarchical network, presented in figure 3.5. The network has five connections across the two clusters: (i) from node 1 to node 20, (ii) from node 2 to node 32, (iii) from node 3 to node 16, (iv) from node 6 to node 30 and (v) from node 4 to node 48. The communication between the two clusters is only over the satellite link, through the gateway nodes - node 4 in cluster 1 and node 15 in cluster 2. Therefore, all the paths for the five connections go through the two gateway

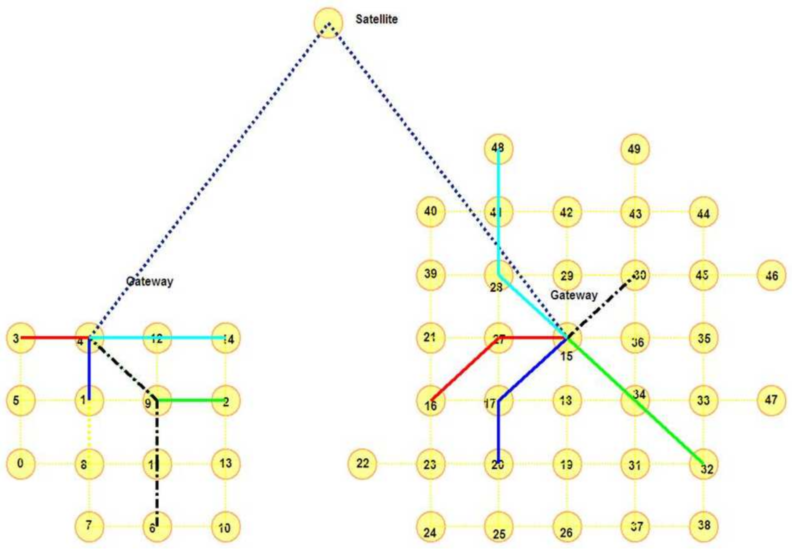


Figure 3.5: Simulation setup: hybrid network topology with 5 connections across 2 clusters.

nodes. We assume that the physical layer loss parameters are fixed and equal to 0.001. The MAC layer loss parameters depend on the link data rates and we use the proposed fixed point method to compute them. For the MAC layer, we use the IEEE 802.11 FHSS with 1 Mbps capacity for each link. For the satellite link, the per-gateway capacity is allocated at 266kbps according to SAMA, while the combined channel error rate is computed to be 0.001080, using the model and the numerical figures described in section 3.4.1. The base user traffic between the two gateway nodes is set at 128kbps, while the traffic on each of the connections considered in the simulations is varied between 100kbps and 550kbps for different scenarios. The number of paths allowed per connection is also varied for different simulation scenarios, between 1 and 5.

We have also conducted discrete event simulations on an identical network topology in Opnet Modeler [3], to establish a point of reference for the fixed point model results,

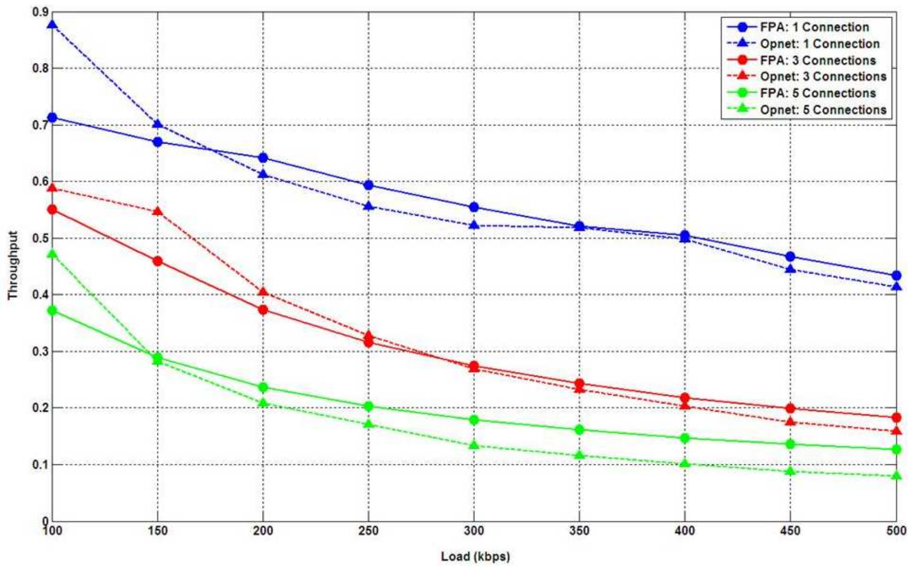


Figure 3.6: Throughput comparison between fixed point method and Opnet discrete event simulation.

and for comparison.

The first experiment compares the variation of the throughput computed by our fixed point method according to the desired load in the network with the same metric estimated by the Opnet simulations. Our routing algorithm finds the shortest paths between the source and destination nodes having nodes involved in different connections. Using these paths we then employ our set of fixed point equations to compute the throughput of these connections according to the desired load. As can be seen in figure 3.6 the fixed point model results are close to the Opnet results. The Opnet results in figure 3.6 thus validate the trend of the curves obtained by our model. The fact that our computations slightly overestimate the throughput as predicted by OPNET can be justified by the approximations we make when computing losses.

Another metric of interest is the time taken for the computation of throughput in

Number of Connections	1	3	5	7	9	11
Fixed Point Method	0.089	0.124	0.702	0.907	1.103	1.295
Opnet Simulation	229	218	228	242	265	265

Table 3.1: Comparison of simulation runtime between fixed point method and Opnet (in seconds).

both cases. Our fixed point method converges really fast - on a Intel Centrino 1.66GHz dual-core computer with 2GB of memory, it takes on the order of seconds for our model to compute the throughput achieved in 10 steps between 100kbps and 500kbps, while corresponding Opnet simulations on the same machine take on the order of several minutes. Table 3.1 compares the time needed by our model and by Opnet as a function of the number of active connections in the network. Therefore the main advantage of our fixed point model over discrete event simulation platforms such as Opnet, is the computation time. This makes our model more suitable to compute approximations of throughput for network management and design which require fast and/or multiple simulations.

We have also run simulations of the fixed point method with AD to enhance the routing performance. Here we assume that a fixed set of paths are given and we want to tune the probabilities (portions) of sending traffic over the paths to maximize the throughput. We consider three inter-cluster connections in the network topology of figure 5 - nodes 1 to 20, nodes 2 to 32 and nodes 3 to 16. We consider three alternative routing strategies: (1) using shortest path only, (2) using all available paths with equal probability, and (3) using AD and gradient projection method to find the optimal probabilities.

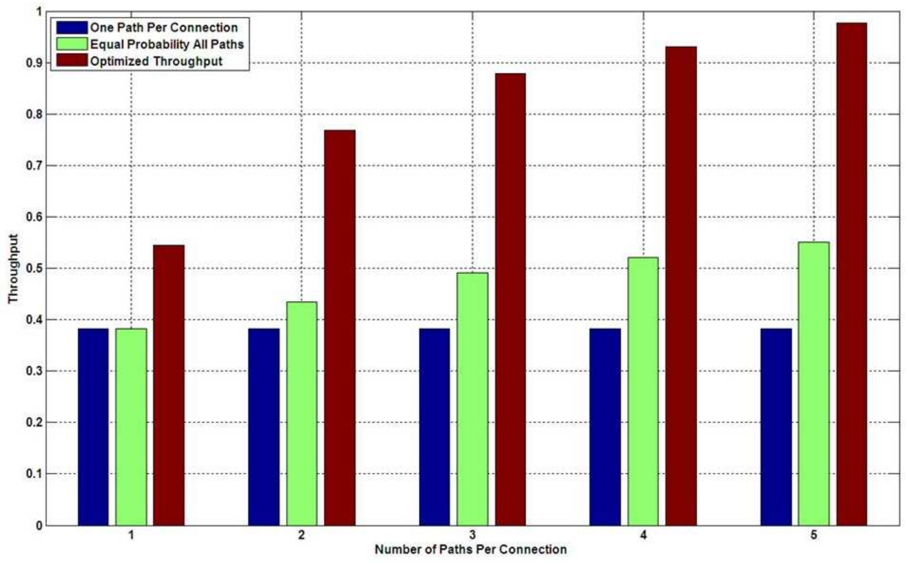


Figure 3.7: Comparison of aggregate network throughput for number of available paths, 500kbps input load.

Figure 3.7 shows the aggregate network throughput for the three connections, versus the number of available paths. The performance of the optimization algorithm improves as the number of available paths increases and it clearly outperforms other policies. If there are more paths available, the optimization algorithm will use them to find the best traffic allocation and thus achieve the highest total throughput.

3.8 Summary

In this chapter, we have described a new method for performance analysis of hybrid networks, by combining loss network models for the MAC and PHY layers, with routing, through fixed point iterations. We have split the network into two levels based on the node and link characteristics, by taking into consideration the natural hierarchy present in the hybrid architecture. We have applied the technique of hierarchical reduced loss network model, adapted for packet-switched networks, to create a reduced load network model for the hybrid network that connects the packet arrival rates in the different levels with the associated physical and link loss rates. For the lower level of terrestrial wireless nodes, we have used Bianchi's 802.11 throughput model and designed a simple throughput loss model that couples the physical, MAC and routing layers. The model provides quantitative statistical relations between the loss parameters used to characterize multiuser interference and physical path conditions on the one hand and traffic rates between origin-destination pairs on the other. For the higher layer consisting of the terrestrial gateway nodes interconnected by the satellite overlay, we have used the Gilbert-Elliot bit error loss model coupled with Slotted Aloha Mutliple Access protocol, to derive a similar relationship between traffic arrival rate and the link loss. We then apply a fixed point approximation approach on each set of relations for the lower layer to derive a solution that converges to a fixed point for the set of parameters arrival rate and link layer loss, while satisfying all the equations in the set. We feed the throughput arrived at by the fixed point method in the lower layer, to the total throughput for the higher layer and perform the fixed point iterations on the higher layer to derive a solution that converges to a fixed point for the parameter set arrival rate and link loss. The higher layer throughput obtained as a result is fed back to the lower layer and the fixed point method is repeated on both layers till the difference in throughput in each layer, between successive iterations, is within pre-determined limits. We thus obtain an implicit model of the selected performance metric, parameterized by the design variables. We use the fixed point method on the loss network model with Automatic Differentiation to compute sensitivities of the performance metrics with respect to design parameters. We thus have a tool for fine-tuning the network parameters to create an optimal network topology. We have illustrated the proposed analysis and design models through the simulations. Comparing the results obtained from the fixed point method to Opnet discrete event simulations, we have shown that our method can compute the network throughput to a great degree of accuracy, while taking significantly less time.

3.9 Related Work

Loss network models [28] were originally used to compute blocking probabilities in circuit switched networks [29] and later were extended to model and design ATM networks [30, 31, 32, 33]. In [33] Liu and Baras have used reduced load approximations to evaluate quite complex ATM networks, with complex and adaptive routing protocols, and multi-service multi-rate traffic (different service requirements). The authors in [33] also describe using Automatic Differentiation for network design and have shown that the method allows for complex network design parameters to be implicitly embedded in the input function to the AD module.

In [34], Dai and Chan propose a unified mathematical framework using a two-stage stochastic programming formulation, to find the most cost-effective network topology for hybrid broadband terrestrial/satellite networks. The solution to the problem formulation gives optimal link capacities and optimal routing strategy for different network topologies. The authors show that for certain parameter values, the hybrid topology is more cost effective than non-hybrid topologies.

Chapter 4

Efficient Source Authentication in Hybrid Satellite/Wireless Networks

4.1 Overview

Security is a necessary parameter in hybrid wireless networks if the communication between a pair of nodes, or a group of nodes, is to be protected from unauthorized access. Due to the open nature of the wireless channel, intruders can eavesdrop on the communication between other nodes if the messages are sent in the clear; they can inject fake messages into the network, purporting to come from other nodes, or attempt to modify or delete messages between other nodes. Therefore, strong security mechanisms to prevent such attacks are important, especially for scenarios like military operations where hybrid networks can be of great use. Security of communication can be achieved using several different mechanisms. Encryption hides the messages in ciphertext and thus prevents eavesdropping on the communication. In the process of authentication, each message is "stamped" with a unique "marker" of the originating node which ensure that messages are accepted from legitimate nodes only, and fake messages are discarded. Associated with authentication are message integrity protocols where each message is similarly stamped with a unique marker by the originating node so that any unauthorized modification in transit invalidates the marker and thus the modification can be easily detected.

The thrust of our research effort is to enable communications and ensure that messages between communicating nodes are correctly delivered. We are not so much concerned with ensuring the confidentiality of the communications. We therefore focus on user authentication and associated message integrity protocols. These security mechanisms are required to prevent attacks against the network protocols and ensure their correct and robust operation.

For unicast communications, solutions for authentication and message integrity are trivial - the two parties communicating nodes A and B share a secret exclusively between themselves and make use of this secret, or a key derived thereof, to “sign” the messages between themselves. Since no other node knows the secret, A can be assured that the message originated at B, and vice versa. The secret or key used can be based on symmetric cryptography that is fast, efficient, and does not consume significant computation or energy resources at the communicating nodes. The corresponding message signature is usually a *Message Authentication Code*, or MAC in short (for example, HMAC [35]), which is resource-efficient to compute and to verify, and limited in size.

The problem is more complicated for group communications. When multiple parties are taking part in a communications session, a shared secret between the parties is not a solution. Since everyone knows the same secret, it is impossible for the receivers to know for sure whether the message originated at the alleged source, or was it spoofed by another node in the group that knows the secret. At best, a shared secret in this setting can assure the involved nodes that the message originated from someone within the group (assuming the secret has not been leaked to outsiders). It is also possible that all the nodes do not share secret beforehand, for example when a group of nodes with no prior knowledge of one another take part in a communications session, and erase the security information once the session terminates (this is true even for two nodes with no prior history together).

In this situation authentication is done based on asymmetric techniques where each node possesses a unique secret known to no other node, and makes use of that secret to authenticate itself, or the messages it generates. Public key cryptography allows such asymmetric authentication to take place. In public-key cryptography, each source uses its private key to sign messages it generates, creating a digital signature that is appended to the message [36]. The receivers can verify the signature using the corresponding public key of the node, which is known to everyone from the source's certificate. The primary requirement is that all users have access to a common third party node called the Certificate Authority (CA) that is universally trusted. The CA is responsible for binding a node's identity to its public key in the node's public-key certificate - for example, PGP [37] and X.509 [38], which are the two most commonly used certificate formats. The certificate can be freely distributed to all nodes in a network, and the correctness of the certificate is verifiable by any node that has access to the CA. Apart from facilitating secure authentication of nodes and message integrity checks, public-key cryptography also provides *non-repudiation* - a node cannot deny later that it generated a message that has been signed using its private key.

Public-key cryptography is a powerful tool that facilitates authentication, message integrity and also data encryption. However, it is computationally very expensive (both in CPU cycles and energy expenditure) to generate digital signatures for messages, and also to verify them [39, 40, 41, 42]. The public and private keys are larger in size compared to symmetric keys, and the certificates also take up considerable storage space. In wireless networks where many of the nodes might have resource constraints, public-key cryptography can be a severe burden. For example, handheld devices have limited processor power, storage capacity and available energy. Performing digital signature generation and verification frequently can consume significant processor capacity and drain the battery quickly. Therefore in hybrid wireless networks it is preferable to use authentication protocols that are based on symmetric cryptographic primitives - being efficient in terms of processing load, symmetric operations would expend less node energy. However, designing authentication protocols for group communications using symmetric cryptography is a significant challenge. The primary difficulty is how to create the asymmetry such that each participant has a unique secret with which to authenticate its messages, while allowing all the receivers the capability for validation. (Here we assume that the security association between each source and the group of receivers is generated on-the-fly, and does not make use of pre-shared secrets between every pair of nodes, which is the trivial solution that does not scale well.)

Our objective, therefore, is to design an asymmetric user authentication protocol for group communications for resource-constrained devices. We propose a source authentication algorithm that uses symmetric cryptographic primitives to achieve asymmetric authentication of nodes in group communications, and also message integrity. The protocol can be efficiently implemented in a hybrid wireless network by taking advantage of the presence of the satellite overlay. The protocol considers the resource limitations of the wireless nodes and the wireless characteristics of the terrestrial segment. It avoids the assumption that the user nodes have some sort of security association established apriori, as many other protocols assume.

4.2 Methodology for Source Authentication in Group Communications

We propose to achieve authentication using a new class of certificates called *TESLA Certificate*. TESLA certificates are based on a symmetric cryptographic primitive - MAC computation using keyed hash functions - and use delayed disclosure of the key by the CA, to achieve the asymmetry required for authentication in group communications. Due to the use of MACs to generate and verify certificates, the scheme is very fast, has low processing overhead, and consumes much less energy that public key signature generation/verification. Therefore the scheme is well-suited to terrestrial wireless nodes with limited resources. The size overhead due to MAC addition to messages is lower compared to digital signatures, and therefore consumes less bandwidth for sending the control messages that are required for secure multicast routing.

The TESLA certificate concept was originally proposed in [43], and we have suggested modifications and extensions to it in [44]. We use the modified TESLA certificate design in proposing a mechanism for authentication and message integrity, using the satellite as the CA.

4.2.1 Review of TESLA Authentication Protocol

The TESLA broadcast authentication protocol [45, 46] represents a fundamental paradigm shift in source authentication in a group setting. TESLA achieves asymmetric authentication between a source and receivers through the use of symmetric cryptographic MAC functions. The asymmetry is obtained through the *delayed disclosure* of the authentication keys. We give a brief description of TESLA in the following paragraphs.

$$\xleftarrow{\text{Sequence of key generation}}$$

$$\begin{tikzcd}
s_0 \arrow[d, "F_2"] & s_1 \arrow[l, "F_1"'] \arrow[d, "F_2"] & \ldots \arrow[l, "F_1"'] & s_{n-1} \arrow[l, "F_1"'] \arrow[d, "F_2"] & s_n \arrow[l, "F_1"'] \arrow[d, "F_2"] \\
 & & \ldots & & \\
K_0 & K_1 & \ldots & K_{n-1} & K_n
\end{tikzcd}$$

$$\xrightarrow{\text{Sequence of key usage}}$$

$$\xrightarrow{\text{Time}}$$

Figure 4.1: TESLA key generation

TESLA divides the time of transmission by the source into n intervals of equal duration. The source generates a random key seed s_n for interval n, and computes a one-way hash chain by repeatedly applying a one-way function F_1 to s_n. The number of elements of the hash chain correspond to the number of intervals that the source transmits. The source computes the MAC computation key for each time interval by applying a second one-way function F_2 to each element of the hash chain. The functions F_1, F_2 are publicly-available and known to all the receivers. The algorithm is illustrated in fig. 4.1.

The sender uses the keys in the reverse order of their generation, that is, starting with K_1 in interval 1, followed by K_2 in interval 2, and so on. Owing to the one-way property of F_1 and F_2, it is computationally infeasible for any node to generate s_i knowing K_i, or to generate s_{i+1} knowing s_i. The sender bootstraps the hash chain by broadcasting to all the receivers the anchor element of the chain, for example s_0, signed with its private key (in case of public-key based bootstrapping), or by encrypting s_0 with the secret key it shares with each receiver in the network (for symmetric-key based bootstrapping).

For each packet generated in time slot i, the source uses the authentication key K_i

to compute a MAC on the packet. The MAC is then appended to the packet, which is transmitted to the receiver(s). When a node receives a packet, it first checks whether the packet is *fresh*, that is, it was sent in a time interval whose corresponding TESLA key has not been disclosed. This is the fundamental security criterion in TESLA. Each receiver discards any packet that does not meet the security criterion, and buffers only the packets that satisfy the freshness condition. The receiver cannot authenticate the packets immediately since it does not know the corresponding key K_i. The sender discloses the key K_i at a later instant in time by broadcasting the corresponding key seed s_i. Upon receiving s_i, each receiver first verifies the authenticity of s_i by checking $s_i \xrightarrow{F_1} s_{i-1}$ (and therefore ultimately verifying against the anchor element s_0 which has already been authenticated). If s_i verifies correctly, each receiver can compute K_i: $s_i \xrightarrow{F_2} K_i$ and subsequently use the computed K_i to verify the MAC on the packets received during interval i.

Once s_i is disclosed, any node with knowledge of s_i can compute K_i and attempt to masquerade as the sender by forging MACs using K_i. Therefore, K_i is used to compute MACs on packets generated only during the interval i, other time intervals use different keys to compute the MACs. The key seed s_i is disclosed only d time slots after i so that no malicious node can compute K_i and forge packets in the intervening period. d is computed based on the maximum network delay from the source to all the receivers. This is the principle of delayed disclosure of keys.

The major advantage of TESLA in this regard is that it allows similar authentication through the use of computationally efficient MAC functions, and is therefore very attractive for authentication in devices of limited capabilities.

The above is a basic description of TESLA. The algorithm has several enhancements to mitigate various drawbacks, they are described in [46].

4.2.2 Review of TESLA Certificate Algorithm

The idea of certificates based on TESLA was proposed in [45]. The idea has been formalized to form a TESLA-based public key infrastructure (PKI) in [43].

In the algorithm described in [43], there is a certificate authority CA who creates certificates for an entity B. During time slot n, the CA generates authentication key aK_{B_n} for B to use to compute the MAC on its messages in that interval. The CA creates a certificate $Cert_{CA_n}(B)$ to bind aK_{B_n} to B for interval n. The CA uses its TESLA key tK_{CA_n} to encrypt aK_{B_n} in the certificate, and uses the same key to compute a MAC on the certificate:

$$Cert_{CA_n}(B) = (ID_B, \{aK_{B_n}\}_{tK_{CA_n}}, n+d, MAC_{tK_{CA_n}}(..)) \tag{4.1}$$

aK_{B_n} is known only to the CA and B during period n, while tK_{CA_n} is known only to the CA. $n+d$ indicates the time at which the CA will disclose tK_{CA_n} to the nodes, that is, it is the expiration time of the certificate. The CA sends $Cert_{CA_n}(B)$ to B alongwith aK_{B_n}, which is encrypted with key $K_{CA,B}$ that is shared between the CA and B.

In the time interval $\langle n, n+d\rangle$, a low-powered device D sends a request to B for using B's service: $D \rightarrow B\colon (request)$. To authenticate itself to D, B sends an authentication packet containing its certificate and a MAC on the request:

$$B \rightarrow D\colon (Cert_{CA_n}(B), MAC_{aK_{B_n}}(request)) \tag{4.2}$$

When D receives the authentication message, it checks the timestamp of $Cert_{CA_n}(B)$ to make sure it has arrived before time $n + d$. If the certificate is "fresh", D buffers the authentication packet. At time $n + d$, the CA discloses tK_{CA_n}. Upon receiving the key, D verifies $Cert_{CA_n}(B)$ by checking the MAC in the certificate using tK_{CA_n}. If the MAC verifies correctly, D obtains aK_{B_n} from the certificate by decrypting with tK_{CA_n}. Subsequently, D checks $MAC_{aK_{B_n}}(request)$ to verify the authenticity of B. Therefore, D is able to verify the identity of B only if it receives $Cert_{CA_n}(B)$ before $n + d$. Once the CA discloses its TESLA key tK_{CA_n}, any node could forge a certificate for the time interval n.

The TESLA certificate algorithm described above allows a node to add authentication to packets for a *single* period in time. Therefore, a source node B that transmits for multiple time intervals will need several TESLA certificates from the CA. If there are many sources that send data over long intervals, this can add up to a substantial overhead.

4.2.3 Extending TESLA Certificates: Protocol Assumptions

We propose a new algorithm for source authentication by modifying the initial concept of TESLA certificates, which we described briefly in section 4.2.2. Our proposal includes the folowing modifications:

- we add mechanisms to extend the lifetime of the TESLA certificate from single use to multiple uses, and
- we allow disclosure of source TESLA keys via proxy.

The TESLA certificate implementation requires the presence of a dedicated Certificate Authority (CA) to generate the certificates. In our algorithm, the CA broadcasts the TESLA keys of the source nodes to the network at periodic key disclosure intervals. In the hybrid network topology, we use the satellite for providing CA services. The reasons for using the satellite as the CA are as follows. The satellite is a network node that is always available, connected to the entire network, and is physically secure. The satellite has higher computing power with on-board processing capability and higher storage compared to terrestrial wireless nodes. Its energy is technically infinite, since it is renewable via solar power. Therefore the satellite can perform processing-intensive cryptographic operations more efficiently compared to the terrestrial nodes. The presence of the satellite thus allows implementation of highly efficient and secure centralized authentication protocols that would have been difficult to implement in terrestrial wireless networks without the centralized satellite infrastructure. We thus consider the satellite as the root CA in our authentication protocol design and assume that the satellite is trusted by all other nodes in the network. In our authentication protocol, the satellite generates the TESLA certificates for all the terrestrial user nodes, and it acts as the proxy for the terrestrial nodes for disclosing the TESLA MAC keys used by the nodes for authentication and message integrity - instead of the source node, the satellite broadcasts the TESLA keys at regular time intervals. Therefore the TESLA keys reach all the user nodes in one broadcast transmission. This saves the delay in TESLA authentication, and reduces the processing load on the source nodes, and also the network transmission overhead. We describe the details of our source authentication protocol in the following sections.

In order to describe the operation of the authentication protocol, let us consider a group of three wireless nodes A, B and C, where A sends messages to B and C. Our objective is to design an authentication mechanism that allows B and C to securely authenticate messages from A using a computationally efficient algorithm that expends low node energy. We make the following assumptions about the initial security setup of the network for authentication purposes:

- all three nodes have limited energy and processing power, and none has any pre-existing security information about the others;
- the public key $+K_{CA}$ of the CA is available to all nodes;
- all nodes are time-synchronized with the CA;
- appropriate security policies are in place to allow each node to securely identify itself to the CA during the initial bootstrapping phase, and each node X shares a unique secret key $K_{CA,X}$ with the CA;
- one-way functions F_1 and F_2 [47] are publicly available;
- message transmission from A to B and C start at time t_0;
- time is divided into intervals, each of duration Δ.

4.2.4 Initial Setup: Key Generation by CA and Source Node

During the initial setup, before any messages are transmitted in the network, the CA and all sources generate the keys that each will need for message authentication. The sets of keys are generated using the TESLA algorithm.

The CA uses a TESLA key chain $\{tK_{CA,i}\}$, $i = \{1,..,N\}$ to authenticate the TESLA certificates that it generates for the group sources. The CA generates a random seed $s_{CA,N}$ and applies one-way function F_1 to $s_{CA,N}$ to form a *hash chain* (4.3):

$$s_{CA,0} \xleftarrow{F_1} s_{CA,1} \xleftarrow{F_1} \ldots \xleftarrow{F_1} s_{CA,N-1} \xleftarrow{F_1} s_{CA,N} \tag{4.3}$$

where $N > 0$ is equal to the number of unique MAC keys that the CA expects to use for authenticating the certificates and messages it generates. The value N depends on the length of each time interval and the total duration that the CA node will perform the function of the CA. We assume that in each time interval, the CA uses only one key for computing the MACs on all the messages it generates in that time interval Therefore, if the total time of CA's functionality is T and the interval for key disclosure it d, we have $N = \frac{T}{d}$.

Subsequently the CA applies function F_2 to each element of the hash chain to obtain the certificate keys $tK_{CA,i}$ that it uses in the certificates (4.4):

$$\begin{tikzcd} s_{CA,0} & s_{CA,1} \arrow[l, "F_1"'] \arrow[d, "F_2"] & \ldots \arrow[l, "F_1"'] & s_{CA,N-1} \arrow[l, "F_1"'] \arrow[d, "F_2"] & s_{CA,N} \arrow[l, "F_1"'] \arrow[d, "F_2"] \\ & & \ldots & & \\ & tK_{CA,1} & \ldots & tK_{CA,N-1} & tK_{CA,N} \end{tikzcd} \tag{4.4}$$

$s_{CA,0}$ is the *anchor element* of the CA's authentication key chain. All TESLA certificates and signed messages from the CA are authenticated using the anchor element during the protocol run. $s_{CA,0}$ is broadcast to the network at time $t < t_0$ (4.5):

$$CA \rightarrow network\colon\ (s_{CA,0}, SIGN_{-K_{CA}}(..)) \tag{4.5}$$

The anchor element itself is authenticated using traditional public-key cryptography: the CA generates a signature on the message containing the anchor element and broadcast the message with the signature. All network nodes receiving the broadcast message verify the signature on the message using the public key $+K_{CA}$ of the CA. If the signature is verified, the nodes store in local memory the key $s_{CA,0}$ along with the broadcast message.

In a manner similar to the above, each source node A generates a random seed $s_{A,n}$ and applies one-way function F_1 to $s_{A,n}$ to form a *hash chain*, before any messages are sent. A subsequently applies F_2 to each key $s_{A,i}$ generated above and obtains the output $s'_{A,i}$ (4.3).

$$\begin{tikzcd}
s_{A,0} \arrow[d, "F_2"] & s_{A,1} \arrow[l, "F_1"'] \arrow[d, "F_2"] & \dots \arrow[l, "F_1"'] & s_{A,n-1} \arrow[l, "F_1"'] \arrow[d, "F_2"] & s_{A,n} \arrow[l, "F_1"'] \arrow[d, "F_2"] \\
s'_{A,0} & s'_{A,1} & \dots & s'_{A,n-1} & s'_{A,n}
\end{tikzcd} \tag{4.6}$$

Here $n > 0$ is equal to the number of unique MAC keys that A expects to use for authenticating its messages. The value n depends on the length of each time interval and the total duration of A's transmission. We assume that in each time interval Δ, a source uses only one key for computing the MACs on all the messages it generates in that time interval Therefore, if the total time of A's transmission is T, we have $n = \frac{T}{\Delta}$.

At time $t < t_0$, A sends $s_{A,n}$, n to the CA, along with details on A's key disclosure interval. The message from A to the CA is secured using the shared secret $K_{CA,A}$ between A and the CA. The CA can obtain all the elements of A's TESLA key chain from $s_{A,n}$ and n, as in equation (4.3).

On successful verification of A's identity, the CA generates the TESLA certificate for A. The key $s_{A,0}$ is included in the certificate as the anchor element of A's key chain. It is encrypted using key $tK_{CA,1}$ from the CA's key chain. The certificate also includes the identity of the source node A and the time $t_0 + d$ upto which the certificate is valid, i.e., after time $t_0 + d$, key $s_{A,0}$ is made public to the group and it can no longer be used

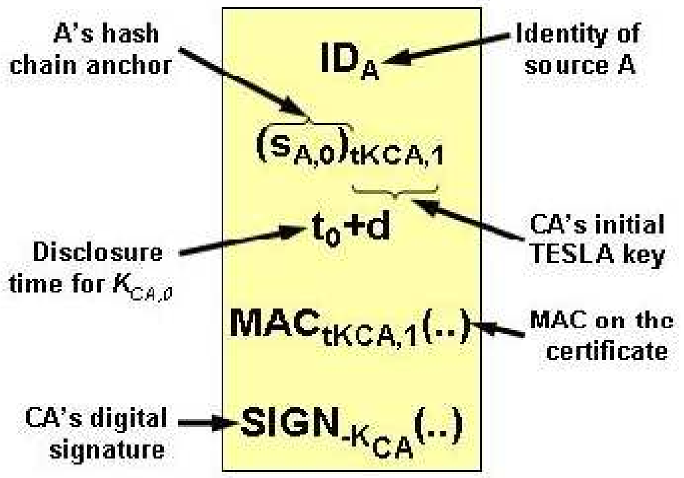


Figure 4.2: TESLA Certificate for node A

for new messages. The certificate also contains a MAC for authentication, computed on the previous elements using $tK_{CA,1}$. For added security, the certificate might also contain CA's public-key signature on all the previous elements (4.7).

$$Cert_{CA}(A) = \left(ID_A, \{s_{A,0}\}_{tK_{CA,1}}, t_0 + d, MAC_{tK_{CA,1}}(..), SIGN_{-K_{CA}}(..)\right) \quad (4.7)$$

$$CA \rightarrow A : Cert_{CA}(A) \quad (4.8)$$

Here $d \geq \Delta$ is the key disclosure delay for the CA TESLA signature key, and $tK_{CA,1}$ is the CA MAC key for the time period $\langle t_0, t_0 + d \rangle$. A schematic of the TESLA certificate for A is given in figure 4.2.

4.2.5 Message Transmission from Source to Receiver

A sends messages to B and C starting in the time interval $\langle t_0, t_0 + d \rangle$. A computes a MAC over the message m_0 using $s'_{A,0}$ and includes its TESLA certificate $Cert_{CA}(A)$

with the message.

$$A \to \{B, C\}\colon \{M_0 | M_0\colon \Big(m_0, MAC_{s'_{A,0}}(m_0), Cert_{CA}(A)\Big)\} \tag{4.9}$$

Each of B and C checks the *freshness* of the certificate by checking the timestamp of $Cert_{CA}(A)$ to make sure it has arrived within the period $\langle t, t_0 + d \rangle$. The receivers also check that $s'_{A,0}$ is not publicly known, i.e., $MAC_{s'_{A,0}}(m_0)$ cannot yet be computed by them. If all the checks pass, B and C store M_0 in their respective buffers, else they discard the message.

Checking the timestamp on $Cert_{CA}(A)$ is critical for the security of the algorithm. Once the CA discloses $s_{CA,1}$ at time $t_1 \gtrsim t_0 + d$, any node in the network can create a fake certificate with timestamp $t_0 + d$, allegedly generated by the CA, similar to (4.7). Therefore receivers will only accept certificates for which the CA TESLA key has not been disclosed at the time of receiving the certificate.

4.2.6 Message Authentication at Receiver

At time $t_1 = t_0 + d$, the CA broadcasts the key $s_{CA,1}$ to the network:

$$CA \to network\colon \left(\langle t_0, t_0 + d \rangle, s_{CA,1}, SIGN_{-K_{CA}}(..)\right) \tag{4.10}$$

If receiver B or C has received the anchor element $s_{CA,0}$ (4.5), they can check the authenticity of $s_{CA,1}$ by verifying $s_{CA,1}$ against $s_{CA,0}$:

$$s_{CA,1} \xrightarrow{F_1} s_{CA,0} \tag{4.11}$$

Otherwise, B or C can verify $s_{CA,1}$ from the signature using $+K_{CA}$. If verification is successful, each receiver derives $tK_{CA,1}$ from $s_{CA,1}$ (4.4) and uses $tK_{CA,1}$ to verify the

MAC on $Cert_{CA}(A)$. If the MAC is correct, receiver B obtains $s_{A,0}$ from $Cert_{CA}(A)$ by decrypting with $tK_{CA,1}$. B obtains $s'_{A,0}$ from $s_{A,0}$ (4.6). Then B checks $MAC_{s'_{A,0}}(m_0)$ using $s'_{A,0}$ and accepts m_0 if the MAC verifies correctly. B saves $Cert_{CA}(A)$ and the anchor element $s_{A,0}$ of A's key chain in long-term memory - they are used for authenticating future keys and messages from A.

Messages from A to B in subsequent time intervals use the corresponding key of A's key chain to compute the MAC. A does not have to include its TESLA certificate in messages subsequent to M_0, under the assumption that every receiver has received M_0 correctly. For example, in the period $\langle t_i, t_i + \Delta \rangle$, message M_i from A to B would look like:

$$A \rightarrow B\colon \{M_i | M_i\colon \Big(m_i, MAC_{s'_{A,i}}(m_i)\Big)\} \tag{4.12}$$

At time $t_i + d$, *the CA broadcasts* $s_{A,i}$ *to the network*. Since $d > \Delta$, when $s_{A,i}$ is disclosed, A is no longer using $s'_{A,i}$ for computing the MACs on its messages. Any receiver B that receives the CA broadcast, verifies that $s_{A,i}$ indeed belongs to A's MAC key chain as:

$$s_{A,i} \xrightarrow{F_1} s_{A,i-1} \xrightarrow{F_1} \ldots \xrightarrow{F_1} s_{A,0} \tag{4.13}$$

The above verification is correct since F_1 is a secure one-way function and $s_{A,0}$ has already been verified from $Cert_{CA}(A)$. However, if B wants to be additionally careful, it can verify $s_{A,i}$ going through the additional steps described above, using the CA key broadcast message and $Cert_{CA}(A)$. Figure 4.3 gives a timing diagram representation of the protocol.

After the initial anchor element broadcast message from the CA signed with $-K_{CA}$, subsequent key disclosure messages from the CA can be authenticated using one-way

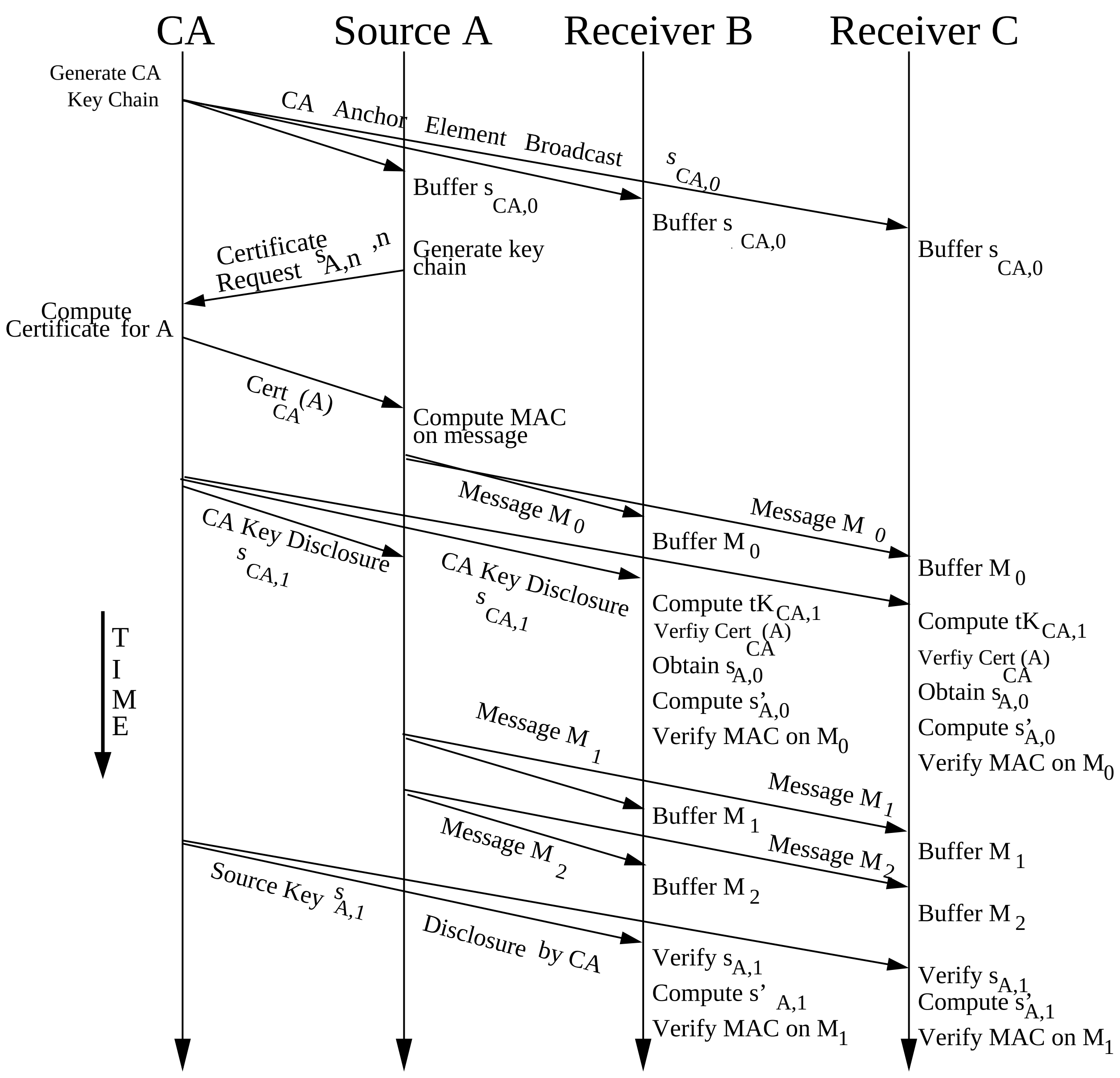


Figure 4.3: Time diagram for packet authentication using TESLA certificate algorithm

chains. For example, CA discloses the key $s_{CA,i}$ used in period $\langle t_i, t_i + d\rangle$ at time $t_i + d$. Receiver B can verify that $s_{CA,i}$ belongs to CA's one-way chain:

$$s_{CA,i} \xrightarrow{F_1} s_{CA,i-1} \xrightarrow{F_1} \ldots \xrightarrow{F_1} s_{CA,0} \tag{4.14}$$

where $s_{CA,0}$ has been verified before using $+K_{CA}$. B does not need to check CA's signature to verify $s_{CA,i}$.

Thus messages from A to B and C can be authenticated. The above algorithm requires A to perform one signature verification to verify the certificate it receives from the CA (4.7). Each receiver also performs one signature verification on the anchor element broadcast message from the CA (4.5). Since A is not a receiver, it does not need the verification in (4.5). All other messages from the CA and the sources can be authenticated using low-computation symmetric MACs. Moreover, sources and receivers do not have to perform clock synchronization directly with one another, synchronizing with the CA is a necessary and sufficient condition for the protocol. This saves additional message rounds and protocol complexity, and also breaks the cyclical dependency between authentication and clock synchronization.

4.2.7 Revocation of TESLA Certificates

The CA might need to broadcast a certificate revocation message at any time circumstances warrant that the TESLA certificate of a node has to be revoked. Assume the CA revokes the TESLA certificate of node A in the time period $\langle t_i, t_i + d\rangle$. Then the CA broadcasts the following message to the network:

$$CA \rightarrow network\colon \ \big(\langle t_i, t_i + d\rangle, REVOKE\left(Cert_{CA}\left(A\right)\right), MAC_{tK_{CA,i+1}}\left(..\right)\big) \tag{4.15}$$

The receiver buffers the message and waits for the CA to disclose $s_{CA,i+1}$ at time $t_i + d$. The traffic received from A in the intermediate period is also buffered, awaiting the verification of the revocation message, due to the possibility that the revocation message might be a fake. At time $t_{i+1} = t_i + 2d$, the CA broadcasts $s_{CA,i+1}$ to the network. B can verify the authenticity of $s_{CA,i+1}$ from (4.14) and thus validate the revocation message. If the revocation message is correctly verified, the receiver discards the buffered messages from A and adds the sender to the revoked users list.

The revocation message can be merged with the key disclosure message, the combined message looks like:

$$CA \rightarrow network\colon \ \big(\langle t_i, t_i + d\rangle, REVOKE(..), s_{CA,i}, MAC_{tK_{CA,i+1}}\left(..\right), SIGN_{-K_{CA}}\left(..\right)\big) \tag{4.16}$$

where the REVOKE field will contain the TESLA certificates to be revoked, the MAC is computed on the revoked certificates and the signature verifies $s_{CA,i}$ for nodes that might need the verification (instead of verifying $s_{CA,i}$ using (4.14)).

4.2.8 Non-repudiation of TESLA Certificates

Non-repudiation is not provided by the TESLA authentication algorithm [45] or the prior TESLA certificate proposal [48]. The symmetric nature of the basic cryptographic primitive used here - MACs - does not allow for non-repudiation. Once the hash key for a particular MAC is disclosed, any group member would be able to generate the MAC for the given message. Therefore, at a later instant in time, it is impossible to prove that the message was generated by a particular source. The lack of non-repudiation is a major drawback of the TESLA certificate protocol compared to digital signatures based on public keys.

In our extended TESLA certificate algorithm, we propose to add non-repudiation by taking advantage of the satellite infrastructure and the proposed mechanism of key disclosure by proxy. This is achieved as follows. The source authenticates each message by *two or more* MACs, computed using keys from two or more key chains, respectively. The anchor element of the root key of each chain is shared between the source and the CA (the satellite) as described in 4.2.4. The source includes all the MACs with each message transmission. Each receiver buffers the message along with all the MACs if the basic security check is satisfied, as described in the protocol in 4.2.6. At the time of key disclosure, the CA broadcasts *only one of the MAC keys* out of the set of MAC keys for the given source and message. Each receiver verifies the single MAC associated with the key broadcast by the CA, and accepts the message as correct if the MAC is verified. If any receiver wants to be able to check the message for non-repudiation at a later time instant, it saves the message along with all its MACs.

The MAC key that is disclosed by the CA is chosen at every disclosure instant, with uniform probability from the set of available keys for that time interval. Therefore, the source cannot know in advance, with a high degree of probability, which key will be used by the receivers for authentication. Hence, if the source would like its messages to be accepted by the receivers, it will have to include all the MACs correctly computed with the corresponding keys.

If at a later instant in time, a receiver would like to prove that a message was indeed generated by the source (i.e., non-repudiation), the receiver can simply send a non-repudiation request to the CA. Upon receiving the request, the CA discloses one of the *previously undisclosed* MAC keys for the message in question. The receiver can compute the MAC for the message with the newly disclosed key and compare the MAC with the set of MACs it had saved previously. If the CA and the receiver operates correctly, the newly computed MAC will match one of the saved MACs. Since: (i) the undisclosed MAC keys were known only to the source and the CA, and (ii) the CA is universally trusted, therefore the saved MAC must have been computed by the source using its MAC key and hence the message must have been generated by the source. Thus non-repudiation is achieved.

The security of the above algorithm is proportional to the number of MACs included with each message. For two MACs per message, the probability of a particular key being disclosed by the CA is 0.5. We term this probability the *r-factor*, where r is acronym for repudiation. It is computed as the inverse of the number of MACs included with each message. A non-conforming source who includes only one correctly computed MAC with its message in order to avoid non-repudiation, can expect the message to be accepted by the receivers only with 50% probability. If four MACs are included with every message, the r-factor drops to 0.25, and so on. There is hence a trade-off between the strength of the non-repudiation algorithm and the security overhead per message in terms of number of MACs involved. There is also the processing overhead at the source since it node has to compute M, number of MACs per message where $M > 1$.

The number of MACs per message also affect the security of the algorithm in the context of the receivers. If there are two MACs per message, the non-repudiation mechanism will be successful for the request from one receiver. For any subsequent request from other receivers for that particular message, non-repudiation will fail since both MAC keys are now known to the receivers. The number of successful non-repudiation requests for a given message is therefore directly proportional to the number of MACs per message. This drawback can be solved by modifying the protocol steps for non-repudiation. Instead of sending a request for an undisclosed key, the receiver can send the entire message along with the saved MACs, to the CA. The CA itself will compute the MACs on the message with any one of the undisclosed keys and compare with the saved MACs sent by the receiver. Since the undisclosed keys are known only to the CA and the source, in the event of a match, the CA can confirm to the receiver that the message was indeed generated by the source. The security of this mechanism depends only on the amount of trust placed on the CA and is independent of the number of MACs per message. The tradeoff is the additional load on the CA and the network overhead in transmission of the message with the MACs, to the CA.

4.2.9 Key Disclosure Delay

4.2.9.1 Time Synchronization

Time synchronization between various parties taking part in the communication is important for the correct operation of the protocol. Due to the global reach of the satellite, we consider it as the time reference. It periodically broadcasts its local time to the network and each terrestrial node synchronizes its time with the satellite reference. Figure 4.4 illustrates the time synchronization of the terrestrial nodes with the satellite/CA. The time synchronization works as follows.

- At periodic intervals, the CA broadcasts its local time t_{CA} to the network, authen-

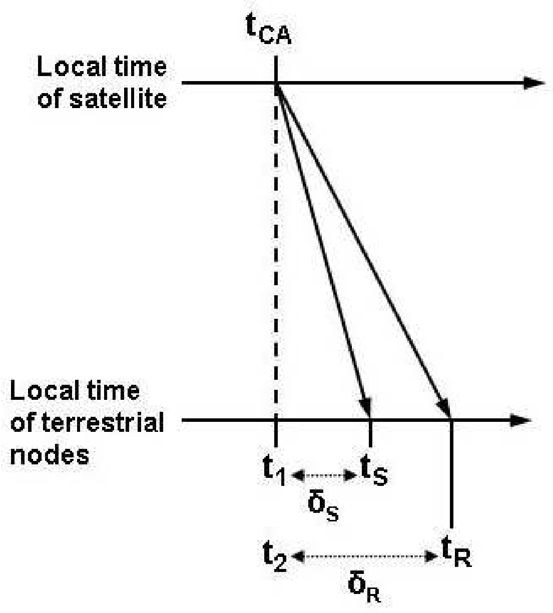


Figure 4.4: Time synchronization between the protocol participants in the extended TESLA certificate protocol

ticated with a digital signature. The local times at that instant at a terrestrial source node S and receiver node R are t_1 and t_2, respectively.

- Sender S receives the CA time broadcast at its local time t_S. The sender computes the maximum difference in time from the CA as $t_S - t_{CA} = t_S - t_1 + t_1 - t_{CA} = \delta_S + \epsilon_S$ where ϵ_S is the synchronization error of the source clock with the time reference. Therefore the upper bound on the CA's local time, with reference to S, is $t \leq t_S + \delta_S + \epsilon_S$.

- Receiver R receives the CA time broadcast at its local time t_R. The receiver computes the maximum difference in time from the CA as $t_R - t_{CA} = t_R - t_2 + t_2 - t_{CA} = \delta_R + \epsilon_R$ where ϵ_R is the synchronization error of the source clock with the time reference. Therefore the upper bound on the CA's local time, with reference to R, is

$t \leq t_R + \delta_R + \epsilon_R$.

The above method of time synchronization with the CA also indirectly synchronizes the time between the terrestrial nodes. After synchronization, the difference in time between the nodes S and R is $\delta_S - \delta_R + \epsilon_S - \epsilon_R$. It can be deduced from figure 4.4 that δ_S and δ_R are the network propagation times from the satellite to the terrestrial nodes, i.e., $\delta_S - \delta_R$ and hence the difference in local time is only $\epsilon_S - \epsilon_R = \epsilon_{SR}$. If S and R are well-synchronized with the CA, we have $\epsilon_{SR} \approx 0$.

4.2.9.2 Computation of the Key Disclosure Delay

The key disclosure delay d is a critical parameter affecting both the security and the performance of the proposed protocol. The key disclosure delay depends on the duration of each time interval Δ and the network propagation delay from the sources to the receivers. As is shown below, if the message transmission from the sources to the receivers happen exclusively over the satellite links, then the key disclosure delay depends only on Δ and the satellite link propagation delay, which is known and fixed.

We follow the method outlined in [46] in computing the key disclosure delay for our proposed protocol. Let us consider a packet p_j being sent from source S to receiver R in time interval I_j. Let the local times at S be ${t_j}^S$ when the packet is sent, while the local time at R is ${t_j}^R$, when the packet is received. The requirement to satisfy the security condition is:

$$\lfloor \frac{{t_j}^R + \delta_R + \epsilon_R - t_0}{\Delta} \rfloor - I_j < d \tag{4.17}$$

where δ_R and ϵ_R are the propagation delay from the CA to the receiver and the time synchronization error respectively, as explained in section 4.2.9.1.

Also, we must have:

$$t_j{}^S < t_0 + I_j * \Delta + \Delta \tag{4.18}$$

If D_{SR} is the network propagation delay from S to R, then:

$$D_{SR} = t_j{}^R + \epsilon_R - t_j{}^S - \epsilon_S \tag{4.19}$$

where ϵ_S is the time synchronization error for S.

From (4.17), (4.18), (4.19), we get:

$$\begin{aligned} & \lfloor \frac{D_{SR} - \epsilon_R + t_j{}^S + \epsilon_S + \delta_R + \epsilon_R - t_0}{\Delta} \rfloor - I_j < d \\ \text{i.e.,} \quad & \lceil \frac{D_{SR} + t_0 + I_j * \Delta + \Delta + \epsilon_S + \delta_R - t_0}{\Delta} \rceil - I_j < d \\ \text{i.e.,} \quad & \lceil \frac{D_{SR} + \epsilon_S + \delta_R}{\Delta} \rceil + 1 < d \end{aligned} \tag{4.20}$$

If we assume that the packet transmission from S to R is over the satellite links, then D_{SR} is one-way propagation delay over the satellite from S to R, while δ_R is the direct propagation delay from the satellite to any terrestrial node. We thus have $D_{SR} = 2 * \delta_R$ and hence from (4.20), we get:

$$\lceil \frac{3 * \delta_R + \epsilon_S}{\Delta} \rceil + 1 < d \tag{4.21}$$

For a given satellite configuration, δ_R is known and fixed. For example, for a geo-stationary satellite, δ_R is of the order of 0.12 seconds. Moreover, for good synchronization of the terrestrial nodes with the time reference, ϵ_S is negligible compared to the satellite propagation delay. Hence, (4.21) gives

$$\lceil \frac{0.36}{\Delta} \rceil + 1 < d \tag{4.22}$$

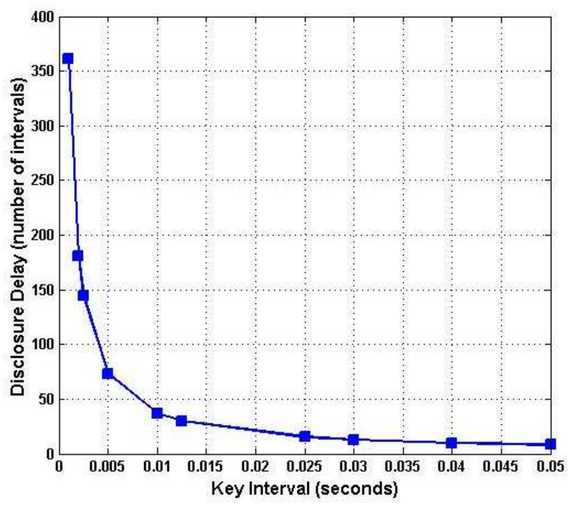


Figure 4.5: Key disclosure delay d as a function of key use interval Δ

Figure 4.5 shows the variation in the key disclosure delay d as a function of the key use interval Δ. As the figure demonstrates, the product of the disclosure delay and the key use interval is bounded by the network propagation delay, which is a constant. Hence as the key use interval increases from 1ms to upto 50ms, the disclosure delay varies between 361ms and 410ms.

4.3 Analysis of Protocol Correctness of the Extended TESLA Certificate Approach

The modified TESLA certificate approach provides strong authentication guarantees to a protocol participant. We analyze the correctness of the protocol using the Strand Space Model (SSM)[49]. SSM has several advantages over other security protocol correctness analysis methods, especially that it models exact causal relation information between events which makes the proofs concise. In our protocol analysis, we make use of the definitions and propositions that are explained in [49], including defining the following:

- **T** $\subset$ **A** is the set of texts representing the atomic terms, where **A** is the set of terms.
- **K** $\subset$ **A** is the set of cryptographic keys, disjoint from **T** and equipped with unary operator inv:**K**$\rightarrow$**K**. For symmetric keys, $inv(K) = K$, while for asymmetric keys, $inv(K) = K^{-1}$.
- **T**$_{name}$ $\subseteq$ **T** is the set of names, containing elements such as A, B.
- K: **T**$_{name}$ $\rightarrow$**K** is an injective mapping that associates a public key with each principal. For example, the public key of the Certificate Authority would be: $K(CA) = K_{CA}$. The corresponding private key: inv(K_{CA})=${K_{CA}}^{-1}$.
- We model the penetrator as defined in section A.4 of [50], having a *penetrator set* of keys K_p and and penetrator strand with traces of the type **M**, **K**, **C**, **S**, **E** or **D**.

Correctness proofs in SSM make use of the authentication agreement properties proposed in [51]. A protocol guarantees a participant B (for example, the receiver) agreement for certain data items $\vec{x}$ if each time a principal B completes a run of the protocol as a responder using $\vec{x}$, supposedly with A, then there is a unique run of the protocol with the principal A as initiator using $\vec{x}$, supposedly with B.

A weaker non-injective agreement does not ensure uniqueness, but requires only that each time principal B completes a run of the protocol as responder using $\vec{x}$, apparently with A, then there exists a run of the protocol with principal A as initiator using $\vec{x}$, apparently with B. We prove the weaker agreement by showing that whenever a bundle

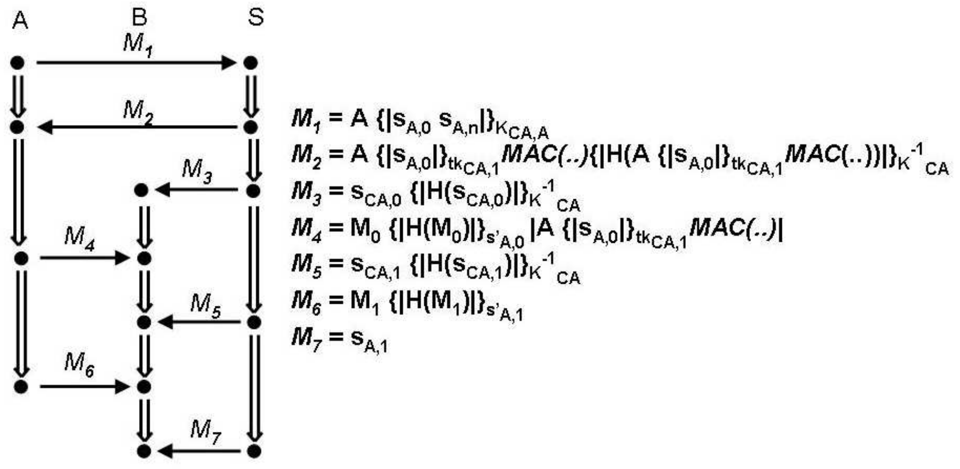

A
B
S
M1
M2
M3
M4
M5
M6
M7
$M_1 = A\ \{|s_{A,0}\ s_{A,n}|\}_{K_{CA,A}}$
$M_2 = A\ \{|s_{A,0}|\}_{tk_{CA,1}} MAC(..)\{|H(A\ \{|s_{A,0}|\}_{tk_{CA,1}} MAC(..))|\}_{K^{-1}_{CA}}$
$M_3 = s_{CA,0}\ \{|H(s_{CA,0})|\}_{K^{-1}_{CA}}$
$M_4 = M_0\ \{|H(M_0)|\}_{s'_{A,0}}\ |A\ \{|s_{A,0}|\}_{tk_{CA,1}} MAC(..)|$
$M_5 = s_{CA,1}\ \{|H(s_{CA,1})|\}_{K^{-1}_{CA}}$
$M_6 = M_1\ \{|H(M_1)|\}_{s'_{A,1}}$
$M_7 = s_{A,1}$

Figure 4.6: Strand Space Model representation of initial message exchange in the extended TESLA certificate protocol

C contains a receiver strand using $\vec{x}$, then C also contains a source strand using $\vec{x}$. We establish agreement by showing that C contains a unique source strand using $\vec{x}$.

4.3.1 Strand Space Model for extended TESLA Certificate Algorithm

Figure 4.6 shows the SSM representation of the initial message exchanges in the extended TESLA certificate algorithm, with A as the source or initiator, B as the receiver or responder, and S as the Certificate Authority. We consider only these three principals in our analysis, with B representing the set of receivers. To prove that messages authenticated by B are sent uniquely by A, only the first few steps shown in figure 4.6 are important, since subsequent message authentication is simply a repetition of M_5 through M_7. In the analysis we consider only the important units of the messages, which are termed *components* [50]. We prove the authentication properties of the extended TESLA protocol using a series of authentication tests, of the type defined in [50].

Let Σ be a strand space model of the extended TESLA certificate protocol. In Σ, the regular strands are defined to be of the form:

1. Source strands in Src[A, s, M, K], with trace:

$$\langle +A|s_{A,0}s_{A,n}|_{K_{CA,A}}, -A|s_{A,0}|_{tk_{CA,1}}MAC(..)|H(A|s_{A,0}|_{tk_{CA,1}}MAC(..))|_{K_{CA}{}^{-1}}, \\ +M_0|H(M_0)|_{s'_{A,0}}|A|s_{A,0}|_{tk_{CA,1}}MAC(..)|, +M_1|H(M_0)|_{s'_{A,1}}\rangle$$

2. Receiver strands in Recv[s, M, K] with trace:

$$\langle -s_{CA,0}|H(s_{CA,0})|_{K_{CA}{}^{-1}}, -M_0|H(M_0)|_{s'_{A,0}}|A|s_{A,0}|_{tk_{CA,1}}MAC(..)|, \\ -s_{CA,1}|H(s_{CA,1})|_{K_{CA}{}^{-1}}, -M_1|H(M_0)|_{s'_{A,1}}, -s_{A,1}\rangle$$

3. CA strands in CA[A, s, K] with trace:

$$\langle -A|s_{A,0}s_{A,n}|, +A|s_{A,0}|_{tk_{CA,1}}MAC(..), +s_{CA,0}|H(s_{CA,0})|_{K_{CA}{}^{-1}}, \\ +s_{CA,1}|H(s_{CA,1})|_{K_{CA}{}^{-1}}, +s_{A,1}\rangle$$

Here $A, B, CA \in \mathbf{T}_{name}$ and $s_{A,i}, s'_{A,i}, s_{CA,i}, tk_{CA,i} \in \mathbf{K}\ \forall i$.

Let LT be the set of long-term keys, which includes the public-private key pair of the CA, and the long-term secret key $K_{CA,A}$ that each principal shares with the CA. We further assume:

1. The CA generates keys in a reasonable manner, i.e., CA[*,s, K]=Φ unless: $s_{CA,i}, tk_{CA,i} \notin K_P$; $tk_{CA,i} = H(s_{CA,i})$; $s_{CA,i} = s_{CA,i}{}^{-1}, tk_{CA,i} = tk_{CA,i}{}^{-1}$; $s_{CA,i}$ is uniquely originating (and therefore $tk_{CA,i}$); and $s_{CA,i}, tk_{CA,i} \notin \mathsf{LT}\ \forall i \in [0..n]$.

2. Similar to above, the source generates the authentication keys in a reasonable manner.

The authentication guarantees provided by the protocol are achieved in the following steps.

1. The long-term keys LT are not uttered by the protocol. Thus if $K \in$LT and $K \notin K_P$, then $K \in S_0$ where S_0 is the set of safe keys as defined in definition 2.3 of [50]. Hence, if the CA signs a message with its private key ${K_{CA}}^{-1}$, then the certificate is valid.

2. The short-term authenticating keys are valid till they are uttered by the protocol. Thus $tk_{CA,i} \in S_0$ till event M_5. Similarly, $s_{A,0} \in S_0$ till event M_5 and $s_{A,1} \in S_0$ till event M_7.

3. The CA receives an unsolicited test that authenticates the initial positive node of the source.

4. The source strand contains an outgoing test for $s_{A,0}$ in $|s_{A,0}s_{A,n}|_{K_{CA,A}}$. This authenticates the server strand.

5. The receiver strand receives an unsolicited test that authenticates the second positive node of the CA strand.

The receiver authenticates the source only in that it authenticates the CA strand, which has authenticated the occurrence of the source strand's initial positive node.

We use unsolicited tests to prove the CA's authentication guarantee.

Proposition 4.1 *Suppose that C is a bundle in Σ; $K_{CA,A} \notin K_P$; and $s \in$ CA[$A, s,$*] has C-height 1.*

Then there exists $s_i \in$ Src[$A, s, M, *$] such that s_i has C-height 1.

PROOF. The term $|s_{A,0}s_{A,n}|_{K_{CA,A}}$ is an unsolicited test, and therefore Authentication Test 3 [50] occur on positive regular nodes in C. Let S be the set of all regular nodes in C having $|s_{A,0}s_{A,n}|_{K_{CA,A}}$ as a component. Since S is non-empty, it has $\leq_C$ minimal member n_0 (Lemma 2.6 in [49]). By Lemma 2.7 in [49], the sign of n_0 is positive. Since the sign of $s \in$ CA[A, s,*] is negative and it has C-height 1, n_0 can only be of the form $\langle s_i, 1\rangle$ for $s_i \in$ Src[$A, s, M, *$].

The following two propositions cover the receiver's authentication guarantees.

Proposition 4.2 *Suppose that C is a bundle in Σ; $A \neq B$; $K_{CA,A}, {K_{CA}}^{-1}, s_{CA,1} \notin K_P$; and $s_r \in$ Recv[$s, M, *$] has C-height 3.*

Then there exists $s_{CA} \in$ CA[$A, s, *$] with C-height 4. Furthermore, there exists $s_i \in$ Src[$A, s, M, *$] such that s_i has C-height 3.

The next proposition is similar to proposition 4.2, but includes the case to cover the authentication guarantee of the receiver for events M_6 and M_7:

Proposition 4.3 *Suppose that C is a bundle in Σ; $A \neq B$; $K_{CA,A}, {K_{CA}}^{-1}, s_{CA,1}, s'_{A,1} \notin K_P$; and $s_r \in$ Recv[$s, *, *$] has C-height 5.*

Then there exists $s_{CA} \in$ CA[$A, s, *$] with C-height 5. Furthermore, there exists $s_i \in$ Src[$A, s, *, *$] such that s_i has C-height 4.

4.4 Security Analysis: Prevention of Authentication Attacks

The extended TESLA certificate authentication protocol is resistant to active attacks by malicious nodes in the network. In the following sections we discuss the security provided by the protocol against specific active attacks. In our analysis, we assume that the CA is always secure, since compromise of the CA is a single point of failure for security in the network.

4.4.1 Malicious Node with Connectivity to Source and Receiver

We consider the case where a malicious node X in the network attempts to create fake packets from a source to the receiver(s). Without loss of generality, we consider one source A is sending data to one receiver B, the data being authenticated using the extended TESLA certificate protocol. We assume that X can hear packet transmissions from A, and can also transmit to B. X can also receive the broadcast messages from the CA. Therefore, shortly after time $t_0 + d$, X has knowledge of $Cert_{CA}\left(A\right)$, message M_0 from A to B, $s_{CA,0}$ broadcast by the CA, and $s_{A,0}$ from the certificate. X can verify that $s'_{A,0}$ belongs to the authentication hash chain of A by performing the verification procedure. Having obtained a verified element of A's authentication chain, X can attempt to spoof messages as coming from A, starting at time $t_0 + kd$, where $k > 0$. To achieve this, X needs to generate $s_{A,k}$ from $s_{A,0}$ where $s_{A,k} = F_1^{-k}\{s_{A,0}\}$. Due to the one-way property of F_1, this is computationally infeasible for X, and is of complexity $O\left(2^K\right)$, where each element of the hash chain is K bits and K is assumed to be large. Without a valid $s_{A,k}$, it would be impossible for X to spoof a message that would be successfully authenticated by B.

X could also attempt to spoof packets from A at any time between $\langle t_0, t_0 + d\rangle$. This would require that X successfully generate an element of A's hash chain without knowledge of any legitimate element of the hash chain. This has the same computational complexity of $O\left(2^K\right)$ and is computationally infeasible for any X with finite resources.

A third approach X could attempt would be to generate an independent hash chain that produces the hash value $s_{X,0}$ that is computationally indistinguishable from $s_{A,0}$. This would allow X to use element $s_{X,0}$ of its own hash chain to authenticate messages purportedly generated by A. However, this is computationally infeasible due to the collision-resistance property of F_1 and F_2.

Failing any attack on A's hash chain as above, X could attempt to masquerade as the CA and generate a fake certificate for A as in equation (4.7), and also generate fake CA key disclosure broadcast message similar to equation (4.10). However, unless X knows the CA private key $-K_{CA}$, it will not be able to correctly sign the fake $Cert_{CA}\left(A\right)$, and therefore the fake certificate will be rejected by A. Likewise, the fake CA broadcast message from X will be rejected by the receivers unless the signature in the message is verified as correct using $+K_{CA}$. As per our assumption of the security of the CA, $-K_{CA}$ is known only to the correct CA, and therefore X would not be successful in this attack.

X could attempt to fake CA key disclosure messages subsequent to (4.10), but (a) the fake hash element $s_{CA_X,i}$ will not verify successfully to the anchor element $s_{CA,0}$ and (b) this does not allow X to fake elements of A's hash chain.

4.4.2 Attack on the CA Revocation Messages

A malicious node X in the network can attempt to broadcast fake revocation messages, similar to (4.15), and thereby attempt to disqualify legitimate sources in the network. To generate a fake revocation message that will be successfully accepted by the receivers, X should be able to compute a MAC on the fake revocation message using the key $s_{CA,i+1}$, with knowledge of at most the key $s_{CA,i}$ where $s_{CA,i+1} = {F_1}^{-1}\{s_{CA,i}\}$. Using reasoning similar to the previous section, owing to the one-way property of F_1, this has computational complexity $O\left(2^K\right)$ and is infeasible for X. At most, X can trick the receivers in buffering the fake revocation message, till the next message disclosure from the CA, when the MAC on the fake message will not verify correctly using the recently disclosed (correct) $s_{CA,i+1}$, and therefore be discarded.

Based on the security analysis above, the extended TESLA certificate approach is secure against message spoofing attacks by malicious nodes in the network.

4.5 Performance Analysis of extended TESLA Certificate Algorithm

We have run simulations of the extended TESLA certificate authentication protocol to analyze the demands the protocol can make of node resources, and also to compare it with widely-used signature-based authentication protocols. In our performance analysis, we consider that all security protocol and data messages from the source to the receivers are sent over the satellite channel where the satellite is in Ka-band and geostationary orbit. Therefore, we also assume that the one-way satellite propagation delay is of the order of 130ms and the terrestrial propagation delays from the group nodes to the local gateways are negligible in comparison. Moreover, the satellite uplink bandwidth is much less in comparison to the satellite downlink bandwidth and usually also lower than the terrestrial wireless bandwidth (assuming the wireless MAC protocol is 802.11agn). By

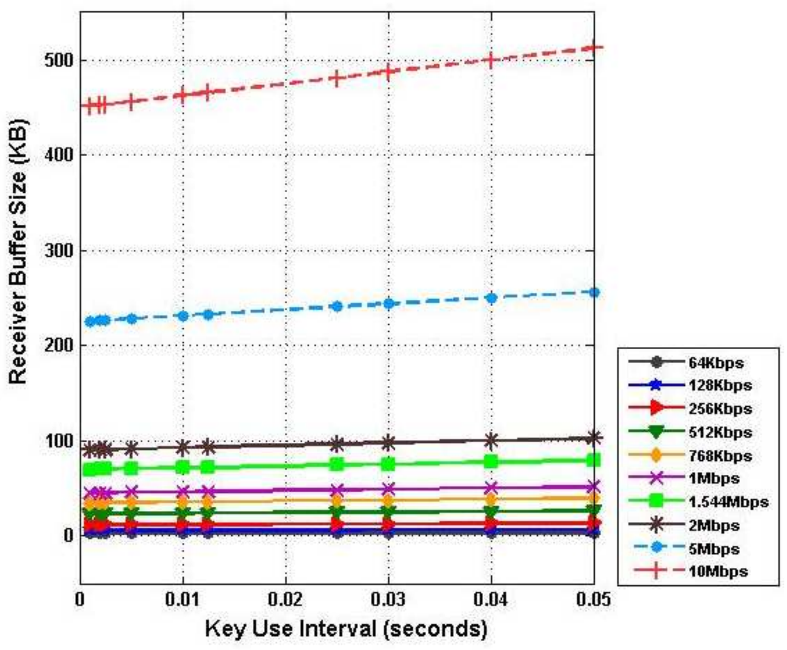


Figure 4.7: Receiver buffer size for varying network bandwidths and key use intervals

our assumption, all data has to traverse the satellite uplink and hence we limit the overall network bandwidth to be equivalent to the satellite uplink bandwidth. In the following analyses, the uplink bandwidth is varied between 64Kbps and 10Mbps.

An important resource consideration is the amount of memory or buffer required in the receivers for temporary storage of the data packets that are pending authentication (that is, they MAC key has not been disclosed by the CA). Figure 4.7 shows the variation in the size of buffer required for different Δ and varying network bandwidths. In theory, the receiver buffer size depends on the time interval of key use (since the key disclosure delay is a function of that) and the network bandwidth. As figure 4.7 shows, the buffer size varies little with the key disclosure delay since it is limited to a narrow range by the network propagation delay. However, the buffer size is significantly affected by the rate

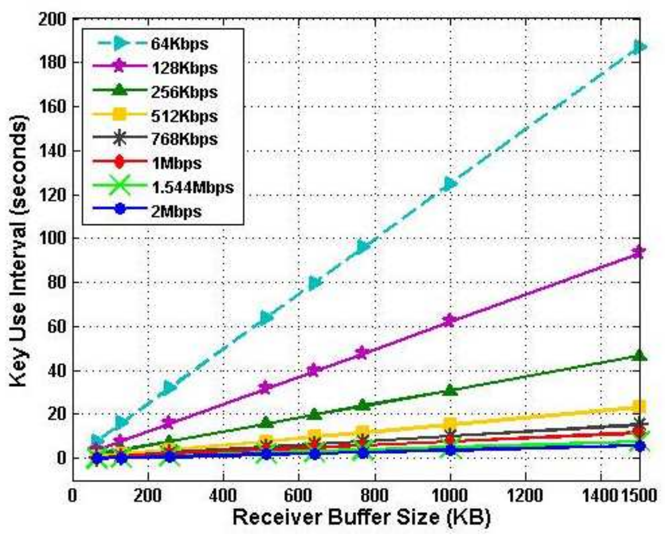


Figure 4.8: Key use interval Δ as a function of receiver buffer size and network bandwidth

at which data is received - for higher rates, larger buffer is required. Even so, the buffer requirement is only in the order of hundreds of kilo-bytes, which is a small fraction of the memory present in mobile computers today.

For network scenarios where the receiver buffer is fixed due to hardware constraints, the key disclosure delay (and hence the key use interval) has to be dynamically determined based on the smallest buffer size available, and the network bandwidth. As shown in figure 4.8, the key use interval can be longer for larger buffers, while it is shorter as for higher network bandwidths.

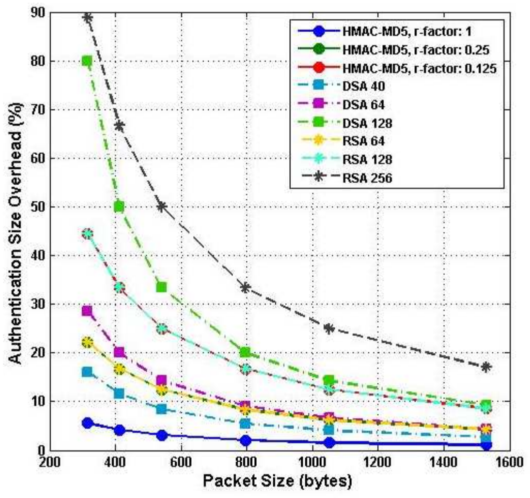
The size of the TESLA certificate and the MACs computed on each message, compare favorably to digital certificates and signatures used in public key-based cryptography. If the MAC algorithm is based on SHA-1 [52], the key used is 160 bits. For a TESLA certificate with fields as shown in 4.2, the certificate size is computed as follows.

- 32 bits for the ID - this will cover 4 billion nodes;
- 160 bits for the encrypted anchor element (128 bits if MD5 is used instead of SHA-1);
- 64 bits for the time field - this gives the current time in UTC since January 1, 1900, with a resolution of 200 pico-seconds [53];
- 160 bits for the MAC; and,
- 1024 bits for the CA digital signature (assuming PKCS# 1-based [54] digital signature with modulus 1024 bits)

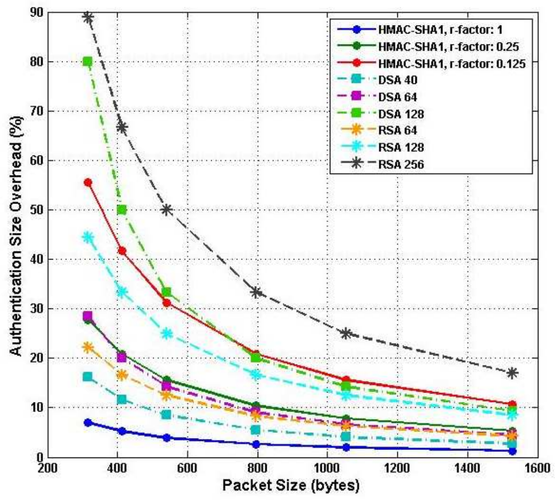
Therefore, the total size of the certificate is 1440 bits or 180 bytes. In contrast, a typical X.509 certificate size is of the order of 1KB.

The size of the MAC appended to each message in our protocol is 128 bits for HMAC-MD5 or 160 bits for HMAC-SHA1 (and with r-factor 1). In comparison, if RSA based digital signature is used, the signature size would be 512 bits, 1024 bits or 2048 bits for RSA modulus $N = 512, 1024, 2048$ respectively. For DSA or ECDSA-based digital signatures, the signature size is 320 bits for a security level of 80 bits [55]. A comparison of the size overhead incurred for authenticating 500MB data using our proposed protocol or DSA or RSA, is shown in figure 4.5. Figure 4.9(a) compares extended TESLA with HMAC-MD5 against DSA and RSA, while figure 4.9(b) compares extended TESLA with HMAC-SHA1 against DSA and RSA. Since the size of the message is larger than the size of the maximum bytes allowed per transmission, the message is split into smaller chunks for transmission and each individual chunk is authenticated separately. The graphs show how the overhead varies as a percentage of the total bytes transferred (message + MAC/signature) as the size of the IP packet varies between 316 bytes and 1528 bytes. For extended TESLA, we consider three cases based on the degree of non-repudiation present - for each packet, r-factor is 1 (one MAC), 0.25 (four MACs) or 0.125 (eight MACs). These are compared to DSA with signature sizes 40, 64 and 128 bytes, and RSA with signature sizes 64, 128 and 256 bytes (N=512,1024 and 2048 respectively). For all the cases, the overhead decreases with increase in the packet size because for the higher packet sizes there are lesser number of chunks and hence lesser number of MACs or signatures. The overhead for our basic protocol is the lowest of all the cases. As we add more MACs for non-repudiation, the overhead goes up and is a significant percentage for eight MACs (r-factor 0.125). This is the tradeoff in terms of size for our non-repudiation scheme. However, even then the overhead for our protocol is significantly less than the overhead due to strong RSA (256 byte signature) or DSA (128 byte signature)-based security. The graphs suggest that we should go for the largest packet size allowed by the underlying link layer protocol, since the overhead drops significantly. However, this has to be considered against the energy expense for authenticating larger packet sizes, as shown in some following figures. Figure 4.10 shows that the size overhead with HMAC-SHA1 is more than that with HMAC-MD5 for all r-factors, since each individual MAC for the former is 1.25 times in size that of the latter. However, for higher r-factors and larger packet sizes, the overheads tend towards equal values. The graphs again emphasize the need for larger packets, since the overhead drops from 50% for the worst-case scenario (HMAC-SHA1, r-factor 0.125, 316 byte packet), to 8.72% (1528 byte packet).

The number of cryptographic keys required for computing the MAC on each block,



(a) HMAC-MD5, DSA and RSA



(b) HMAC-SHA1, DSA and RSA

Figure 4.9: Comparison of percentage byte overhead due to authentication for 500MB data

for 1GB data, as a function of the buffer size and the network bandwidth, is shown in figure 4.11. Here for each time interval Δ, one key is used. Δ is also lower bounded by 10ms. The buffer size is varied from 64KB to 1536KB in steps of 64KB. All the graphs exhibit a knee for buffer size 128KB. The graphs show that for high transmission rates, the number of keys required is high for small buffers, while it decreases sharply as the buffer size increases. The decrease with increasing buffer sizes is more smooth for bandwidths lower than 1Mbps. For large buffer sizes, the number of keys required becomes largely independent of the network bandwidth. When the buffer sizes are small, the key use intervals have to be small too, and they are more susceptible to variations in the data rate. For large buffers, the effect of the bandwidth is less important. Therefore, one should consider the receiver buffer size as a more important factor than the network bandwidth in designing the protocol parameters.

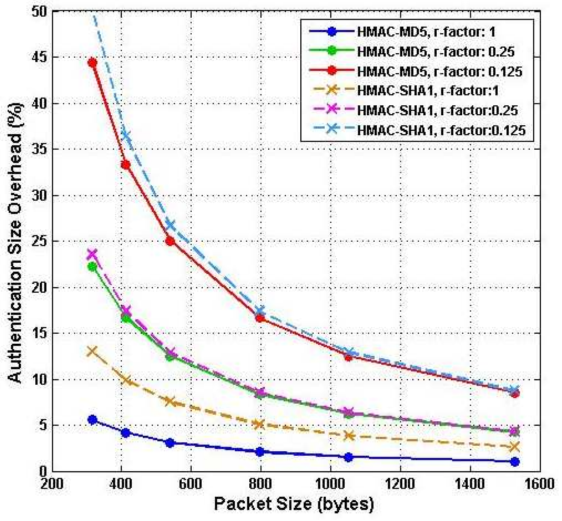


Figure 4.10: Percentage size overhead comparison between HMAC-MD5 and HMAC-SHA1, 500MB data

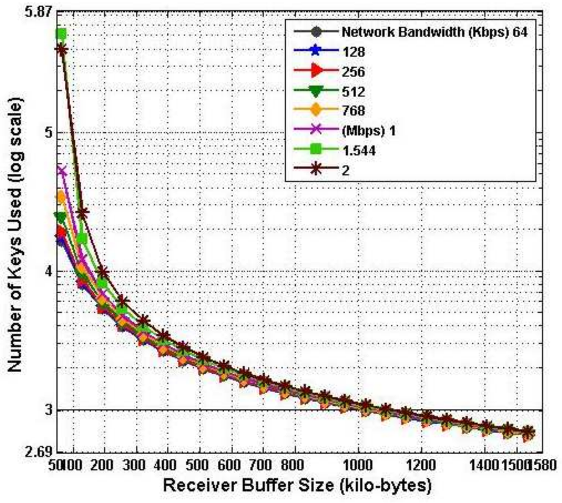
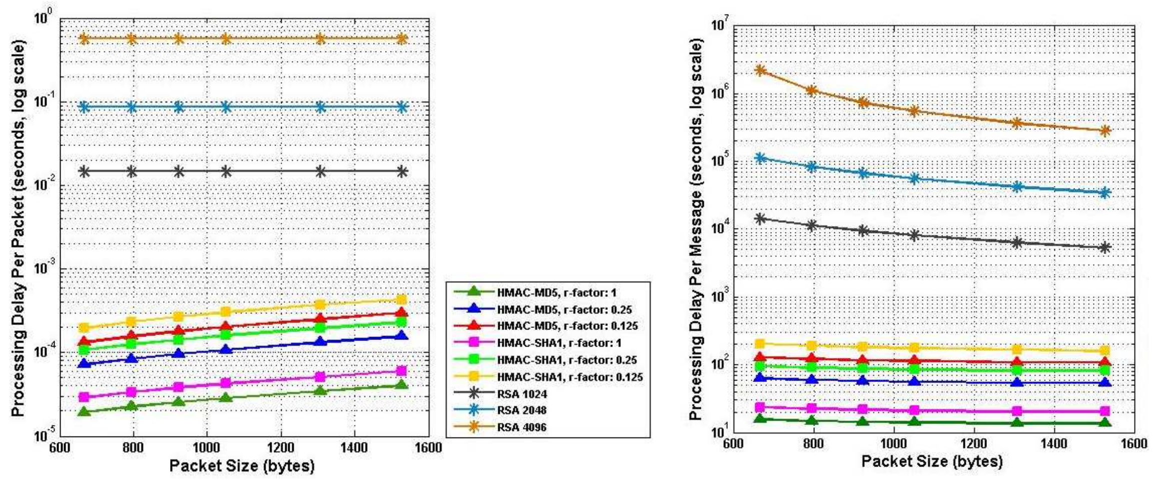


Figure 4.11: Number of keys required for authenticating 1GB message

Processor	**Key length (bits)**		
	1024	*2048*	*4096*
PIII-500MHz	14.6	85.6	562.8

Table 4.1: RSA signature timings (ms) per packet on 500MHz Pentium III

An analysis of the processing delay overhead of the extended TESLA protocol, and its comparison to the processing delay for RSA signatures, is given in figure 4.12. We simulate the delay due to authentication for 500MB data on a 500MHz Pentium III machine. The delay figures for HMAC-MD5 and HMAC-SHA1 are computed based on the approximation that each operation is executed in one processor clock tick for the 500MHz PIII processor. The delay figures for RSA per packet for the 500MHz PIII processor are from [56] and are reproduced here in table 4.1.



(a) Processing delay per packet (b) Processing delay per message

Figure 4.12: Comparison of authentication processing delay for 500MB data, PIII500MHz

Figure 4.12(a) validates a key feature of each algorithm compared here. For the HMAC algorithms, the processing delay is proportional to the number of 512-bit blocks being processed. As the packet sizes increase, the number of blocks per packet also increase and therefore the processing delay for HMAC increases. Also, HMAC-SHA1 has performs more operations (1110 per 512-bit block) compared to HMAC-MD5 (744 per 512-block) and this fact is reflected in the graphs. For RSA, the processing delay depends only on the size of the modulus N and is independent of the size of the message. The cheapest RSA delay, 14.6ms per packet (for modulus 1024 bit), is still significantly higher than the most expensive HMAC delay, 0.43ms (for HMAC-SHA1 with eight MACs). This is a major advantage of our protocol over signature-based schemes.

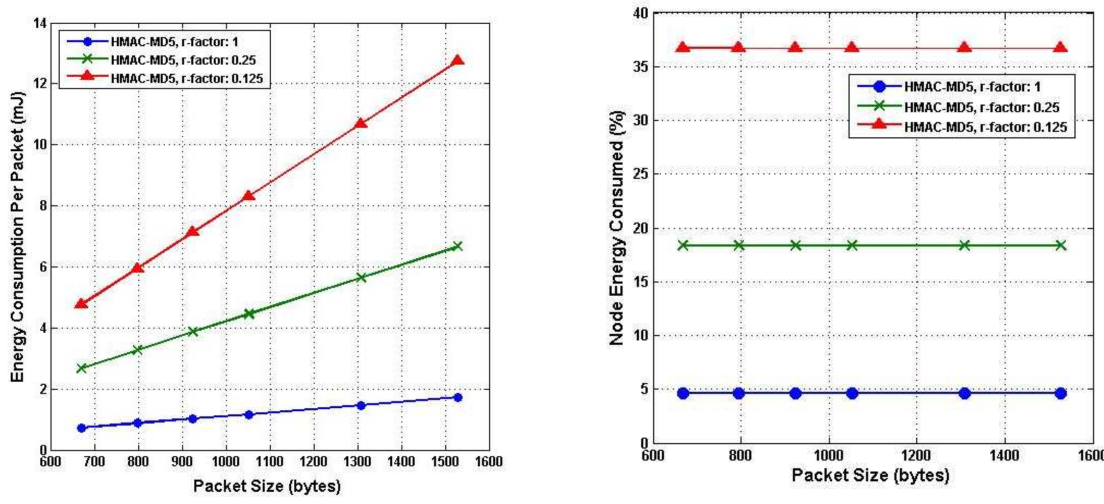
The total processing delay for authenticating 500MB data is shown in figure 4.12(b). The total number of 512-bit blocks in the overall message is independent of the size of each packet, and hence also the HMAC processing delay for the entire message. However, for RSA the processing delay is directly proportional to the number of message chunks processed. Here also our protocol performs significantly better than using RSA signatures - the worst-case delay is 204 seconds (for HMAC-SHA1, 8 MACs), while the best-case delay for RSA is a prohibitive 5321 seconds (for N=1024, 1528-byte packets).

We analyze the energy consumption of our protocol for authenticating 500MB data in figure 4.13. We simulate the energy consumption for a Compaq iPAQ H3670 handheld computer [57], which is fairly representative of the low-power terrestrial mobile nodes that we have considered in our protocol design. The handheld contains an Intel SA-1110 StrongARM processor clocked at 206MHz. It is powered by a Li-polymer battery with capacity 950mAh at 3.7V. The base figures for energy expenditure of different crypto-

Algorithm	**Key size** (*bits*)	**Sign** (*mJ*)	**Verify** (*mJ*)
RSA	1024	546.50	15.97
DSA	1024	313.60	338.02
ECDSA	163	134.20	192.23
HMAC	1.16 (μ*J/B*)		

Table 4.2: Energy cost of digital signature algorithms and HMAC, for Compaq iPAQ H3670

graphic operations of the handheld are obtained from [58] and are reproduced here in table 4.2. Figure 4.13(a) shows that the energy consumption for authentication each packet ranges between 0.7238mJ and 12.73mJ, with larger packets and higher r-factors consuming more energy, as is expected. The total battery capacity is 12654 Joules. Figure 4.13(b) shows the total energy consumed for authenticating 500MB data, as a percentage of the battery capacity. For higher r-factors, the energy consumption is a significant percentage of the capacity and implies that more than 500MB data cannot be authenticated without recharging. It is to be noted that in most cases, the higher energy expense for the higher r-factors is incurred by the source node only. The receivers can authenticate the messages by computing only one MAC and hence the figures for r-factor 1 is indicative of the energy expense of the receivers. Given the energy constraints, the maximum number of packets that can be authenticated by our protocol, using HMAC-MD5, is given by figure 4.14. In this simulation we assume that only 50% of the total energy is used in the protocol operation and the rest for other purposes such as packet transmission. The



(a) Energy consumed per packet (b) Percentage energy consumed for 500MB data

Figure 4.13: Energy consumption for HMAC-MD5 authentication only, iPAQ H3670

graphs show that for larger packet sizes, the total number of packets that can be processed are lower, since more energy is spent per packet, due to the higher number of 512-bit blocks. In the worst case, a source with r-factor 0.125 can process between 1.3 million (IP packet size 668 bytes) and 0.49 million (1528 byte per packet) packets. On the other hand, a receiver (with r-factor 1) can process between 8.74 and 3.66 million packets for packet sizes 668 bytes and 1528 bytes, respectively.

Figure 4.15 compares the amount of energy that would be required to authenticate 500MB data on the iPAQ handheld for different authentication algorithms. Clearly, authenticating 500MB data without additional energy sources is not possible except for the extended TESLA protocol. Given the energy constraints above, authenticating 5MB data on the handheld is more realistic - figure 4.16(a) compares the percentage of the total node energy that is spent in authenticating 5MB data. The graphs validate our claim that

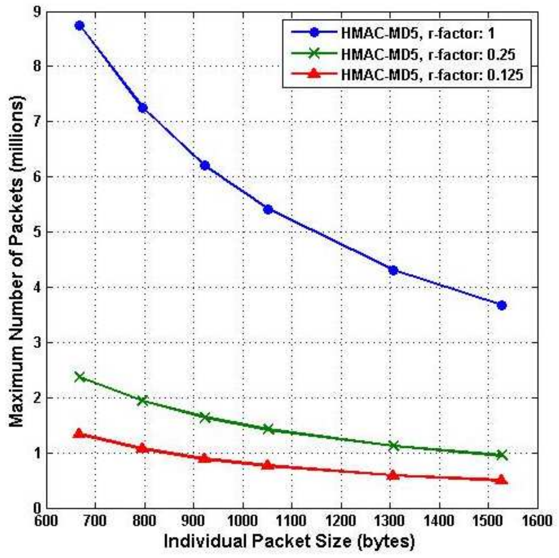


Figure 4.14: Total number of packets processed by extended TESLA with HMAC-MD5 on iPAQ H3670

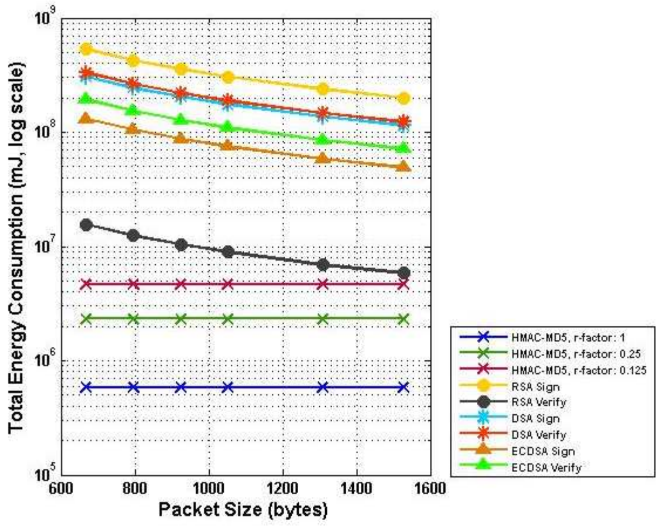
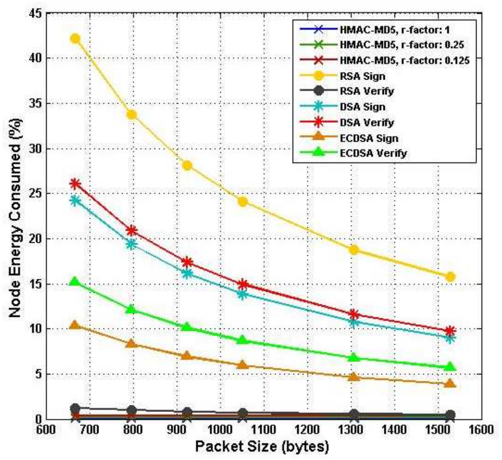
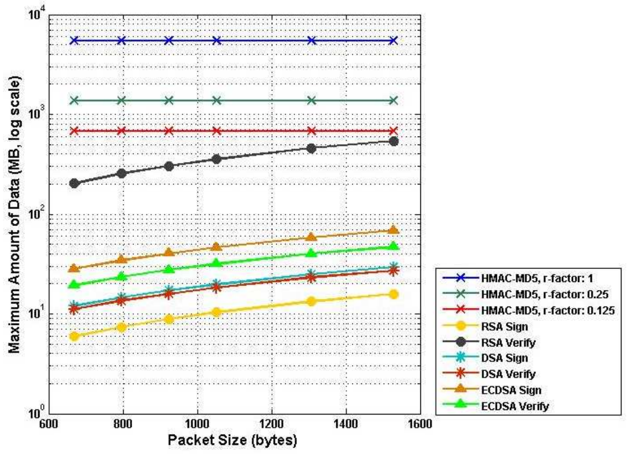


Figure 4.15: Comparison of total energy required for authenticating 500MB data on iPAQ H3670

the energy consumption of the extended TESLA protocol is significantly less in comparison to standard signature protocols, even efficient protocols like ECDSA. Moreover, the graphs show that for nodes with limited energy, the standard signature algorithms cannot be applied for authenticating every packet - even in the best case scenario, ECDSA signing algorithm consumes nearly 4% of the node energy for only 5MB data. For the iPAQ handheld, the standard protocols can authenticate only a few mega-bytes of data before they completely spend the available energy, as shown in figure 4.16(b). Here we assume that upto 50% of the node energy is spent in authenticating the data packets. The best scenario for the standard protocols is for RSA signature verification, where a node can authenticate nearly 543MB of data if it is split into 1528 byte packets. The worst scenario is for RSA signature generation, where only 6MB of data can be authenticated, for 668 byte packets. The extended TESLA protocol with HMAC-MD5 performs significantly better in comparison - being capable of authenticating 682MB even with r-factor 0.125.



(a) Percentage energy consumed for 5MB data



(b) Maximum amount of data processable

Figure 4.16: Energy performance comparison of different authentication protocols on iPAQ H3670

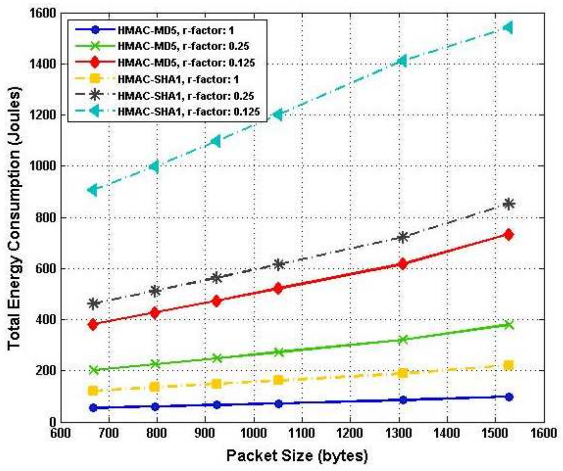
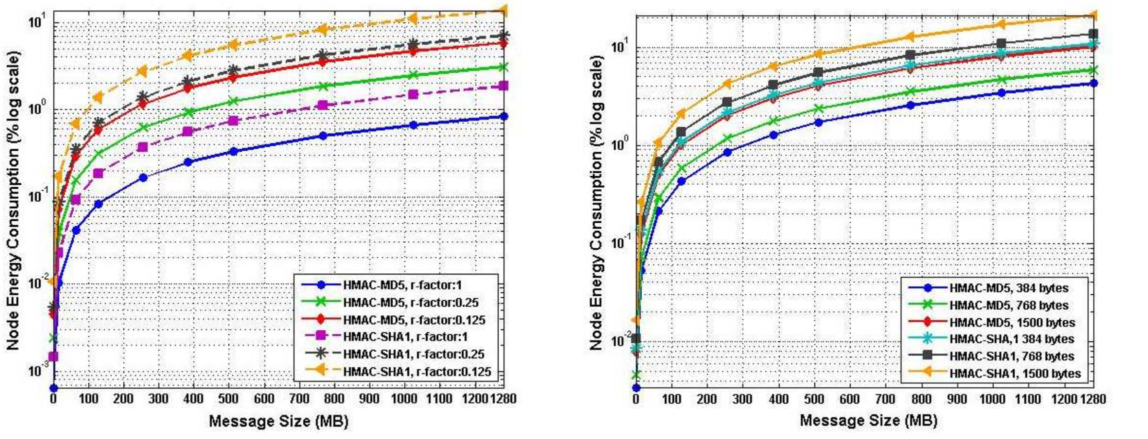


Figure 4.17: Total energy required for authenticating 500MB data, ARM1176JZ(F)

The analysis above with iPAQ H3670 has the limitation that the processor power and battery capacity is on the lower end of the scale for mobile nodes that are available today. To illustrate the energy performance of the extended TESLA protocol for a more representative handheld, we have performed a simulation analysis of the energy consumption for different message sizes on the Apple iPhone processor, ARM1176JZ(F) [59]. The processor has an operating frequency of 620Mhz and consumes 0.45mW per cycle. We consider the node energy capacity to be equivalent to the iPhone battery capacity, 1400mAh at 3.7V [60]. Under the assumption that one HMAC operation is performed in one cycle of the processor, figure 4.17 gives the total energy consumed for authenticating 500MB data, for both HMAC-MD5 and HMAC-SHA1. As is expected, HMAC-SHA1, r-factor 0.125, consumes the most energy. However, even in the worst case, the 1541 joules of energy consumed represent 8.2% of the total battery capacity. Finally, figure 4.5 shows how much energy is consumed for authenticating varying message sizes. Figure 4.18(a) gives the values for the node energy as a percentage of the battery capacity



(a) Percentage of total energy for varying r-factor (b) Percentage of total energy for varying packet sizes

Figure 4.18: Energy consumption of extended TESLA protocol on ARM1176JZ(F) for different message sizes

for both HMAC-MD5 and HMAC-SHA1, when the r-factor is varied. Here the individual UDP payload is held constant at 768 bytes. Even the highest consumption - HMAC-SHA1, r-factor 0.125 authenticating 1280MB data - is only 13.71% of the total battery capacity. When the packet size is varied, while keeping the r-factor constant (0.25), the energy consumption percentage is shown in figure 4.18(b). The maximum consumption is again for HMAC-SHA1, for UDP payload size 1500 bytes, but it is still an acceptable 21.16% of the total energy. Even though these figures are only approximate, they demonstrate that the extended TESLA protocol can be efficiently used for authentication in present-day mobile computers.

4.6 Summary

In this chapter, we have proposed a source authentication protocol for group communication in hybrid satellite/wireless networks. As a basic building block of the proposed protocol, we have used an extension a new class of lightweight, symmetric-key certificates called TESLA certificate. We have modified the TESLA certificate construct by including a keyed hash chain in each certificate that extends the lifetime of each certificate to multiple uses. The TESLA certificate construct binds the identity of the sender to the anchor elements of the sender's key chain. Messages from the sender are authenticated by MACs computed with keys from the chain. For the authentication protocol, we have used the satellite as the Certificate Authority to generate and distribute the TESLA certificates to all the nodes in the network. We have also used the satellite as the proxy node for the senders in disclosing the MAC keys to the receivers in the network. Furthermore, we have proposed a novel concept of probabilistic non-repudiation that is based on having the satellite node as the proxy for key disclosure to the receivers. In terms of performance, due to the use of symmetric MAC functions, the proposed protocol is much less expensive on node processing power and energy compared to digital signatures. Through simulations, we have shown that the performance of the authentication protocol is superior to using public-key based digital signatures for authentication, for the metrics node energy and processing delay. We have shown through security analysis that the protocol is secure against malicious adversaries. We have also proven the correctness of the protocol through Strand Space analysis.

4.7 Related Work

There has been significant research on efficient multicast source authentication algorithms based on symmetric cryptography that attempt to minimize the computation expense on the devices. Canetti et al [61] proposed one of the early solutions to use symmetric MACs for multicast source authentication. In their scheme, the source has l keys and computes l MACs on each packet. Each recipient holds a subset of the l keys, and verifies the MAC according to the keys it holds. The authentication protocol has probabilistic security - the choice of key subsets held by each recipient is critical in insuring that with high probability no coalition of up to w colluding members know all the keys held by a good member (where w is the security parameter), and thus maintaining the security of the scheme. The scheme also requires the multicast group members to store a large number of keys. [62] has proposed a method known as stream signing, where one regular digital signature is transmitted at the beginning of a stream, and each packet either contains a cryptographic hash of the next packet, or a one-time public key using which the one-time signature on the next packet can be verified. However, this approach requires reliable packet transmission, since the loss of even one packet means that the information required to authenticate future packets will be lost. For most multicast protocols, such reliability cannot be guaranteed, since the transmission protocol is UDP, which is best-effort. [63] has proposed an approach where the source is allowed to delay and group together several consecutive packets. The source collects the packets in a time-interval into an authentication tree and signs the root of the tree. The root signature and hash information on the nodes of the tree are included in each transmitted packet. The signing and verification operations are thus amortized over many packets, and the protocol operations are one to two orders of magnitude faster compared to individual packet signatures. Rohatgi has proposed a hybrid scheme [64] using off-line/on-line signature generation scheme for creating k-time public/private key pairs so that the cost of signature generation can be amortized over k signatures. The size of the keys is reduced by using hash functions with target collision resistance. The size overhead of the proposed scheme is however still considerable on a per-packet basis (of the order of 300 bytes per packet). Anderson et al have proposed the Guy Fawkes protocol in [65], which achieves source authentication using a small number of hash computations. In this protocol, the source selects a series of one-time secrets $X_0, X_1, X_2, ..$; the source commits to X_i in message M_{i-1} and reveals it in message M_{i+1}. The commitment for X_i has the form $h\left(M_{i+1}, h\left(X_{i+1}\right), X_i\right)$, while the first secret X_0 is committed by some external mechanism such as a conventional digital signature. In the Guy Fawkes protocol, the secrets are not related to one another, and the authentication mechanism cannot tolerate packet losses - if a commitment is lost, the corresponding secret cannot be authenticated. Perrig has proposed a broadcast authentication scheme named BiBa [66] which exploits the birthday paradox in trying to find two or more colliding hash computations on a given message, where the hash values are computed using a set of self authenticating values (SEALs) $s_1, ..., s_t$. The two or more SEALs for which the hash on the message collide form the signature. The scheme exploits the asymmetric property that the source has more SEALs than the adversary, and hence it can easily generate the BiBa signature with high probability. However, the adversary only knows the few SEALs disclosed by the source, and hence has a low probability of forging a valid BiBa signature. The algorithm is probabilistic in nature in the signature generation, and has a significant computation overhead at the source to find a valid signature. Also, the probability of an adversary to forge a signature increases with time as more and more SEALs are disclosed by the source.

Chapter 5

Efficient and Secure Multicast Routing in a Hybrid Satellite/Wireless Networks

5.1 Overview

Group communications is one of the most important applications in wireless network scenarios we consider. For example, applications like military operations or disaster relief require coordination amongst the users for correct operation. Group communications can be enabled efficiently by multicast routing. There have been several protocols proposed for multicast routing in wireless networks of mobile hosts - AMRoute [67], AMRIS [68], MAODV [69] and ODMRP [70], amongst others. All the protocols assume a flat network topology where the nodes are peers and routing paths from the sources to the receivers are available. If the network gets partitioned due to link breakages each partition forms its own multicast group. However, the proposed protocols are not efficient for the wireless hybrid network topologies we consider where the links are not uniform throughout and the nodes have different capabilities and connectivity. The networks are large in size and the multicast group members might be widely separated, even located in physically disconnected wireless LANs, as illustrated by the logical diagram of the hierarchical network topology in fig. 5.1. The gateway nodes offer multiple paths between different segments of the terrestrial network. The path through the satellite channel has

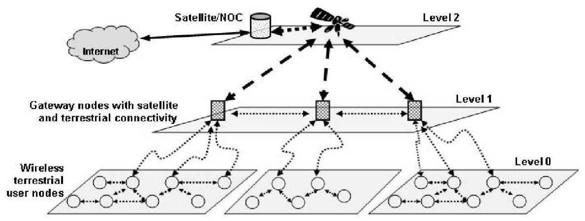

Satellite/NOC
Level 2
Internet
Gateway nodes with satellite and terrestrial connectivity
Level 1
Wireless terrestrial user nodes
Level 0

Figure 5.1: Hierarchy in the hybrid satellite network

high propagation delay and the routing protocol should consider this delay in establishing the multicast distribution tree. In the event of a network partition, there is no purpose for a partition to form its own multicast group if there are no sources within that partition. Rather, there should be a mechanism such that the partition can take steps to establish connectivity with the rest of the group. Therefore, attempting to implement one of the proposed “flat” wireless multicast protocols in this network will lead to large overhead in routing traffic, inefficient multicast distribution tree (for example, the root node is located in a part of the network far from the sources and destinations), constantly changing tree topology and packet drops for the multicast traffic.

Securing the multicast routing protocol in a wireless network is important since the network layer is susceptible to a number of attacks that threaten the correct functioning of the multicast routing protocol. For example, a malicious node can inject, modify or delete routing control packets to prevent the multicast routing protocol from establishing a correct distribution tree. An active attacker could also eavesdrop on the communication traffic attempting to read the multicast data or perform traffic analysis. Based on the multicast routing protocols for hybrid and wireless networks proposed in the literature, the active attacks on a routing protocol can be broadly classified into either *route dis-*

ruption attacks or *resource consumption* attacks. These attacks can be successful if node authentication and message integrity checks are not included in the control of the routing protocol. The following briefly summarizes a few well-known route disruption attacks.

- In protocols that use the hop count metric (number of hops from source to destination) to discover routing paths, a malicious node can modify the hop count value in the routing protocol control packet and set it to a lower value, to include itself in the distribution tree. Subsequently, the node might drop all the data traffic that it receives, thereby disrupting the communication. This is called a *blackhole* attack [71]. A special case of this is the *grayhole* attack, where the malicious node selectively drops some packets but not others, for example, dropping data packets while forwarding routing control packets correctly.

- For protocols that use *sequence numbers* to track the freshness of route discovery messages, a malicious node could set the sequence number in the route reply messages to a value higher than the actual, to redirect group traffic through itself. Subsequent correct route reply messages received by the source, with lower sequence numbers, would be discarded. The attack is similar to the blackhole or grayhole attacks described previously.

- Two or more malicious nodes operating in collusion could launch a *tunneling* or *wormhole* attack. Malicious node A would use pre-existing routing paths to unicast every control packet to malicious node B, thereby circumventing several hops, and vice versa. Thus the forwarding path that goes through the nodes A and B will appear to be several hops shorter than it actually is.

- On-demand routing protocols with duplicate suppression (e.g. AODV [72]) are vulnerable to the *rushing attack* [71]. An attacker can spread fabricated route request, route reply or other control messages quickly through the network, thus suppressing legitimate route request or other control messages because nodes will drop the packets received later due to duplicate suppression.

- A mobile attacker can move around the network and broadcast spoofed route reply messages as coming from legitimate nodes, such that a routing loop can form in the network [73].

- Wireless routing protocols that allow intermediate nodes to learn routes by sniffing packets in promiscuous mode (e.g. DSR [74]), are vulnerable to *route cache poisoning*. An attacker which is in the vicinity of nodes in the forwarding path to a destination *D*, can broadcast spoofed packets declaring (non-existent) routes to *D* through itself. Neighboring nodes may update their route caches with the wrong route on hearing the spoofed message.

- An attacker could send several route reply messages to the source with different spoofed addresses, making the source node believe that multiple paths exist to the destination.

Multicast routing protocols with no mechanism to control group membership at the network layer level are vulnerable to *resource consumption* attacks, some of which are briefly described below.

- A malicious node might inject fake data packets into the network, which will be forwarded to all group members along the distribution tree. This will consume network bandwidth and the transmission will waste energy of the forwarding nodes.

- Similar to the above, an attacker could inject fake control packets into the network at a high rate or flood the network with packets for non-existent groups, consuming network bandwidth in forwarding the packets along the distribution tree, and wasting energy at the forwarding nodes in processing the packet.

- Several malicious nodes might join the multicast group as receivers. This will cause the multicast distribution tree to be extended to the unauthorized receivers, and thus not reflect the correct group membership. Forwarding nodes will waste network bandwidth and their own energy in forwarding the data packets to the malicious receivers, who might simply drop the packets. This attack is also possible by a single attacker who spoofs multiple receiver addresses.

All the above attacks result in *denial of service* (DOS) to the legitimate nodes in distributing and receiving multicast data correctly in the network.

Therefore, the correct operation of the multicast routing protocol requires that the routing control messages be verifiable and the membership in the multicast group be tightly controlled. These can be achieved by the following security mechanisms:

- authentication of user nodes to join or leave a multicast group,

- authentication of in-tree multicast routers that forward multicast traffic meant for the group,

- authentication and message integrity check of the control messages required to maintain and update the multicast tree, and

- authentication of sender nodes before they are allowed to send data to the multicast group.

In this chapter, we describe the design of a secure multicast routing protocol for hybrid networks that takes into consideration the physical hierarchy in the network topology. We propose a hierarchical multicast routing protocol that takes into account the different performance characteristics of the wireless and the satellite channels, and minimizes the amount of routing traffic exchanged over the satellite links so that the delay overhead for the group creation and maintenance is minimized. The protocol also proposes a method to reconnect partitions to the rest of the group, instead of allowing the partitions to create new groups. Subsequently, we use the authentication algorithm proposed in chapter 4 to build a framework to secure the multicast routing protocol. The use of symmetric cryptographic primitives makes the proposed secure multicast protocol efficient in terms of node processing power, delay and energy consumption. The byte overhead of the proposed protocol is lower in comparison to using digital signatures, and therefore it can be used for securing the multicast routing protocol without introducing high control overhead for maintaining the multicast tree.

5.2 A Hierarchical Framework for Multicast Routing

We consider the hybrid network topology of figure 5.1 as the basis for designing our multicast routing protocol. Each wireless LAN has one or more stationary gateway. For our present design, we consider the case where there is one gateway node in each LAN.

A gateway node connects the local users to users in other LANs, and to the Internet, through links to other gateway nodes. The gateway nodes interconnect by both multi-hop terrestrial links, and one-hop bidirectional satellite links. The satellite has multiple spot-beams, an on-board router and is capable of on-board processing and switching between the LANs. We consider the case where a multicast group exists with group members (sources and receivers) spread across different LANs.

The physical network topology in figure 5.1 forms a logical hierarchy in three levels. At the lowest level are the user nodes grouped into terrestrial LANs. The user nodes route data traffic in a LAN using ad hoc routing protocol. The wireless user nodes have limited energy and processing power. At the second level are the gateway nodes, each of which connects the users in its local LAN to users in other LANs. We define each LAN as a *level-0* cluster with its local gateway node as the clusterhead. At the highest level is the satellite which connects all the LANs and has a global view of the network. We define the group of gateway nodes as forming a *level-1* cluster with the satellite as the clusterhead. We propose a hierarchical multicast routing framework that effectively and efficiently utilizes the hierarchy in the network topology. Our design minimizes the multicast routing overhead in the level-1 cluster to minimize the impact of the propagation delay of the satellite channels. We aim to minimize the control overhead for routing via satellite channels to limit network bandwidth expense. We also aim to re-connect partitions in an efficient way using the satellite backbone.

Multicast routing in a terrestrial LAN is mostly similar to the routing in a “flat” ad hoc network. The important difference is that in the hybrid network the satellite gateway always has to be in the multicast distribution tree of any multicast group that has members in its local LAN. We therefore base the multicast routing in the terrestrial LANs on an existing ad hoc multicast protocol that closely matches our design objectives, but modify the protocol to make efficient use of the hierarchy in the hybrid network. The selection of the base multicast routing protocol depends on the following design criteria:

- the protocol should require low state maintenance in the group members,
- the protocol should limit the flooding of control messages and redundant data transmissions,
- the protocol should be be loop-free, and
- the protocol should have at least one core node in the multicast group.

The multicast extension to Ad-hoc On-Demand Distance Vector routing protocol (MAODV)[69] satisfies the desired properties. We therefore adopt MAODV for multicast routing in the terrestrial segment, and modify it for our hierarchical topology. A description of the MAODV algorithm is found in [69]. In the following, we describe our hierarchical algorithm that utilizes and adapts MAODV messages for multicasting in the terrestrial network, and introduces new control messages for multicasting using the satellite overlay. For ease of understanding the similarities of our hierarchical multicast protocol with MAODV, we follow the description in [69].

The routing in the hierarchical multicast protocol can be segmented into two parts based on the logical hierarchy in the network: (a) multicast group formation within a terrestrial LAN, and (b) extending the multicast group across terrestrial LANs. We fuse the two segments by making use of core nodes, which are an extension of the multicast group leader concept defined in the MAODV specification. The following sections explain the role of the core nodes, and how the multicast group is formed within a LAN and across LANs.

5.2.1 Role of Multicast Group Leader

MAODV has a multicast group leader who is responsible for maintaining the multicast group sequence number (which is primarily used to eliminate stale routes) and disseminating this information to all the group members. MAODV specifies that the first member of a multicast group becomes the leader for the group, and remains in that position until it decides to leave the group, or until partition merging.

We look upon the group leader as the “core” of the multicast group, and propose additional functions for the group leader. In our design, the gateway node in a LAN is *always* the group leader of *all* multicast groups that are active in its LAN. Each LAN with members in the same group, has its own unique multicast group leader, namely, the local gateway. Thus in our design, a multicast group may have multiple peer group leaders. Since there is no longer a single group leader, we refer to them as the multicast group *core* nodes. In the hierarchical routing protocol, a core node is thus responsible for maintaining the local sequence numbers of all multicast groups active in its LAN, and periodically distributing this information to the local group members. In addition, a core node is responsible for periodically sending information about all local active groups to its peer core nodes. This mechanism is described later. Most of the modifications we make to MAODV are related to the choice of the core node and its capabilities.

5.2.2 Multicast Group Formation in a LAN

When a node has data to send to a multicast group, or it wishes to join the multicast group to receive data, it generates a Route Request message (RREQ) if it does not already have a route to the group. If the source node is aware of the address of the local core (gateway) node and it has a route to the core, it unicasts the RREQ to the core node. Otherwise, the RREQ is broadcast within the local network.

If the group is nascent, such that the multicast distribution tree is not yet formed, the RREQ message will eventually reach the core node, which replies with a Route Reply message (RREP) to the requesting node. (We do not consider the case of a network partition whereby the core node is unreachable from the requesting node.) Otherwise, if the multicast tree is already set up, response to the RREQ message follows standard MAODV algorithm. The local distribution tree is thus created with the gateway as the core. The core is also a member of the multicast group, and it thus receives the multicast data from its local sources.

5.2.2.1 Core Notification Message on Member Join/Leave

When a receiver nodes broadcasts a RREQ message to join a group, it can receive multiple RREP messages from nodes in different branches of the multicast tree. The new node joins the multicast group by unicasting a Multicast Activation message (MACT) to its selected upstream node in the multicast distribution tree [69]. The MACT message propagates up the distribution tree till it reaches the on-tree router that had originated the RREP selected by the joining node. On receiving the MACT message, the router realizes that a new branch has been grafted in the multicast tree, with itself at the root of the branch, and the joining node (whose address is obtained from the MACT message) located at the branch leaf. We propose that the router unicasts a $Join\ Notification$ message to the core node, informing the core of the address of the newly joining member. If the router is the root of multiple new branches, it can include all the addresses in the Join Notification message.

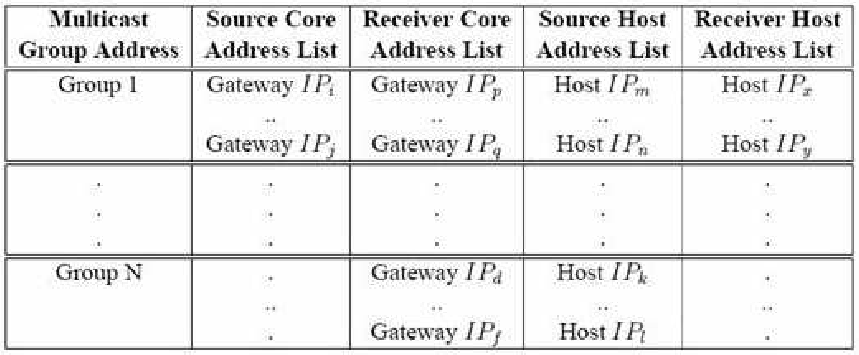
Similarly, when a group member leaves, it sends a MACT message to its upstream node with the $Prune$ flag set. The prune MACT message eventually reaches the root of the multicast branch, which unicasts a $Leave\ Notification$ message to the core informing it of the address of the leaving member.

The Join or Leave Notification message has no effect on the operation of the multicast routing protocol within the LAN, but it is required by the core node for efficient routing between LANs, as described in the following section.

5.2.3 Multicast Group Formation across LANs

Each node in the network that supports MAODV maintains three tables: (a) the *Route Table*, (b) the *Multicast Route Table* and (c) the *Request Table* [69]. We propose that in addition, each gateway node maintain a *Core Table* to share information about existing multicast groups with other LAN gateways in the network (figure 5.2). Each entry in the core table has the following information:

- Multicast group address
- List of *source* core addresses



Multicast Group Address	Source Core Address List	Receiver Core Address List	Source Host Address List	Receiver Host Address List
Group 1	Gateway IP_i .. Gateway IP_j	Gateway IP_p .. Gateway IP_q	Host IP_m .. Host IP_n	Host IP_x .. Host IP_y
. . .	. . .	. . .	. . .	. . .
Group N	. .. .	Gateway IP_d .. Gateway IP_f	Host IP_k .. Host IP_l	. .. .

Figure 5.2: Multicast Core Table maintained by each core (gateway) node

- List of *receiver* core addresses
- Local source addresses
- Local receiver addresses

We term a gateway node which has multicast sources in its local LAN as a source core, whereas a gateway node with multicast receivers in its local LAN is a receiver core. A gateway can thus be both a source core and receiver core for a group.

A core gets the addresses of its local sources either from the RREQ messages that it receives, or from the multicast data packet headers. The core gets the addresses of the local receivers either from RREQ messages, or from the join notification messages it receives from multicast tree members. A timer is associated with each active source; if no data is received from a source node within a timeout interval, the source node address is removed from the list of local source addresses. A receiver address is removed from the list of local receivers either when the core receives a MACT message with the *Prune* flag set, or when the core receives a leave notification message from a multicast tree member.

5.2.3.1 Core Hello Message

When a multicast group is created in a LAN, the local gateway node enters the information in its core table. If the group creation is driven by a source node RREQ, then the core enters its own IP address in the list of source core addresses, and the address of the source node in the list of local source addresses. The core subsequently broadcasts a Core Hello message to all the gateway nodes over both the satellite channel and the terrestrial links between the core nodes. The Core Hello message has the following fields:

$$\langle H\_flag, Broadcast\_ID, Source\_Addr, Num\_Groups, Group\_Addr \rangle \tag{5.1}$$

The H_flag is set for Core Hello messages. The broadcast ID is incremented by the core for each Core Hello message it sends out. The core places its own address in the $Source\_Addr$ field, and a list of the multicast group addresses for which it has local sources in the $Group\_Addr$ field. Num_Groups gives the number of group addresses present in the $Group\_Addr$ field.

The Core Hello message informs other gateways that the gateway that generated the message has sources in its local LAN for the multicast groups identified in the message. A gateway node receiving the message enters the information in its own core table. Corresponding to each multicast address in the message, it creates a new entry, with the address of the source gateway node added to the list of source core addresses corresponding to the new entry. A timer is associated with each source core address. A core with local sources periodically re-broadcasts the Core Hello message. Every reception of the Core Hello message by a gateway resets its local timer corresponding to that particular code node; if no message is received within a timeout period, the address of the source core node is removed from all entries in the multicast table that have the address. If the table entry becomes empty, then the entry itself is deleted. The Core Hello message also allows gateway nodes that have been recently activated to learn of all the active groups in the network.

5.2.3.2 Core Join Message

When a multicast group is created in a LAN with receivers, the local core updates the information for the group in its core table. It places its own IP address in the list of receiver core addresses for that group, and the addresses of all known local group members in the list of local receiver addresses. The core then sends a Core Join message to the source core nodes for the group. The addresses of the source core nodes are obtained from the core table. The Core Join message has the following fields:

$$\langle J\_flag, Broadcast\_ID, Source\_Addr, Num\_Groups, Group\_Addr \rangle \tag{5.2}$$

The J_flag is set for Core Join messages. The broadcast ID is incremented by the core for each Core Join message it sends out. The core places its own address in the $Source\_Addr$ field, and a list of the multicast group addresses for which it wants to receive data, in the $Group\_Addr$ field. Num_Groups gives the number of group addresses present in the $Group\_Addr$ field. When a source core node receives the Core Join message, it adds the address of the receiver core node to the list of receiver cores for each relevant multicast address entry in its core table, and starts sending the multicast data packets for the subscribed groups to the joining core. The joining core on receiving the packets forwards them to its local distribution tree.

5.2.3.3 Core Leave Message

When all the members of a multicast group in a LAN leaves the group, the local core updates the group entry in its core table. It deletes the local receiver addresses from the group entry. If it was receiving group data from remote LANs, it sends a Core Leave message to the remote source core nodes. If also there are no local sources for the group, then the core removes the group entry altogether from the table. The Core Leave message has the following fields:

$$\langle L\_flag, Broadcast\_ID, Source\_Addr, Num\_Groups, Group\_Addr \rangle \tag{5.3}$$

The L_flag is set for Core Leave messages. The broadcast ID is incremented by the core for each Core Leave message it sends out. The core places its own address in the $Source\_Addr$ field, and a list of the multicast group addresses which it wants to leave, in the $Group\_Addr$ field. Num_Groups gives the number of group addresses present in the $Group\_Addr$ field. When a source core node receives the Core Leave message, it removes the address of the receiver core node from the list of receiver cores for each relevant multicast address entry in its core table, and stops sending data for those groups to the leaving core node.

5.2.4 Multi-path Routing in the Satellite Overlay

A core node might have both local sources and local receivers for a multicast group, or only sources or receivers. It might also be receiving multicast group data from multiple remote source core nodes, and/or it might be sending group data to multiple remote receiver core nodes. Thus at the level of the core nodes, a multicast mesh is created,

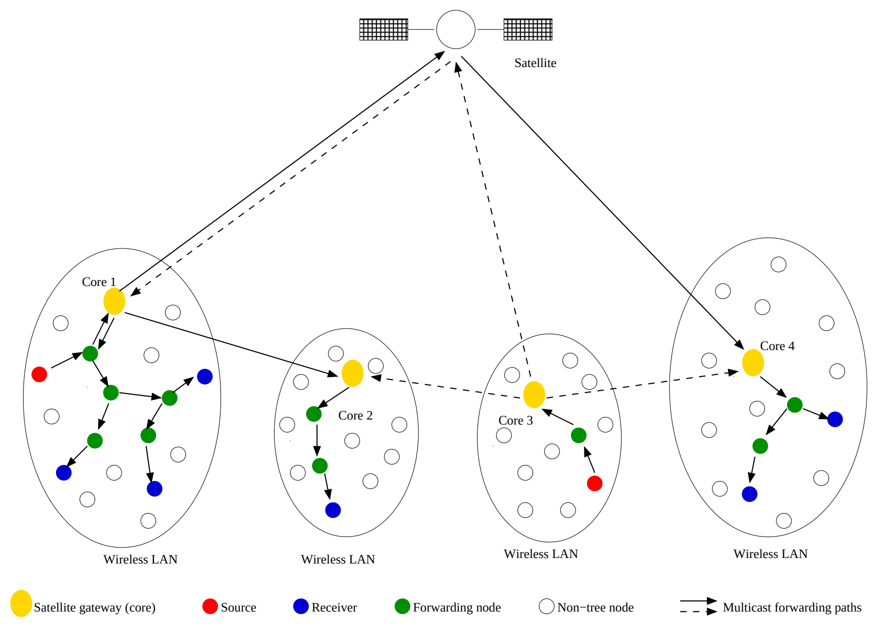


Figure 5.3: Multicast distribution tree in the wireless hybrid network

while each LAN has a multicast tree. Multicast routing across the hybrid network is thus achieved in a hierarchical manner, with a hybrid distribution method (figure 5.3).

The gateway nodes are interconnected by both terrestrial forwarding paths and the satellite links. In traditional wireless multicast routing protocols, the metric for selecting the forwarding path is usually shortest distance measured in number of hops. For the multiple paths between the gateway nodes, this metric will lead to the single-hop satellite link being selected in case the terrestrial forwarding path has two or more intermediate nodes. This will not reflect the fact that the satellite link might have more delay compared to a longer terrestrial path. On the other hand, the satellite links are more robust (the intermediate node is always available), can reach multiple other gateways in a single transmission, and offers very high bandwidth. Also, for terrestrial forwarding paths that are longer than a few hops, the delay in the two paths might be comparable.

We propose that a gateway node sends out the Core Hello on all its physical interfaces, both terrestrial and satellite. A gateway node that receives the message over both paths selects the one that arrives first (lower delay), and unicasts the Core Join message back to the source core using the selected path. In case no terrestrial path is available, the gateway will receive the Core Hello message from the satellite transmission. However, when a receiver core sends a Core Join or Core Leave message for multiple groups with several source cores, the receiver uses the satellite transmission to efficiently reach multiple core nodes simultaneously.

A source core sends multicast data traffic to other core nodes along all the interfaces at which it received Core Join messages. Therefore the data might follow both terrestrial and satellite paths. A receiver core can potentially receive the data from both paths - it forwards the data received on the interface on which it joined the group, and discards data received on the other interface. For high data-rate traffic, data forwarded along terrestrial paths might see more packet drops compared to the satellite channels. A receiver core that is receiving data on a terrestrial interface and suffers from packet drops, can switch to the satellite interface for better reception. For this, it first sends a Core Join message to the source cores over the satellite links, and then sends a Core Leave message over the terrestrial paths. The source cores will add the receiver core to the satellite forwarding path for the specific group (the $Broadcast\_ID$ of the Core Join message indicates its freshness), and delete the same receiver from the terrestrial interface. The intermediate nodes in the overlay also delete the corresponding forwarding path on receiving the Core Leave message. A receiver core node therefore has to maintain information for the multicast packets it receives on a per-group basis (to drop duplicates), and also maintain information on packet receive rates and drops for each interface.

The satellite links also provide robustness against failure of terrestrial forwarding nodes in the overlay. When one or more intermediate nodes in the forwarding path from a source core to a receiver core fail, the receiver core can still receive Core Hello messages from the source core over the satellite links, and it can subscribe to the group over the satellite forwarding path.

5.2.5 Dealing with Network Partitions

Due to node mobility or failure of intermediate nodes in the forwarding paths, a subset of group members in a LAN can become disconnected from the rest of the group. The MAODV protocol specifies that the multicast tree node downstream of the link break initiate a recovery and that the disconnected partition elect its own group leader, and the group continue to function independently in its partition till a merge with the rest of the network is possible. We propose that the partition re-establish connection to the network as a newly-joining LAN. This is possible only if there are one or more nodes in the partition with dual terrestrial/satellite connectivity. Therefore we propose that a subset of the normal user nodes have dual connectivity, with the satellite transmission being activated only in emergencies such as network partitions.

In our proposal, we deviate from the MAODV specification in the algorithm for selecting the group leader for the partition. A node can become the group leader only if it is both a group member and has dual connectivity. To discover such a node in the partition, the node (say A) downstream of the link break broadcasts a $Partition\ Leader$ message to the other nodes in the partition. Any node receiving the broadcast and satisfying the criteria for becoming a group leader, unicasts a reply to A. If A receives multiple replies, it selects the node with the lowest IP address and unicasts a $MACT$ message with the GL_flag set. As per the MAODV specification, the new group leader announces itself to all the nodes in its partition. In addition, the new group leader connects to the satellite, creates a Core Table and updates it with information for all the group members in its partition. If it has local sources, it broadcasts a Core Hello over the satellite links. If it has local receivers, when it receives a Core Hello for the group from other gateways, it sends a Core Join message in reply. A new branch of the multicast group is thus established in the partition, and it gets re-connected to the rest of the network.

In case none of the dual connectivity nodes present in the partition is a group member, then one of them joins the group on receiving the $Partition\ Leader$ message broadcast. Selection of the group leader then follows the method outlined above.

In the event of the partition re-connecting to the local LAN, we follow the MAODV specification. But the group leader is always selected as the original gateway node in the LAN, and the group leader of the partition ceases this function.

5.3 Methodology: Authenticated Multicast Routing using TESLA Certificates

We use our proposed approach for TESLA certificate authentication, along with public-key cryptography for bootstrapping, to design a framework for secure multicast routing in the hybrid terrestrial/satellite network. We use the hierarchical multicast routing protocol described in section 5.2 as the base routing protocol for the security framework.

5.3.1 Security Assumptions

We assume the terrestrial wireless channel protocol provides reliable transmission on a link-basis (for example, IEEE 802.11) and we do not consider the attacks that are possible against the link layer protocol. The satellite channel is also assumed to be always secure. The wireless links are bi-directional. A one-to-one mapping exists between the link layer address and the IP address of each node. All the nodes operate with omnidirectional antennas and the transmission of each node can be received by all its neighbors.

In the security framework, we propose to utilize the infrastructure in the hybrid network to build a certificate authority. We propose to use two levels of CAs. The GEO satellite is at the root of the hierarchy and it serves the role of a centralized CA that is universally trusted by all the nodes and gateways. Each gateway node acts as the CA for its local LAN, with appropriate delegation by the central CA so that the user nodes trust a gateway node for certification purposes. A local CA carries a certificate signed by the central CA which establishes its credentials to act as the local CA for the particular LAN. A user node can contact the local gateway for TESLA certification, instead of having to contact the central CA. The peer gateways in level-2 are can communicate directly with the central CA, and exchange their own node certificates signed by the central CA. We therefore make use of two different authentication algorithms in different levels in the network. In a terrestrial LAN, we use TESLA certificates for authentication, with the local gateway as the CA. At the higher level of the gateway nodes, we use traditional public-key certificates for authentication. The use of public key certificates in the higher level is acceptable since the gateway nodes have significant processing power, storage and energy. Also, the network bandwidth available in this level is high, provided by the broadband satellite channels. We propose to dedicate a small percentage of the satellite bandwidth as a "signaling" channel, which is to be used strictly for exchanging control messages for routing and security. In the following description of the operation of the secure multicast protocol, we assume that nodes do not move across LANs, but they can be mobile within their local LANs.

5.3.2 TESLA Certificate with Source/Receiver Information

A node initially obtains a TESLA certificate from the local CA before it can participate in any authenticated communication. If the node knows the address of the local CA, it can unicast a request for TESLA certificate to the CA, and include the anchor element of its authentication chain in the request. Otherwise, the node broadcasts a CA discovery message to its neighboring nodes. The CA discovery broadcast message propagates through the local network till it reaches a node that has the credential certificate for the local CA, which is signed by the central CA, or till the message reaches the local CA itself. In either case, the node with the CA certificate, or the CA itself, replies to the requesting node by sending the CA certificate, and the anchor element broadcast message from the CA, given in equation 4.5. The new node thus obtains the CA address and the anchor elements of the CA hash chains. In addition, the local CA periodically broadcasts its credentials and the signed anchor elements of its key hash chains to the local network.

In the request to the CA for a TESLA certificate, a node A also specifies the multicast group address G and whether it wants to join the group as a sender or a receiver. The CA includes this information in the TESLA certificate it generates for the node:

$$Cert_{CA}(A) = \big(ID_A, GRP_G, \{ID_A, GRP_G, \{src|rcv\}, s_{A,0}\}_{tK_{CA_1}}, t_0 + d,$$
$$MAC_{tK_{CA_1}}(..), SIGN_{-K_{CA}}(..)\big) \quad (5.4)$$

A general TESLA certificate, as specified in (4.7), is sent to A if the request does not specify the multicast group. The response from the CA to the requesting node also contains the current group sequence number, and the hop count of the CA from the requesting node. This information is needed for the correct operation of the MAODV protocol.

5.3.3 Hop Count Authentication Using Hash Chains

The mutable fields in MAODV control messages are (i) Hop_cnt and $Mgroup\_hop$ in RREP message, and (ii) Hop_cnt in the Group Hello message which is periodically broadcast by the group leader. These fields are set to 0 by the node generating the RREP message or the Group Hello message, and they are incremented by one by every node that forwards these messages. For a source node, the Hop_cnt field in the RREP message it receives, indicates the distance of the source node from the responding node that has a current route to the MAODV distribution tree. For a node joining the group as receiver, the $Mgroup\_hop$ in the RREP message it receives, indicates its distance from the root node of the local branch in the distribution tree. The Hop_cnt field in the Group Hello message indicates the distance in hops from the group leader.

To prevent intermediate nodes from claiming a lower hop count than the actual, we make use of one-way hash chains that are tied to the hop count value [75, 76]. A node A creating a RREP message (or the group leader/core node creating the Group Hello message) generates a one-way hash chain starting with a seed value h_n:

$$h_n \xrightarrow{F_1} h_{n-1} \xrightarrow{F_1} ... \xrightarrow{F_1} h_1 \xrightarrow{F_1} h_0 \tag{5.5}$$

Therefore ${F_1}^n\left(h_n\right) = h_0$ where n is the maximum number of hops in the network. A includes h_n in the RREP message, and the anchor element of the chain h_0, which is authenticated using A's TESLA key. The Hop_cnt and $Mgroup\_hop$ fields are set to 0. A neighbor node B receiving the RREP, computes $h_n \xrightarrow{F_1} h_{n-1}$, and re-transmits the RREP message with Hop_cnt and $Mgroup\_hop$ incremented by 1 and h_n replaced by h_{n-1}. An intermediate node C that is 1 hop away from B can check the accuracy of the Hop_Cnt and $Mgroup\_hop$ fields in the RREP it receives as follows:

$$\begin{gathered} {F_1}^{n-Hop\_cnt}\left(h_{n-1}\right) \overset{?}{=} h_0 \\ {F_1}^{n-Mgroup\_hop}\left(h_{n-1}\right) \overset{?}{=} h_0 \end{gathered} \tag{5.6}$$

The use of one-way hash chains to secure the hop count still allows a node to claim to be one hop closer to the group leader than its actual distance, if it simply forwards to its downstream node the hop count chain element that it obtained from its upstream node [77]. In addition, this method is vulnerable to attacks by a mobile attacker operating in promiscuous mode, unless the hash chain element is encrypted. A mobile attacker can move several hops upstream and sniff the control messages as they are broadcast downstream by a forwarding node closer to the source, and thus obtain a hash chain element associated with a lower hop count value. The attacker can then move downstream and use this hash chain element to advertise a false path with lower hop count to the source or multicast tree, thereby potentially drawing all forwarding paths in its vicinity to itself, and thus making sure that all the traffic in its neighborhood flows through it.

5.3.4 Secure Multicast Protocol Operation

5.3.4.1 Route discovery and establishment

When a user node A wants to find a route to a multicast group G in the interval $< t_i, t_i + d >$, it broadcasts a authenticated RREQ message to the network. A computes a MAC on the RREQ message using a fresh (undisclosed) key from its TESLA key chain, and includes the MAC along with $Cert_{CA}(A)$ with the RREQ message.

Any intermediate node B receiving the RREQ message compares the destination address in the RREQ message to the group address in the certificate. If the two do not match, the RREQ is discarded. Otherwise, B buffers the RREQ message along with the certificate, and processes the message. B computes a MAC on the RREQ message using its most current (undisclosed) TESLA key, and forwards the message with the MAC and its TESLA certificate. Each subsequent node (say C) that receives the RREQ stores the RREQ message with B's MAC and TESLA certificate. C then replaces B's MAC and TESLA certificate with its own MAC and TESLA certificate and forwards the message. The forwarding nodes also set up reverse path pointers to node A according to MAODV operation. (The above assumes that two neighboring forwarding nodes B and C are communicating for the first time and hence B needs to send its TESLA certificate to C. If B previously had authenticated communication with C, it does not need to send the certificate since C already has an authenticated element of B's TESLA key chain.)

When the RREQ message reaches a node (say D) that has a fresh enough route to the group (as determined by the sequence number in the RREQ message), the processing depends on A's intended role in the group, and whether D is a router in the multicast tree, or merely a node with a current route to the tree.

(i) If A is a source node for G, D does not need to be an on-tree router. D creates a RREP message to be sent to A and includes its hop count authenticator H_n (which in this case indicates D's distance n in hops from the multicast tree) and the anchor element H_0 for the hop count in the RREP message. D computes a MAC on the RREP message and sends it along with its TESLA certificate to A in the reverse path. Intermediate nodes receiving the RREP message increment the hop count and accordingly modify the hop count authenticator in each step. The forwarding nodes also store the RREP message and the TESLA certificate of D and the preceding node who had forwarded the RREP. The *next hop* field in the route table entry for this path is not enabled till the stored messages are authenticated. Figure 5.4 shows the message exchanges that take place.

At time $t_i + d$, the local CA broadcasts the TESLA key it has used for the certificates generated in $< t_i, t_i + d >$. A, D and the intermediate nodes in the forwarding path can subsequently verify the TESLA certificates they have cached. If A, D and any intermediate node are using fresh TESLA key chains to authenticate their MACs in the above route discovery process, then the key used to authenticate the MAC is obtained from the TESLA certificate itself, and the MACs can be immediately verified. Otherwise, the CA

$$
\boxed{
\begin{array}{l}
\quad A \rightarrow *\colon RREQ, MAC_{s'_{A,j}}\left(RREQ\right), Cert_{CA}\left(A\right) \\
B \rightarrow *\colon \langle RREQ, MAC_{s'_{A,j}}\left(RREQ\right), Cert_{CA}\left(A\right)\rangle, MAC_{s'_{B,j}}\left(RREQ\right), Cert_{CA}\left(B\right) \\
C \rightarrow *\colon \langle RREQ, MAC_{s'_{A,j}}\left(RREQ\right), Cert_{CA}\left(A\right)\rangle, MAC_{s'_{C,j}}\left(RREQ\right), Cert_{CA}\left(C\right) \\
D \rightarrow C\colon \langle RREP, n, H_0, 0, H_n\rangle MAC_{s'_{D,j}}\left(..\right), Cert_{CA}\left(D\right) \\
C \rightarrow B\colon \langle\Big(RREP, n, H_0, 0, H_n, MAC_{s'_{D,j}}\left(..\right), Cert_{CA}\left(D\right)\Big), 1, H_{n-1}\rangle, MAC_{s'_{C,j}}\left(..\right), Cert_{CA}\left(C\right) \\
B \rightarrow A\colon \langle\Big(RREP, n, H_0, 0, H_n, MAC_{s'_{D,j}}\left(..\right), Cert_{CA}\left(D\right)\Big), 2, H_{n-2}\rangle, MAC_{s'_{B,j}}\left(..\right), Cert_{CA}\left(B\right)
\end{array}
}
$$

Figure 5.4: Message exchange for authenticated route discovery using TESLA certificate algorithm. A is the source and D is a node that responds to A with a fresh route to the group. B and C are intermediate nodes.

broadcasts the TESLA key the node has used to compute its MAC in $< t_i, t_i + d >$. Intermediate nodes can subsequently verify the MACs of their upstream or downstream neighbors. The TESLA key $s_{D,i}$ used by node D for the RREP message is received by A from the CA broadcast who can then verify the RREP message created by D. If the MAC verifications are successful, the $hopcount$ field is enabled in the route table entry and the forwarding path from A to D becomes active. If any node cannot verify the MAC from its neighbor (or A or D cannot verify their MACs), the forwarding path is canceled and the node which noticed the error sends an error message to A.

(ii) If A wants to join the multicast group as a receiver, D has to be either a group member or an on-tree router so that it can reply with an RREP message. D creates an RREP message to be sent to A and includes its hop count authenticator h_{D_n} and the anchor element h_{D_0} for the hop count in the RREP message. The hop count authenticator in this case authenticates both the Hop_Cnt and $Mgroup\_hop$ fields in the RREP message, which are both set to 0 by D. D computes a MAC on the RREP message and sends it along with its TESLA certificate to A in the reverse path. D also creates both a route

$$\begin{aligned} A \rightarrow B &: MACT, MAC_{s_{A,j+1}}(MACT) \\ B \rightarrow C &: \langle MACT, MAC_{s_{A,j+1}}(MACT)\rangle, MAC_{s_{B,j+1}}(..) \\ C \rightarrow D &: \langle MACT, MAC_{s_{A,j+1}}(MACT)\rangle, MAC_{s_{C,j+1}}(..) \end{aligned}$$

Figure 5.5: Authenticated MACT message for activating the chosen route from A to D.

table entry and a multicast route table entry for A, but does not activate them. Intermediate nodes receiving the RREP message authenticate it similar to the RREQ message, incrementing the hop count and accordingly modifying the hop count authenticator in each step. The forwarding nodes also store the RREP message and the TESLA certificate of D and the node who had forwarded the RREP. The $next\ hop$ field in the route table entry and multicast route table entry for A is not enabled till the stored messages can be authenticated and a MACT message is received from A.

At time t_i+d, the local CA broadcasts the TESLA key it has used for the certificates generated in $< t_i, t_i + d >$. A, D and the intermediate nodes in the forwarding path can subsequently verify the TESLA certificates and the MACs of the control messages that they have cached, as described in the previous section. The joining node A might have received multiple RREP messages along different paths to the multicast tree. It selects one of the paths according to MAODV operation and the selected path must have been successfully authenticated. A generates a MACT message and authenticates it with a MAC computed using the next undisclosed key from its key chain. A then unicasts the authenticated MACT message to the selected next hop. The MACT message propagates upstream, with each forwarding node authenticating it with a MAC computed using the

$$
\begin{aligned}
A \rightarrow B &: MACT_{Prune}, MAC_{s'_{A,j+k}}(MACT_{Prune}) \\
B \rightarrow C &: \langle MACT_{Prune}, MAC_{s'_{A,j+k}}(MACT_{Prune})\rangle, MAC_{s'_{B,j+k}}(..) \\
C \rightarrow D &: \langle MACT_{Prune}, MAC_{s'_{A,j+k}}(MACT_{Prune})\rangle, MAC_{s'_{C,j+k}}(..)
\end{aligned}
$$

Figure 5.6: Authenticated $MACT\ Prune$ message for deleting the multicast branch from node D to A. Here A is a group receiver, and D is the root of the branch with A as leaf node.

next undisclosed key in its key sequence, till the message reaches the on-tree router. At the next time interval of key disclosure, the TESLA keys used to compute the MACs are disclosed and subsequently the MACs can be authenticated. If the authentications are successful, each node enables the corresponding entry in its multicast routing table and the new branch of the multicast distribution tree is activated. Figure 5.5 shows the authenticated MACT message propagating upstream from A to D.

5.3.4.2 Multicast tree branch pruning

A group member or a router A can leave the multicast tree only if it is a leaf node. To do so, A unicasts a MACT message with the P_flag (Prune flag) set, to its upstream neighbor on the tree. The MACT message is authenticated with a MAC computed using the most recent undisclosed key from A's key chain. The next hop processes the message only if A is its downstream neighbor (which is established during the route discovery process). The next hop buffers the MACT message till the next interval when A's MAC key is disclosed by the CA broadcast. If the MACT message is subsequently authenticated, the next hop deletes the entry for A from its multicast route table, and sends a authenticated

$$Core \rightarrow *\colon \langle GRP\_HELLO, N, H_0, 0, H_N \rangle MAC_{s'_{Core,j}}(..)$$
$$B \rightarrow *\colon \langle \Big(GRP\_HELLO, N, H_0, 0, H_N, MAC_{s'_{Core,j}}(..) \Big), 1, H_{N-1} \rangle, MAC_{s'_{B,j}}(..)$$

Figure 5.7: Authenticated Group Hello message broadcast from the core node. Here N is the diameter of the network, and node B is one-hop away from the core.

notification to the core that group member A has left. The next hop can similarly prune itself from the tree if it is not a group member and A's leaving makes it a leaf node. The process continues recursively till a current group member or a non-leaf router is reached. Figure 5.6 shows the authenticated $MACT\ Prune$ message propagating upstream from leaf node A to branch root node D.

5.3.4.3 Group Hello messages

The core node periodically broadcasts Group Hello messages with the latest sequence number for the group. The Hop_Cnt field in the Group Hello message is authenticated using a hop count authenticator. The core includes the hop count authenticator and the hop count anchor element with the Group Hello message. The message is also authenticated by a MAC which is computed using the next undisclosed key from the TESLA key chain of the core. Nodes receiving the Group Hello message update the hop count and the hop count authenticator before forwarding the message. Each node buffers the message till the next interval when the core TESLA key is disclosed, when the message can be authenticated. Figure 5.7 shows the authenticated Group Hello message being forwarded by node B one-hop away from the core node.

5.3.4.4 Authenticating Core Hello, Core Join and Core Leave messages

At the upper level of the peer core nodes, the multicast routing messages are authenticated using public-key-based digital signatures. Any core node sending a routing message to another core node verifies the message by computing a digital signature on the message, and also includes its public-key certificate generated by the top-level CA, namely, the satellite.

5.4 Security Analysis of the Routing Protocol

5.4.1 *Active-y-x* Attacker Model

We consider an active attacker model. The attacker may attempt to compromise the correct operation of the routing protocol by exhibiting arbitrary, *Byzantine* [78] behavior. The attacker may try to create, modify, drop or replay routing packets, as described in section 4.1, and in general cannot be expected to behave correctly with respect to the routing protocol functionality. Based on the attacker model in [71], we characterize the attacker on the basis of the number of malicious nodes it owns and the number of good nodes it has compromised. We denote such an attacker as *Active-n-m* attacker, where *m* is the number of malicious nodes the attacker owns and *n* is the number of good nodes it has compromised. We assume the attacker owns all the cryptographic keys of the nodes it has compromised and can distribute them amongst all its nodes. Classified in the order of increasing attack capability, the attacker hierarchy is: Active-0-1 (attacker own one malicious node and has compromised no good nodes), Active-0-*x* (attacker owns *x* malicious nodes), Active-1-*x* (attacker owns *x* nodes and has compromised 1 node) and Active-*y*-*x*. In addition, we define an Active-VC attacker as an attacker that controls all the nodes in a vertex cut of the network, such that the network is partitioned into disjoint sections and the good nodes in one section has to go through the attacker nodes to communicate with good ones in another section. We also make the assumption that the attacker can own only the lowest-level terrestrial nodes in the hybrid network; the network nodes of the higher level have more security and behave correctly. This includes the satellite and the gateways.

5.4.2 Security Against Active Attacks

Based on the attacker model above, the multicast routing protocol described above is secure against several attacks mentioned in section 4.1.

1. Authentication of the hop count using one-way hash chains in RREP or Group Hello messages prevents a malicious forwarding node from claiming a shorter distance than the actual to the multicast tree member or the core node, respectively. This prevents blackhole or grayhole attacks. However, the malicious node can claim the same distance as the previous node from which it received the RREP or Group Hello message. Also, the attack where a malicious node increases the hop count more than the actual value, is also not prevented. The effect of this attack is similar to the case where the malicious node simply drops the packets instead of forwarding them.

2. Authentication of the RREQ or RREP messages by the generating node using

TESLA key-based MACs prevents intermediate nodes from modifying the sequence number in the control messages. This prevents the attack where a malicious forwarding node sets the sequence number to a value higher than the actual to attract routes to itself.

3. The use of TESLA certificate to specify whether a node wants to join the group as source or receiver prevents unauthorized nodes from being able to find a path to the multicast tree to send messages or to graft new branches on the tree to receive messages. Thus the on-tree nodes are spared from unnecessary forwarding of data packets and wasting their energy in doing so.

4. The secure multicast protocol assumes that the CA generates TESLA certificates for authorized nodes only. We also make the assumption that the CA does not generate more than one TESLA certificate for a node. If legitimate nodes behave correctly, then attacks based on spoofing control messages are prevented, since any unauthorized node that tries to send spoofed packets will not have a correct TESLA certificate to authenticate the packets it generates. Therefore the rushing attack, route cache poisoning attack and creating routing loops are prevented. The attacker also cannot spoof multiple paths to the multicast tree, since it would be unable to obtain multiple TESLA certificates for the different identities required to launch this attack.

   Due to the TESLA key disclosure delay involved in authenticating control messages in our protocol, an attacker could initially succeed in having fake control packets forwarded by legitimate nodes. However, once the TESLA keys are disclosed and the attacker's control packets fail authentication, legitimate nodes in the attacker's neighborhood would refuse to forward packets from the attacker.

The secure multicast routing protocol thus protects against *Active-0-1* and *Active-0-x* attackers. It also prevents several attacks by *Active-1-x* and *Active-y-x* attackers, though not all the attacks are prevented. For example, if the attacker can compromise one legitimate node with valid TESLA certificate, it can use the compromised node to send spoofed control messages. Multiple compromised nodes can be used to launch tunneling attacks or network-partitioning attacks. However, if the network is able to detect the compromised nodes, then the CA can revoke their certificates to prevent future instances of these attacks.

5.5 Summary

In this chapter, we have described the design of a secure multicast routing protocol for hybrid networks. We have discussed the importance of group communication in hybrid networks and why the group communication in the wireless environment in especially susceptible to attacks by malicious adversaries. We have described a range of attacks that are possible on the routing protocol that forms the underlying construct for enabling group communication. For the design of secure multicast protocol, we have proposed a hierarchical multicast routing protocol, based on MAODV, that takes into consideration the physical hierarchy in the network topology. The proposed multicast protocol minimizes the amount of routing traffic exchanged over the satellite links so that the delay overhead for the group creation and maintenance is minimized. The protocol also incorporates a method to reconnect partitions to the rest of the group, instead of allowing the partitions to create new groups. We combine the authentication algorithm proposed in chapter 4 with the multicast routing protocol to build a secure multicast routing framework. The control messages in the secure multicast protocol are authenticated using TESLA certificates and symmetric MACs. The use of symmetric cryptographic primitives makes the secure multicast protocol efficient in terms of node processing power, delay and energy consumption. The byte overhead of the proposed protocol is lower in comparison to using digital signatures, and therefore it does not have high control overhead for maintaining the multicast tree.

5.6 Related Work

Several protocols have been proposed for multicast routing in wireless ad hoc networks, for example, AMRoute [67], ODMRP [70], MAODV [69]. All the protocols assume a flat network topology and uniform wireless links.

There have been relatively fewer proposals for hierarchical multicast routing in wireless networks. Gui and Mohapatra [79] have proposed two methods to construct a hierarchical multicast tree - *domain-based* and *overlay driven*. The authors use Differential Destination Multicast (DDM) [80] as the base routing protocol, and enhances it by adding an overlay of root nodes, thus creating the Hierarchical DDM protocol. The hierarchy in the proposed approaches is a logical hierarchy, which is created by dividing a flat ad hoc network into clusters with root nodes. The use of DDM limits the scalability of their protocol since DDM uses *source routes* to multicast the data to the group members. Ko et al [81] have proposed PHAM (Physical Hierarchy-driven Ad Hoc Multicast), which is a multicast routing protocol for mobile ad hoc networks that have a physically hierarchical architecture, for example, low-powered nodes grouped into clusters with more powerful clusterheads. The authors describe the operation of PHAM in a two-level hierarchy with level-1 nodes in each physical group managed by a dedicated level-2 super-node with higher capabilities. In their proposal, a super-node is knowledgeable of all the level-1 nodes in its group, and the level-2 super-nodes are inter-connected via tunnels. In PHAM, control packets are flooded in the multicast group. Source nodes always register with the super-node, but receiver nodes do not need to do so. A multicast group in a physical group might not include the local super-node. When a super-node in another group receives data packets, it floods the packets to the local physical group. The authors simulate the PHAM framework using ADMR (Adaptive Demand-driven Multicast Routing) [82] in each physical group, and the performance evaluation shows higher throughput, more efficient use of control packets and shorter latency for PHAM compared to pure ADMR protocol. In [83], the authors propose an integrated architecture for multicast routing in cellular networks, using ad hoc relays for the last mile to the node with the lowest throughput. The paper characterizes the relay capability of IEEE 802.11b ad-hoc network for multicast traffic, and based on a general interference model, derives a polynomial-time 4-approximation algorithm for constructing an optimal-throughput multicast forest.

In the area of secure routing for wireless networks, several protocols have been proposed for secure unicast routing in ad hoc networks. Most of these protocols, for example, ARIADNE [71], SEAD [77], ARAN [73], SAR [84] and SAODV [75], have focused on adding security mechanisms to previously proposed unicast protocols AODV or DSR to remove security vulnerabilities that are present in the original specification of the routing protocols. SRP [85] has been designed as a unicast routing protocol with integrated security mechanisms.

Roy et al. [76] have proposed security additions to the MAODV protocol to protect the multicast routing against attacks. This is one of the first proposals for secure multicast routing in ad hoc networks. The paper proposes the use of public key cryptosystem to authenticate node identities and routing control messages. Group members and non-members who act as in-tree routers need authorization in the form of digital certificates (called *node certificate*) signed by a CA. Only nodes that possess a node certificate can take part in routing. A group member needs an additional *group membership certificate* to to prove that it belongs to a particular multicast group. This certificate binds the group member's public key to its IP address and the multicast group IP address. All nodes that are in the MAODV multicast tree have an additional credential called the *tree key*, to distinguish them from non-tree nodes in the network. Also, a node on the multicast tree establishes *pairwise shared keys* with each of its immediate neighbors. All messages exchanged between neighboring nodes include a MAC computed using the shared key to provide strong source authentication and to prevent impersonation attacks. The pairwise keys are also used to securely disseminate the tree key amongst the tree members along the multicast tree. To prevent modification of the hop count in the routing control messages, the authors propose using *one-way hash chains* to securely bind the hop count to elements of the hash chain [75]. The group leader generates the hop count hash chain, and securely disseminates the hop count authenticator (the anchor element of the chain) to all the group members along with the tree key. Using these cryptographic mechanisms, all the control messages in MAODV are secured. A node S joining the multicast group signs its route request RREQ message using its public key. Each intermediate forwarding node adds its own signature to the RREQ message, replacing the signature of the previous forwarding node, in a technique similar to ARAN [73]. The route reply RREP message from an in-tree node is similarly authenticated as it travels in the reverse path to S. S includes a hash computed over its ID and the tree key when it sends the subsequent link activation (MACT) message to prove it was the node that had initiated the route discovery. Branch pruning messages are authenticated by a MAC computed using the pairwise key shared between a node and its upstream neighbor, to prevent impersonation attacks.

The secure multicast protocol design in [76] helps to prevent several attacks against MAODV, but it also suffers from several limitations. The use of public key cryptography is expensive in terms of byte overhead, computation and energy expenditure. As the simulations in the paper illustrate, the security additions increase the byte overhead of MAODV by 3.5 to 4.2 times, and the overhead increases rapidly with node mobility. The use of public key cryptography actually makes the network vulnerable to additional DoS attacks - an attacker can send spurious control messages with fabricated signatures at a very high rate, and the good nodes will have to spend considerable processing power verifying the messages, thereby eating up CPU computational cycles and depleting the node energy rapidly. Flooding the network with fake signatures also increases the chance of network congestion due to the additional byte overhead. These problems are not limited to [76] alone, but is applicable to all secure ad hoc routing protocols that rely on public-key cryptography (e.g.[73]). The method of distributing the tree key in [76] suffers from the cyclic redundancy problem [86], since it initially uses the multicast tree to distribute the key. Apart from the byte overhead, the protocol in [76] is also not efficient. The use of pairwise neighbor keys is superfluous when public key cryptosystem is used, since digital signatures can prevent impersonation attacks.

For security in satellite networks, Howarth et al. [87] have looked at the problem of data encryption for large multicast groups with high member dynamics in satellite networks. The authors propose performing encryption of the traffic at the application level, by using Logical Key Hierarchy [88] integrated with multi-layer IPSEC [89]. The solution does not deal with security of the routing protocol at all. In a related work, Cruickshank et al. [90] have proposed modifications to the Digital Video Broadcasting Return Channel System (DVB-RCS) [91] to provide secure multicast services over satellites. The solutions deal with providing traffic encryption at the link layer with different keys for different multicast groups. Multicast routing at the network layer is not considered. Duquerroy et al. [92] have proposed “SatIPSec”, for key distribution and secure communication for both unicast and multicast in a satellite network.The solution is based on IPSEC [93], with the addition of the Flat Multicast Key Exchange (FMKE) to support key management for secure group communication. This proposal is also meant for encryption of the data traffic, and does not consider security of the routing protocol. Kong et al. have designed and evaluated a security framework for multi-level wireless networks with unmanned aerial vehicles (UAVs) in [94]. The paper provides a framework for the security services by using both symmetric key cryptosystems and public key cryptosystems. The design is aimed at securing unicast data traffic and does not consider the security of multicast routing.

Chapter 6

Performance-aware Security of Unicast Communications in Hybrid Satellite Networks

6.1 Overview

6.1.1 Use of Performance Enhancing Proxy Server (PEP) and HTTP Proxy Server in Satellite Networks

Satellite links suffer from longer propagation delays compared to terrestrial links. The delay can be as high as 500ms round-trip for a geostationary satellite link. Most of the Internet traffic uses the Transmission Control Protocol (TCP), which is highly susceptible to the delay-bandwidth product and exhibits very poor performance in satellite channels. Satellite TCP connections need large transmit windows to fully utilize the available bandwidth. However, due to the TCP *slow start algorithm* and large propagation delay in the satellite channel, it takes much longer for satellite TCP connections to reach the target window size, in comparison to terrestrial TCP connections. Also, the window is very vulnerable to congestion due to the multiplicative decrease strategy of TCP. The problem is compounded by the fact that TCP misinterprets link layer corruption (which is the prevalent source of loss in satellite links) as congestion (which is rare) and consequently reduces the window.

To mitigate the negative effects of the satellite propagation delay on Internet traffic, commercial satellite networks usually implement a split-connection TCP Performance Enhancing Proxy (PEP) [95]. A PEP agent is installed at the satellite gateway between the satellite network and the Internet. The PEP agent inspects every TCP packet that flows through the network. For data packets, the PEP sends back premature acknowledgments to the TCP senders, without waiting for the TCP segments to be actually delivered to the receivers. These premature acknowledgments are specially formatted to be indistinguishable from real acknowledgments and they considerably shorten the perceived round-trip delay. Studies have shown that this technique leads to significant performance improvement in satellite networks [96, 97, 98]. TCP PEPs have hence been widely deployed in satellite networks today.

Commercial satellite networks also employ an HTTP proxy server to improve the speed of response to web browsing requests for Internet traffic. When a user browses content on the Internet, the application layer protocol in use is the Hyper Text Transfer Protocol (HTTP). A typical HTTP exchange involves a request by the browser for a webpage ("GET"), and a response from the web server, which contains the HyperText Markup Language (HTML) text of the requested webpage. A typical HTML page contains multiple embedded "objects" such as images, embedded media or scripts, etc. Each embedded object has to be retrieved with a separate HTTP request-and-response exchange. Therefore, a webpage that contains $n - 1$ embedded objects takes $n * RTT$ time to load fully, where RTT is one round-trip time. This can be extremely costly in the satellite network due to the high RTT.

The HTTP proxy server (also known by various other names depending on the vendor) is implemented in satellite networks to get over this problem. In a typical implementation, this requires a local web proxy server at each user location, and a remote proxy server at the central hub facility of the satellite network, i.e., the NOC. The web browser at the user location should be able to recognize the local proxy (which can be either software on the client machine, or a separate hardware connected in between the client machine and the local satellite terminal). When the browser makes a request for a webpage, the HTTP GET request is sent to the local web proxy, which forwards the request to the destination web server. The web server responds with the requested base HTML page. This page is intercepted by the proxy server at the network hub facility. The hub proxy server reads the base HTML page and sends multiple GET requests to the destination web server for all the embedded objects in the base HTML page. This exchange occurs over a high-speed terrestrial connection between the hub and the Internet, thereby saving the time each request would have needed for a round trip over the satellite link. As the objects of the web page are retrieved by the hub, they are immediately forwarded to the proxy at the user location. As the user browser receives the base HTML documents, it generates appropriate GET requests to fetch the objects corresponding to the links embedded in the document. The browser GET requests are terminated at the local web proxy server, which forwards the pre-fetched documents to the user browser immediately. The net result is that only a single GET request from the user browser traverses the satellite link, while a set of rapid responses quickly deliver the requested webpage and associated elements to the browser. The need for satellite capacity is also reduced, which is the most costly element of a satellite network.

In figure 6.1, the proxy server at the user represents both the PEP (user side) and the HTTP proxy (user side). There is a hub proxy server located at the NOC with the

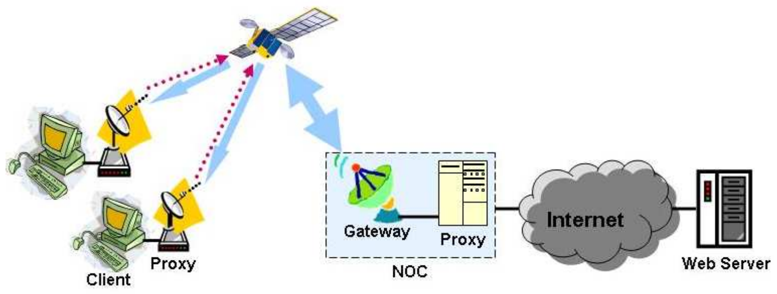


Figure 6.1: Proxy server placement in a commercial satellite network

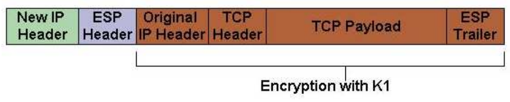
hub satellite gateway - this proxy server represents the gateway proxy for both TCP and HTTP performance enhancements.

6.1.2 Proxy Performance Problem with Unicast Communication Security

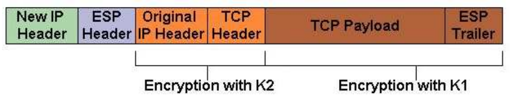
Two protocols that are widely used for secure unicast communication are the Internet Security Protocol (IPSEC) [93] and the Secure Socket Layer (SSL) protocol [99].

IPSEC is a network layer security protocol that performs cryptographic operations on the IP packets. IPSEC creates an end-to-end *secure channel* at the network layer for the secure transfer of traffic between two end users. The two end points in the communication negotiate security parameters known as *Security Association* (SA) before traffic can be encrypted. Once the SA has been established in the handshake phase, the IP packets are encrypted using the algorithms and the keys specified in the SA. This is done when IP Encrypted Security Payload (IPSEC ESP) [100] is used. IPSEC ESP provides for both data encryption and authentication. There is another variant called IP Authentication Header (IPSEC AH) [101] which provides only authentication, but no encryption. In our discussion, we will imply IPSEC ESP whenever IPSEC is mentioned.

The problem with using IPSEC in satellite networks is that it disables the functionality of the PEPs. The IP packet payload, which includes the TCP header, is encrypted with keys known only to the end points (figure 6.2(a)). Therefore a TCP PEP, which is an intermediate node in the communication path, cannot read or modify the TCP header, since the PEP does not know the keys. Consequently the PEP cannot function, leading to a degradation in the performance of the TCP protocol. The HTTP proxy also cannot function when IPSEC ESP is used. Since the HTML page is encrypted end-to-end, the HTTP proxy cannot read the webpage to pre-fetch the embedded objects. Therefore use of IPSEC leads to a degradation in performance for both the TCP PEP and HTTP proxy.



(a) Original IPSEC ESP tunnel mode encryption



(b) Layered IPSEC ESP tunnel mode encryption

Figure 6.2: Packet format for IPSEC and Layered IPSEC encryption. Key $K1$ is shared between end-points only. Key $K2$ is shared between end-points and TCP PEPs.

The Secure Socket Layer (SSL), on the other hand, operates above the transport layer in the protocol stack and establishes a secure session for web browsing on a need basis. The resulting connection is known as *secure HTTP*, or HTTPS, and it encrypts the application layer HTTP data end-to-end between the client and the server. SSL works only with a connection-oriented transport protocol like TCP. SSL encrypts the TCP payload between the client and the server, but the TCP header is transmitted in the clear.

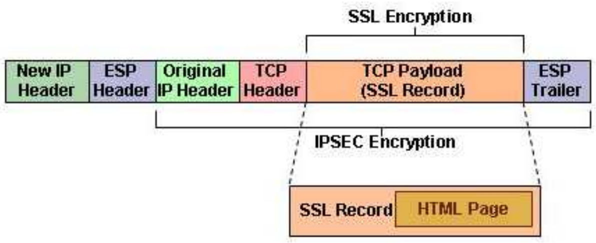


Figure 6.3: IPSEC and SSL encryption on a packet

Therefore the TCP PEPs can function correctly with SSL. However, SSL encryption does not allow the HTTP proxy to function correctly. The HTML webpage encrypted into SSL records are readable only by the client and the server who have the decryption keys. The keys are not available to the proxy, and therefore the proxy cannot read the HTML webpage. Consequently, the hub proxy server cannot send requests to the web server for the embedded objects in the page and therefore HTML object pre-fetching cannot take place. The net result is that a web page with $n-1$ embedded objects takes $n * RTT$ to get loaded, an increase in delay by a factor of n. Fig. 6.3 illustrates the encryption regions of SSL and IPSEC.

Our objective is to propose solutions that allow IPSEC and SSL to work in conjunction with TCP and HTTP proxy servers in hybrid satellite networks, so that the unicast communication is secured without sacrificing the performance optimization algorithms. For this, we look at prospective candidate protocols and evaluate their performance through simulations.

6.2 Methodology: Layered IPSEC for co-existence with TCP PEPs

For network layer encryption and integrity protection, we consider the Layered IPSEC Security protocol (LES), which is based on the concept of breaking the IPSEC encryption in multiple encryption regions or zones on a single packet basis. The method has been proposed independently in [89, 102]. Although the finer details in the two approaches are different, the basic idea is the same. Known as multilayer IPSEC or ML-IPSEC [89], and Layered IPSEC [102], the idea is to encrypt different regions of the IP packet using different keys (fig. 6.2(b)). The TCP payload is encrypted with key $K1$ which is shared only between the endpoints. The original IP header and the TCP header are encrypted with key $K2$ which is shared by the endpoints with intermediate authorized nodes like the TCP PEP. Therefore a TCP PEP can decrypt the header portion of the ESP packet with $K2$ and read the TCP header to do its performance optimizations. But the PEP cannot read the TCP payload and therefore cannot access the actual data since it does not posses the key $K1$.

In [102], the authors have demonstrated the correctness of operation of LES by implementing the protocol for firewalls and network monitoring. The authors have also evaluated the performance of LES for packet encryption and have shown that the performance is comparable to IPSEC for the specific cryptographic algorithms used. A quantitative analysis of the packet byte overhead and software overhead between IPSEC and ML-IPSEC is given in [89]. The paper shows that the throughput overhead of ML-IPSEC in a simple test-bed is 2%-7% compared to IPSEC.

We believe that the Layered IPSEC approach would be an effective security solution in hybrid satellite networks that would allow TCP PEPs to function effectively. However, whether the protocol improves the delay performance in a hybrid satellite network, compared to IPSEC, is an open question. This is because Layered IPSEC introduces higher complexity and higher communication overhead compared to IPSEC in the establishment of the secure channel that now requires co-ordination not only between the end points, but also with the proxy servers. We therefore investigate the performance of end-to-end traffic when Layered IPSEC is implemented in conjunction with TCP PEP optimizations in a hybrid satellite network, and compare to the case where IPSEC is implemented (and therefore the TCP PEPs cannot function). In this context, the performance criteria of interest is the application round-trip delay.

6.2.1 Modification to Internet Key Exchange Protocol (IKE) for Layered IPSEC

Both [102] and [89] assume pre-shared keys between the end points and the proxy servers to establish the secure channel. However, a more realistic situation is the case where the end points do not have any apriori security association; the keys required for IPSEC encryption are dynamically determined at the time the end points want to establish a secure communication channel. In this case, the Internet Key Exchange (IKE) [103] is used to generate the keys. IKE performs a series of handshakes between the end points to establish the two keys used by IPSEC for encryption/decryption and two keys for authentication (the usage of the keys in IPSEC is uni-directional).

IKE operates in two phases - *phase 1* and *phase 2*. The phase 1 exchange happens once initially, based on the identities of the two endpoints and security parameters such as public key pairs or digital signatures or pre-shared secrets between the two identities. The phase 1 exchange creates a security association that allows multiple phase 2 connections to be set up between the client and the server. The phase 1 exchange has 3 steps:

1. *Negotiation of protection mechanisms*, in which the initiator and responder peers exchange information on their identities and the security algorithms that each uses;

2. *Diffie-Hellman exchange*, in which the initiator and the responder exchange public keying data that is used for generating the session keys; and,

3. *Authentication*, in which the two parties verify the keying material that each has independently generated by exchanging hash values computed on the keying materials.

The IKE phase 1 exchange can happen in either the *main* mode, in which there are 3 pairs of message exchanges between the end points, or in the *aggressive* mode, in which all the exchanges are condensed into a total of 3 messages.

In the IKE phase 2 *quick* mode, there is a total of 3 exchanges between the initiator and the responder peers, during which the two parties verify the keying material that each will use for the session. The phase 2 exchange uses the session keys established in phase 1 to do mutual authentication and establish a phase 2 session key. Based on the phase 2 session key, the two end points agree on a set of four keys that is used by IPSEC to protect the data traffic between the two nodes by performing authentication and/or encryption in each direction of the secure channel.

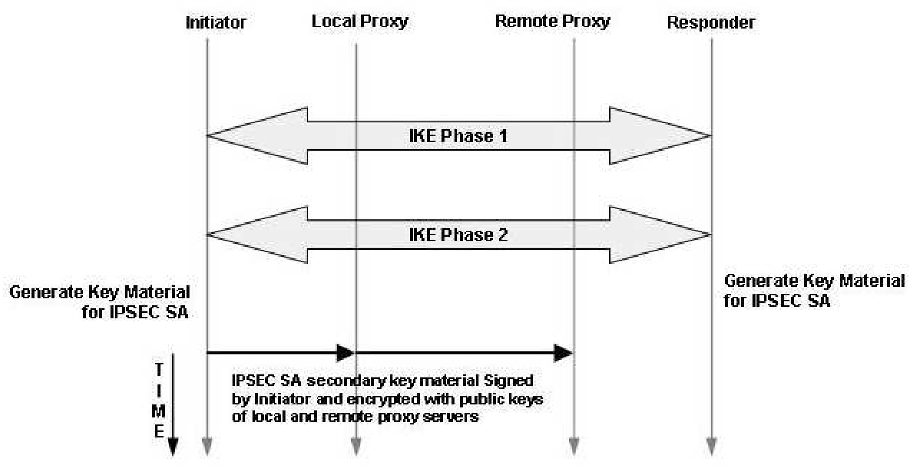

Initiator
Local Proxy
Remote Proxy
Responder
IKE Phase 1
IKE Phase 2
Generate Key Material for IPSEC SA
Generate Key Material for IPSEC SA
IPSEC SA secondary key material Signed by Initiator and encrypted with public keys of local and remote proxy servers
TIME

Figure 6.4: Addition to IKE handshake mechanism for Layered IPSEC key management

We propose to modify the IKE protocol to incorporate the generation of additional keys needed for Layered IPSEC. In the modified protocol, in IKE phase 1 the initiator entity (which would be the remote client node in our scenario) includes the certificates of the remote and hub proxy nodes in the protection mechanism negotiation stage with the responder entity (the server node in our scenario). The keying data that is exchanged between the end points in the modified IKE phase 1 is subsequently used in IKE phase 2, so that the client and the server agree on a set of six keys - the four keys for forward and reverse encryption and authentication between the client and server, and two additional keys to be used by the sender to perform layered encryption on the IP header and also layered authentication. We add a fourth message dissemination to IKE phase 2, in which the client distributes the two additional keys to the local and remote proxy servers. The client encrypts the IP header encryption keys using the public keys of the proxy servers (we assume that the public keys and certificates of the proxy servers are available to the client), authenticates the message using a digital signature, and sends the authenticated message to the proxy servers. Figure 6.4 illustrates the step that we add to IKE phase 2 for key management for Layered IPSEC.

Additionally, in situations where the security association between the end points require dynamic establishment, we propose to use only the aggressive mode of IKE phase 1 exchange, and the quick mode of IKE phase 2 exchange. This is to contain the negative effect of the long round-trip delay on the overall performance - we have to ensure that the delay incurred due to the IKE message exchanges do not neutralize the advantages that might be gained due to the use of Layered IPSEC. The IKE phase 1 aggressive mode will reduce the delay by 50%, compared to the IKE phase 1 main mode. Whether that is sufficient savings, we evaluate through simulations in section 6.2.2.

6.2.2 Performance Evaluation of Layered IPSEC with IKE modifications for Hybrid Satellite Networks

We have analyzed the performance of Layered IPSEC with IKE modifications through simulations on a Opnet Modeler [3] testbed (figure 6.5). The simulation setup consists of a remote client connected to a server via a satellite link. The connection to the server goes through the satellite NOC or hub. TCP PEPs exist both at the remote site and at the satellite hub. All communications between the client and the server pass through the remote and hub TCP PEPs. The satellite is in geostationary orbit, with link delay of 130 milliseconds. The uplink bandwidth is set to 256 kbps, while the downlink bandwidth is 70 Mbps. Simulations have been run on various combinations of the parameters that affect TCP acceleration in satellite networks, and these have been compared to the base case.

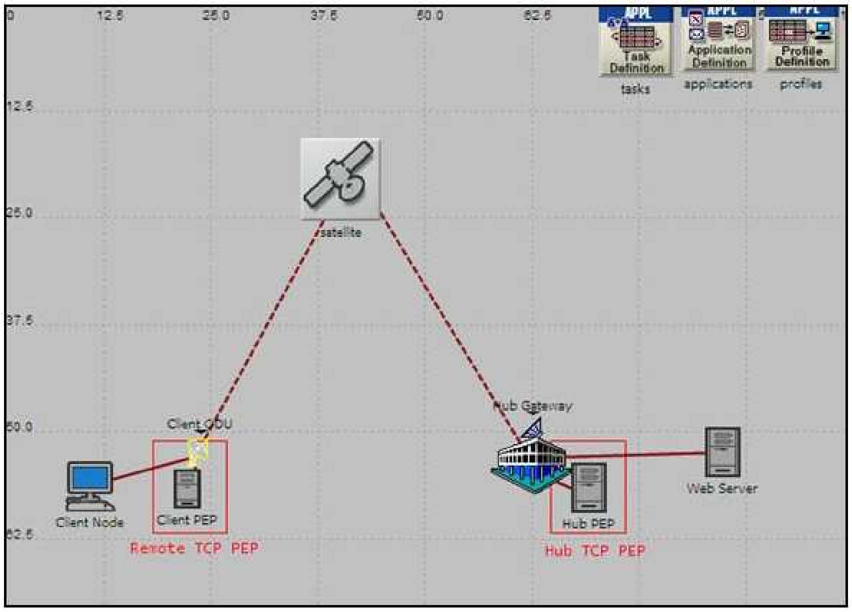


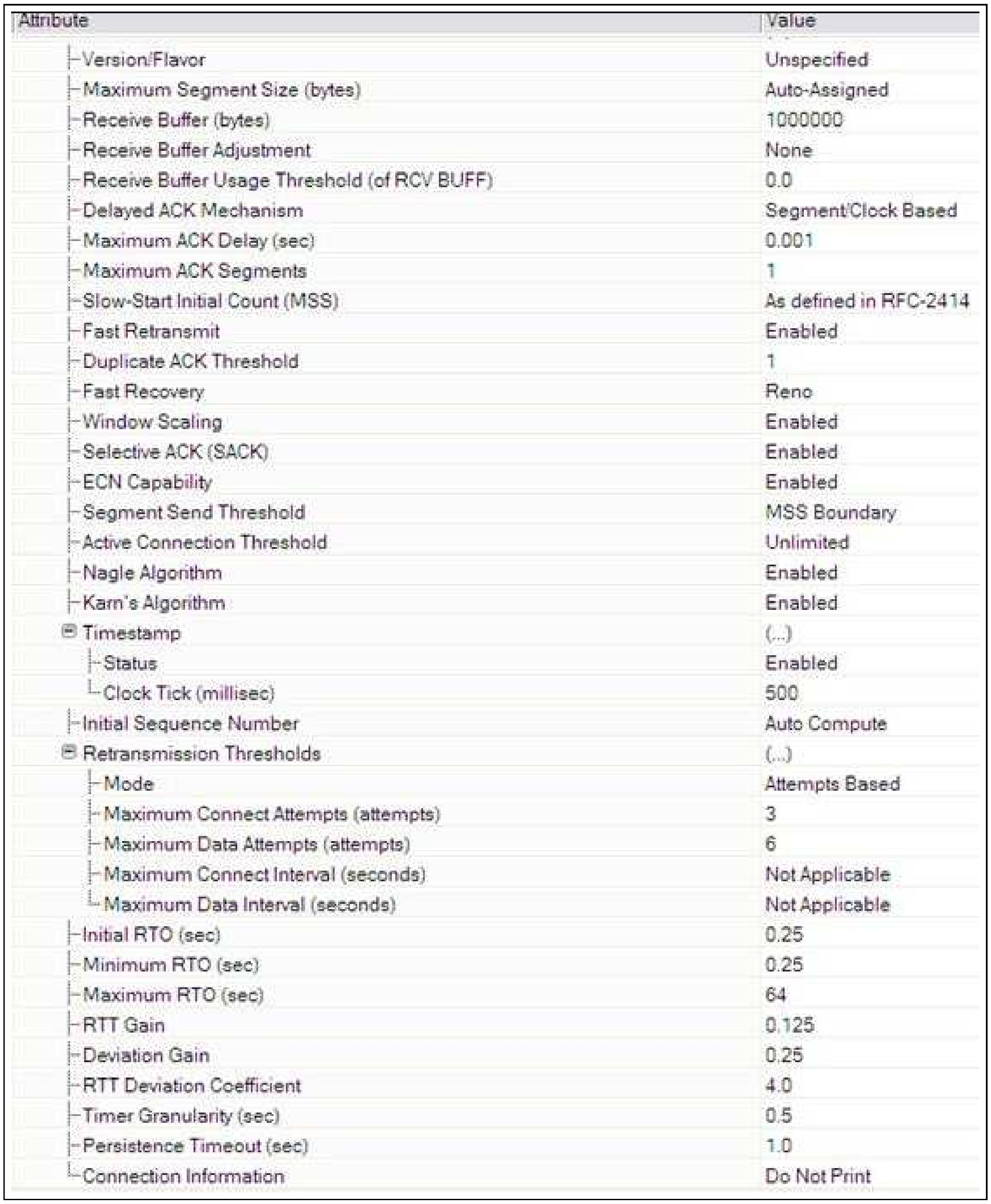
Figure 6.5: Opnet Modeler Simulation Testbed for Layered IPSEC with IKE Modifications

The following enumerates the different simulation scenarios that have been considered.

1. We have used TCP with default parameters in the cases where no TCP acceleration is enabled, while for scenarios where TCP acceleration is enabled, full-featured TCP is used with various performance improvements, such as much larger receive buffers (1000000 bytes as compared to 8760 bytes for the default case), window scaling and fast retransmit. Figure 6.6 shows the different TCP parameters that we used for the enhanced version of TCP.

2. Satellite channel bit-error-rates (BER) of $1 \times 10^{-6}, 1 \times 10^{-5}, 1 \times 10^{-4}$ have been considered in the scenarios.

3. We have considered two cases when Layered IPSEC security is used - with modified IKE and with pre-shared keys (and hence no IKE). Likewise, scenarios with



Attribute	Value
Version/Flavor	Unspecified
Maximum Segment Size (bytes)	Auto-Assigned
Receive Buffer (bytes)	1000000
Receive Buffer Adjustment	None
Receive Buffer Usage Threshold (of RCV BUFF)	0.0
Delayed ACK Mechanism	Segment/Clock Based
Maximum ACK Delay (sec)	0.001
Maximum ACK Segments	1
Slow-Start Initial Count (MSS)	As defined in RFC-2414
Fast Retransmit	Enabled
Duplicate ACK Threshold	1
Fast Recovery	Reno
Window Scaling	Enabled
Selective ACK (SACK)	Enabled
ECN Capability	Enabled
Segment Send Threshold	MSS Boundary
Active Connection Threshold	Unlimited
Nagle Algorithm	Enabled
Karn's Algorithm	Enabled
⊟ Timestamp	(...)
Status	Enabled
Clock Tick (millisec)	500
Initial Sequence Number	Auto Compute
⊟ Retransmission Thresholds	(...)
Mode	Attempts Based
Maximum Connect Attempts (attempts)	3
Maximum Data Attempts (attempts)	6
Maximum Connect Interval (seconds)	Not Applicable
Maximum Data Interval (seconds)	Not Applicable
Initial RTO (sec)	0.25
Minimum RTO (sec)	0.25
Maximum RTO (sec)	64
RTT Gain	0.125
Deviation Gain	0.25
RTT Deviation Coefficient	4.0
Timer Granularity (sec)	0.5
Persistence Timeout (sec)	1.0
Connection Information	Do Not Print

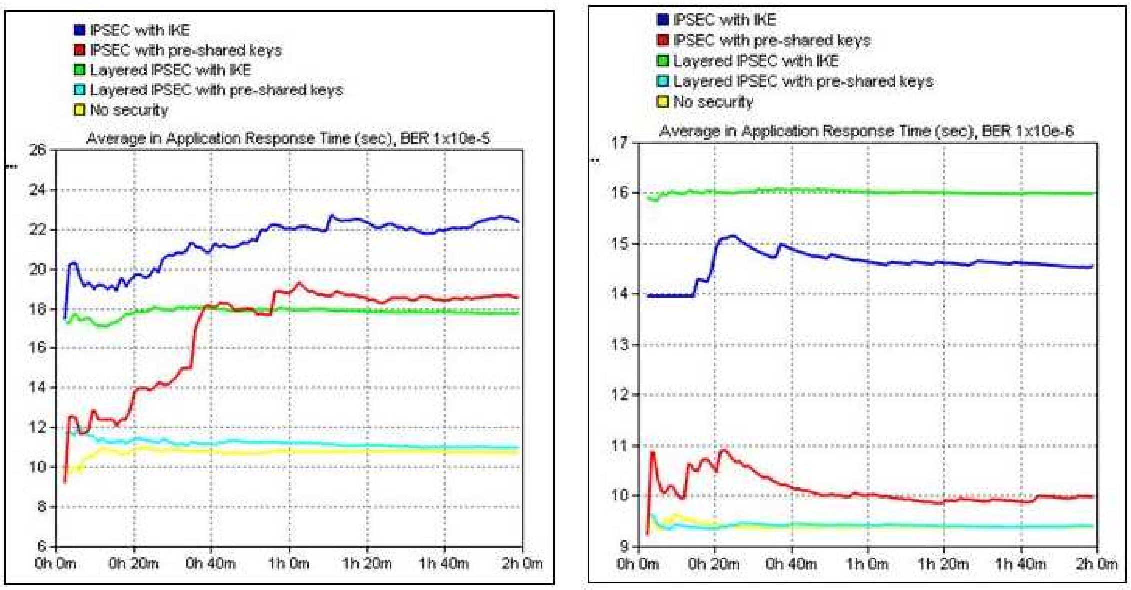
Figure 6.6: TCP Enhancements for Layered IPSEC Simulation Testbed

traditional IPSEC also had two sub-types - with standard IKE and with pre-shared keys. We have also considered basic scenarios with unsecured data transmission.

4. The overhead of the IKE handshake over the satellite links between the client and the server may introduce substantial delay in the overall communication. Therefore to mitigate the negative consequences of IKE, we have used the *aggressive mode* of IKE phase 1, which condenses the client-server handshake into 3 exchanges, compared to 6 exchanges for the *main mode* of IKE phase 1. For a similar reason we have used the *quick mode* of IKE phase 2.

5. For applications, we have considered two application types. One is a custom application where the client and the server engage in a series of secure data exchanges where the client both stores and retrieves data from the server. This simulates a secure telecommuting session executed by an off-site employee with the office server. The second application we have considered is web browsing with HTTP 1.1 over the secure IPSEC tunnel.

The primary objective in the simulations is to verify the theoretical assumption that Layered IPSEC improves the secure client-server communication since it allows TCP enhancements, compared to standard IPSEC. This is verified by comparing the end-to-end application response times in the two cases. We also compare to the case where no security is present to identify the additional delay introduced by the security protocols (that is, the tradeoff). In addition to demonstrating the improvement in application delay response, if any, we are interested in quantifying how much overhead is introduced by Layered IPSEC over that of standard IPSEC. This is done by comparing the TCP load in the various cases.



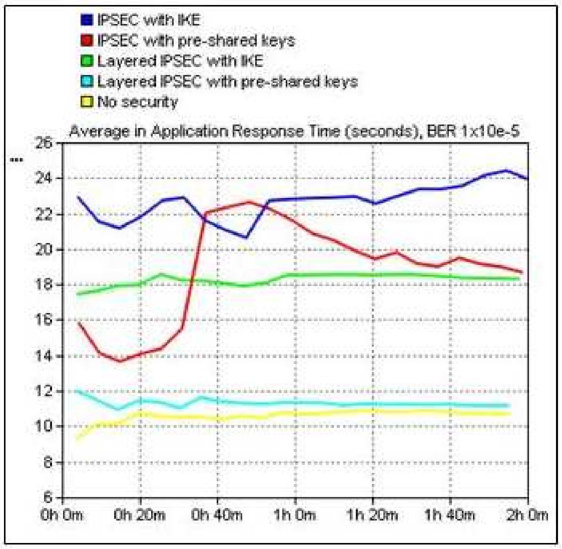
(a) Satellite link BER 1×10^{-5} (b) Satellite link BER 1×10^{-6}

Figure 6.7: Comparison of custom application response time (global average) of Layered IPSEC, IPSEC and unsecured transmission. *X-axis is the simulation time in minutes; Y-axis is the application response time in seconds.*

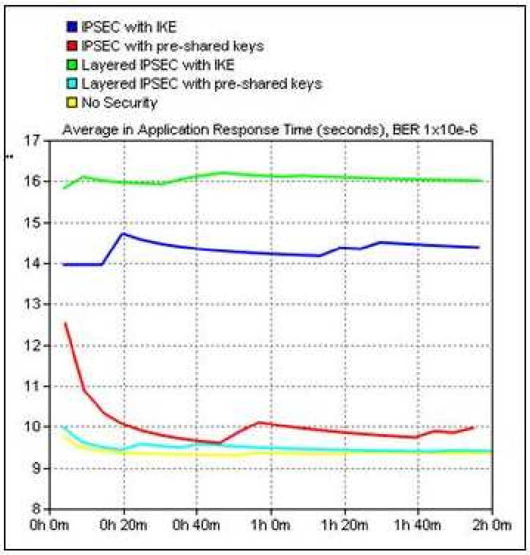
Figure 6.7 shows the overall average application response time when the custom application at the client is communicating with the server. When the satellite channel BER is high, 1×10^{-5}, the application response time for layered IPSEC is significantly better than that of IPSEC, both for the case when IKE is used for secure channel establishment, and also when pre-shared keys are used (figure 6.7(a)). This follows because when IPSEC is used, the TCP optimizations are not working and therefore TCP considers the channel BER to be signs of congestion and thus goes into recovery mode quicker. The graphs also indicate that using IKE adds significantly higher delay compared to using pre-shared keys, which is further explored in later simulation results when the IKE handshake is de-coupled from the actual data transfer. This is due to the multiple message exchanges between the client, the server and the PEPs that are needed by IKE to establish the secure channel. Each message exchange goes over the satellite links and adds to the overall delay. In fact, the delay for Layered IPSEC with pre-shared keys is nearly as low as that of unsecured data transmission with full TCP optimization, which has the lowest delay. The slightly higher delay for the former is primarily due to the IPSEC processing overhead at the nodes, and the slight transmission overhead due to the larger packet sizes due to Layered IPSEC headers and trailers. The effect of TCP optimizations is so pronounced that the lack of optimizations can have a greater effect on the overall delay than the IKE overhead. This is illustrated by the delay graph of IPSEC with pre-shared keys, which starts out much lower compared to Layered IPSEC with IKE (as can be expected), but it climbs higher when the un-optimized TCP in the former case runs into channel errors.

When the channel BER is lower at 1×10^{-6}, the IKE overhead dominates the TCP enhancements and hence Layered IPSEC with IKE has higher overall response time compared to IPSEC with IKE (figure 6.7(b)). Both Layered IPSEC with pre-shared keys and IPSEC with pre-shared keys have significantly lower delays in this case compared to the IKE graphs, with the former being much lower than the latter and is almost the same as the delay for unsecured transmission.

We also ran simulations for satellite channel BER 1×10^{-4}. The very high BER meant that TCP was unable to establish and/or maintain connections for long enough such that meaningful application data could be transferred. Simulation logs indicated that the connections were continually broken for both basic TCP and its enhanced flavor, and hence we could not obtain any meaningful simulation data for this BER scenarios.



(a) Satellite link BER 1×10^{-5}

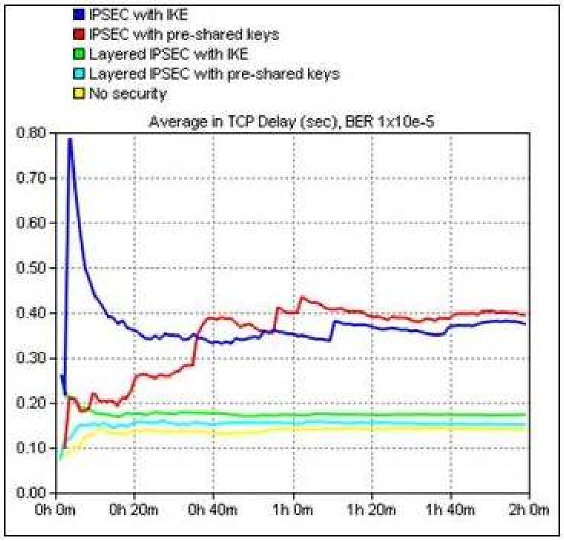


(b) Satellite link BER 1×10^{-6}

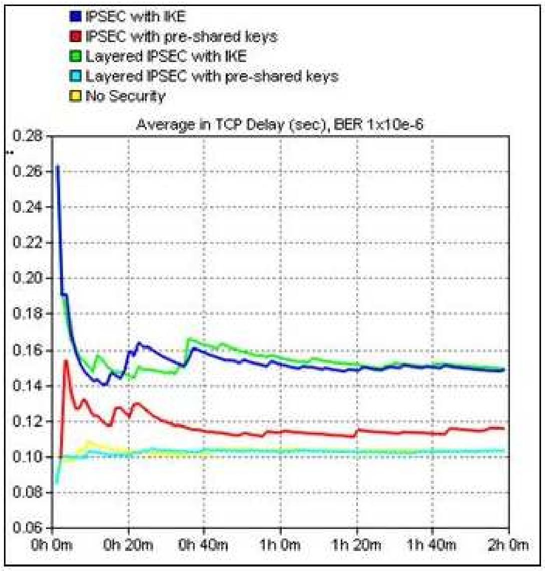
Figure 6.8: Custom application average response time at client for Layered IPSEC, IPSEC and unsecured transmission. *X-axis is the simulation time in minutes; Y-axis is the application response time in seconds.*

The analysis above is reinforced by the average application response time as seen from only the client in figure 6.8. The graphs in this case follow the same trend as for the global response time graphs in figure 6.7.

Figures 6.9 and 6.10 respectively show the network parameters TCP delay and retransmission counts overall, for both BER 1×10^{-5} and 1×10^{-6}. For the high channel BER, the basic TCP (for the two IPSEC graphs in figure 6.9(a)) have higher delay since they cannot take advantage of the TCP optimizations that are available to enhanced TCP (the other graphs in figure 6.9(a)). However, for lower BER when the TCP connection is broken less frequently, the IKE overhead dominates and hence the two scenarios with IKE have higher TCP delay (figure 6.9(b)). The advantage of the larger buffer sizes in enhanced TCP is indicated in the higher retransmission counts for the Layered IPSEC and



(a) Satellite link BER 1×10^{-5}

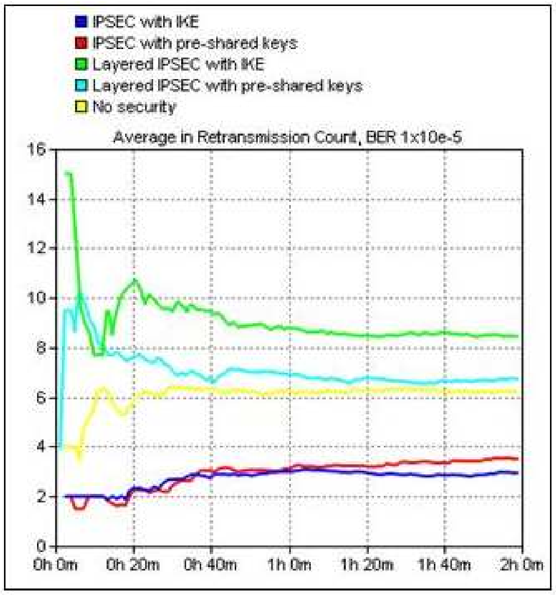


(b) Satellite link BER 1×10^{-6}

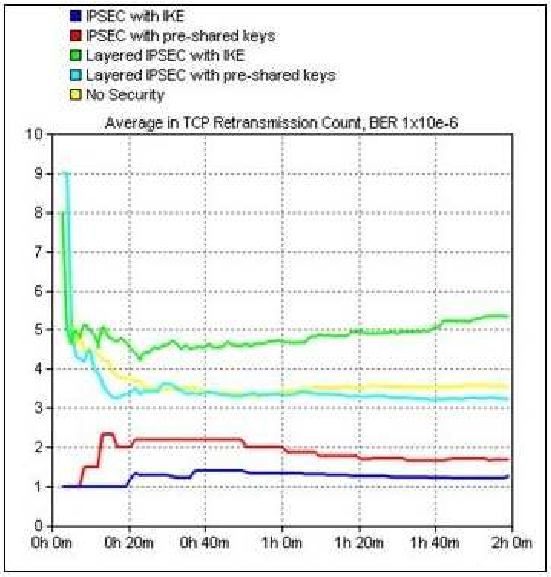
Figure 6.9: Average of overall TCP delay for Layered IPSEC, IPSEC and unsecured transmission. *X-axis is the simulation time in minutes; Y-axis is the TCP delay time in seconds.*

unsecured transmission graphs in figures 6.10(a) and 6.10(b). The enhanced TCP version makes multiple attempts to transmit the TCP segments if no acknowledgment is received before timeout, while basic TCP drops the segments since the buffer gets filled up much faster.

The byte overhead due to the security additions is shown by the network TCP load at the client in figure 6.11. Figure 6.11(a) shows that the average TCP load due to Layered IPSEC with IKE is much higher than the other cases. At simulation time 1 hour 40 minutes, the Layered IPSEC TCP load is 89.938562bps, which is 42% higher than the TCP load for unsecured transmission (63.333333bps), and 37.7% higher that IPSEC with IKE (65.318627bps). However, this high overhead is mostly due to IKE. In the case of Layered IPSEC with pre-shared keys, the TCP load is 66.686275bps at simulation



(a) Satellite link BER 1×10^{-5}

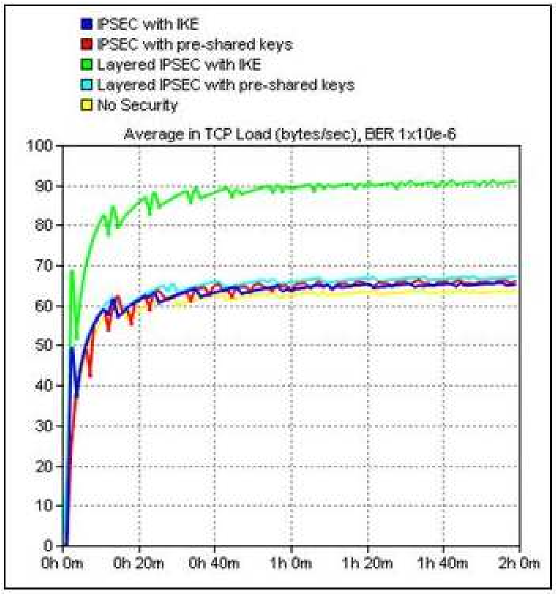


(b) Satellite link BER 1×10^{-6}

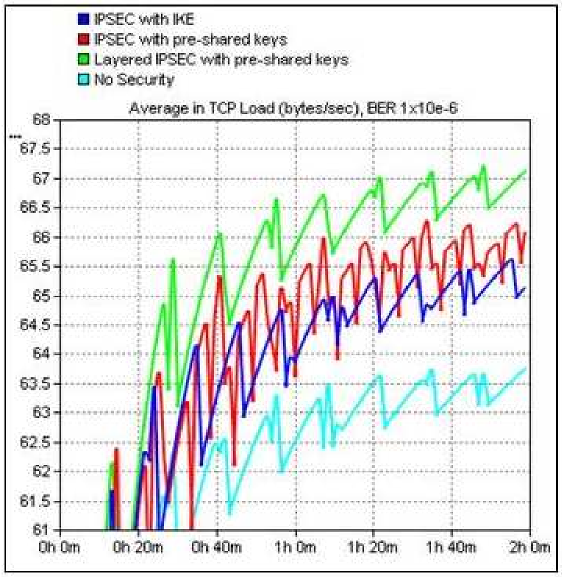
Figure 6.10: Average count of overall TCP retransmissions for Layered IPSEC, IPSEC and unsecured transmission. *X-axis is the simulation time in minutes; Y-axis is the TCP retransmission count.*

time 1 hour 40 minutes (figure 6.11(b)). This is only 1.1% higher than IPSEC with pre-shared keys (65.941176bps), and 5.3% higher than unsecured transmission. The values are consistent with the case for satellite channel BER 1×10^{-5} (not shown). These results indicate that Layered IPSEC can be a viable alternative to IPSEC for satellite networks, with comparable byte overhead while providing significant improvement in application performance. However, this holds true only if the secure channel is established apriori. While using IKE will still result in improved application response times for high channel error conditions, it might introduce unacceptably high overhead even with the proposed modifications.

The dominant effect of IKE when channel errors are low, is highlighted in figure 6.12. The global average of the response time for IKE handshake when the channel BER



(a) Satellite link BER 1×10^{-6}

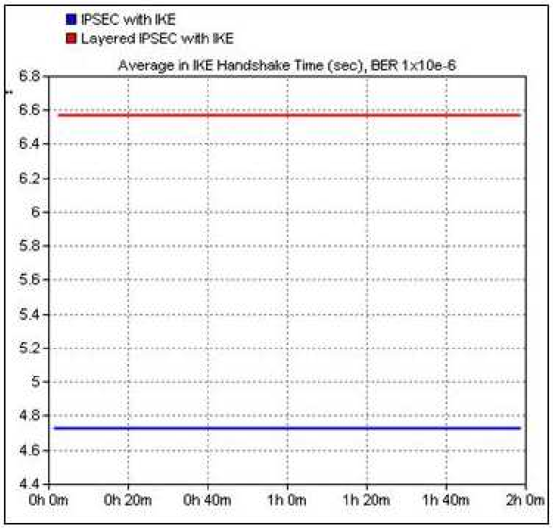


(b) Satellite link BER 1×10^{-6}, sectional closeup

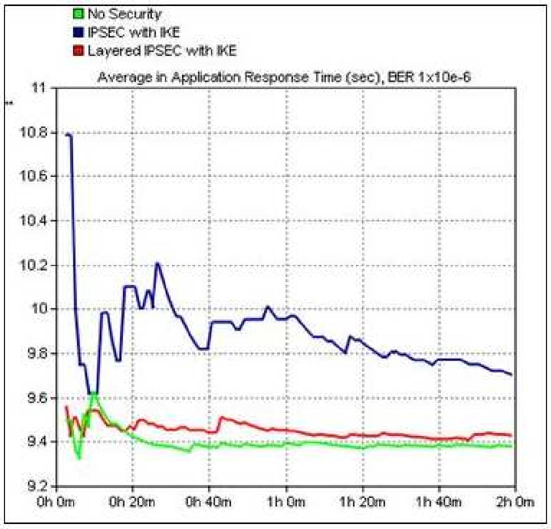
Figure 6.11: Average TCP load at client for Layered IPSEC, IPSEC and unsecured transmission. *X-axis is the simulation time in minutes; Y-axis is the TCP load in bytes/second.*

is 1×10^{-6}, is shown in figure 6.12(a). With this IKE overhead de-coupled from the the actual application, the response times for the application are given in figure 6.12(b). The latter figure clearly shows the advantage of Layered IPSEC over IPSEC, which is obscured when IKE handshake is integrated into the response time (see figure 6.7(b)).

Finally, figure 6.13 compares the IKE handshake time and application response times that are obtained for web browsing when Layered IPSEC with IKE is used, compared to IPSEC with IKE and unsecured transmission. The results are consistent with those for the custom application discussed above. For higher channel BER, the advantage of enhanced TCP with Layered IPSEC is visible but this advantage is neutralized by the IKE overhead when the BER is lower (figure 6.13(b)). However, as figure 6.14 shows, when pre-shared keys are used, Layered IPSEC is clearly better than IPSEC and is close



(a) Response time for IKE handshake



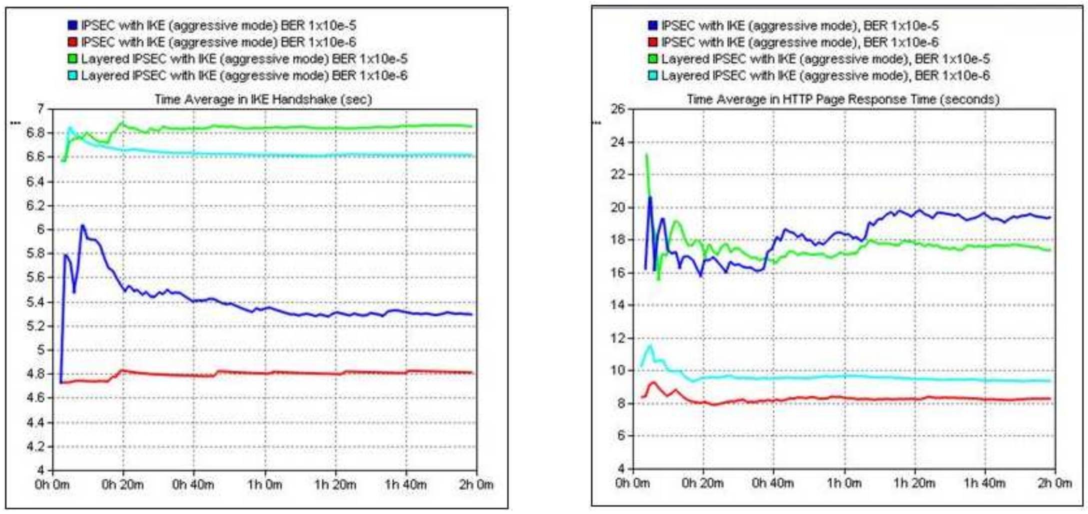
(b) Response time for custom application without IKE handshake

Figure 6.12: Effect of IKE handshake on application response times, BER 1×10^{-6}. *X-axis is the simulation time in minutes; Y-axis is the application response time in seconds.*

to unsecured web browsing.

6.3 Dual-Mode SSL: HTTP proxy-friendly Secure Web Browsing

When the HTTP traffic is secured using SSL only, and there is no IPSEC tunnel in use, and the security policy does not allow for trusted third parties, we propose the use of a modified SSL protocol, the *Dual-Mode SSL* (DSSL) protocol. As shown in fig. 6.15, the secure connection in DSSL has two modes - an end-to-end *main* mode connection between the client and the web server, and a *secondary* mode connection that has the hub HTTP proxy as an intermediate node. When secure HTTP traffic is requested, the DSSL main mode connection is first negotiated between the client and the server. As part of the handshake for the main mode, the client and the web server also negotiate



(a) IKE handshake time (b) HTTP response time

Figure 6.13: Comparison of IKE handshake time and application response times for HTTP traffic with Layered IPSEC, IPSEC and unsecured transmission. *X-axis is the simulation time in minutes; Y-axis is the application response time in seconds.*

the parameters for the secondary mode. As shown in figure 6.16, in DSSL the client first contacts the hub proxy and obtains its certificate. The client then initiates a traditional SSL connection to the server. However the client now forwards to the server information about the proxy, including the proxy certificate, in addition to its own security information. The key exchange done by the client includes the secondary key for the proxy. Once the client and server have agreed on the keys for the DSSL connection, the client triggers the hub proxy to contact the server. The proxy initiates a DSSL connection to the server and the latter sends the secondary key materials to the proxy. Once the handshake between the server and the proxy is over, the proxy informs the client that it has obtained the secondary key material from the server, and the handshake phase is completed by the client. All the message exchanges in DSSL are authenticated using digital certificates.

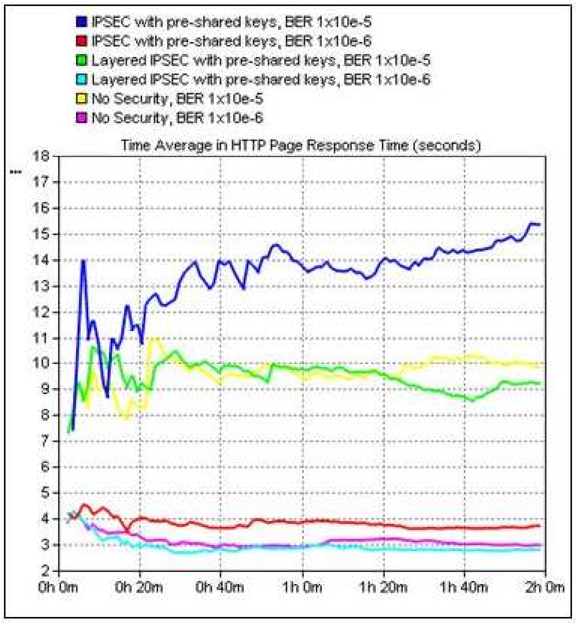


Figure 6.14: Comparison of HTTP application response times with Layered IPSEC and IPSEC with pre-shared keys and unsecured transmission. *X-axis is the simulation time in minutes; Y-axis is the application response time in seconds.*

Let $K1$ be the encryption key for the main mode, and $K2$ be the encryption key for the secondary mode. When the client makes a HTTP request, the client proxy sends local replies to the client browser. The web server, on receiving the request, parses the requested HTML page to obtain the embedded object links, which are collated into a new HTML page. The object links HTML page is then encrypted by DSSL using $K2$ to create the proxy SSL record. DSSL encrypts the base HTML page using $K1$ to create the primary SSL record. The two records are appended together and sent to the client in an IP packet (fig. 6.15). The hub proxy intercepts the IP packet, extracts the object links from the proxy SSL record using $K2$, and pre-fetches the embedded objects. The web server always encrypts the actual objects using $K1$, so that the hub proxy cannot read

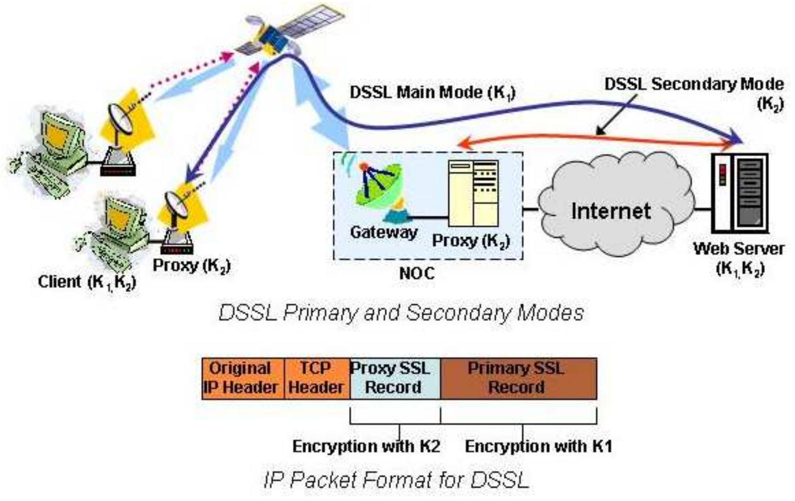


Figure 6.15: Dual-Mode SSL for HTTP optimization

the base HTML page data. The hub proxy transfers the embedded objects to the client together at one time. Therefore the HTTP proxy functionality is preserved in DSSL while maintaining end-to-end security of the HTML page contents.

The DSSL concept is partly similar to the multiple-channel SSL concept proposed in [104]. However, in that the authors do not differentiate encryption in primary and secondary SSL records - they suggest that HTTP traffic with lower security requirements be encrypted entirely with keys known to intermediate nodes. For our security requirements, that approach would not be acceptable.

6.3.1 DSSL: Protocol Specification

Figure 6.16 illustrates the message exchanges in DSSL. Since DSSL extends the SSL protocol to include support for HTTP proxy servers, the message structures and many of the protocol steps are similar. DSSL introduces a series of additional message

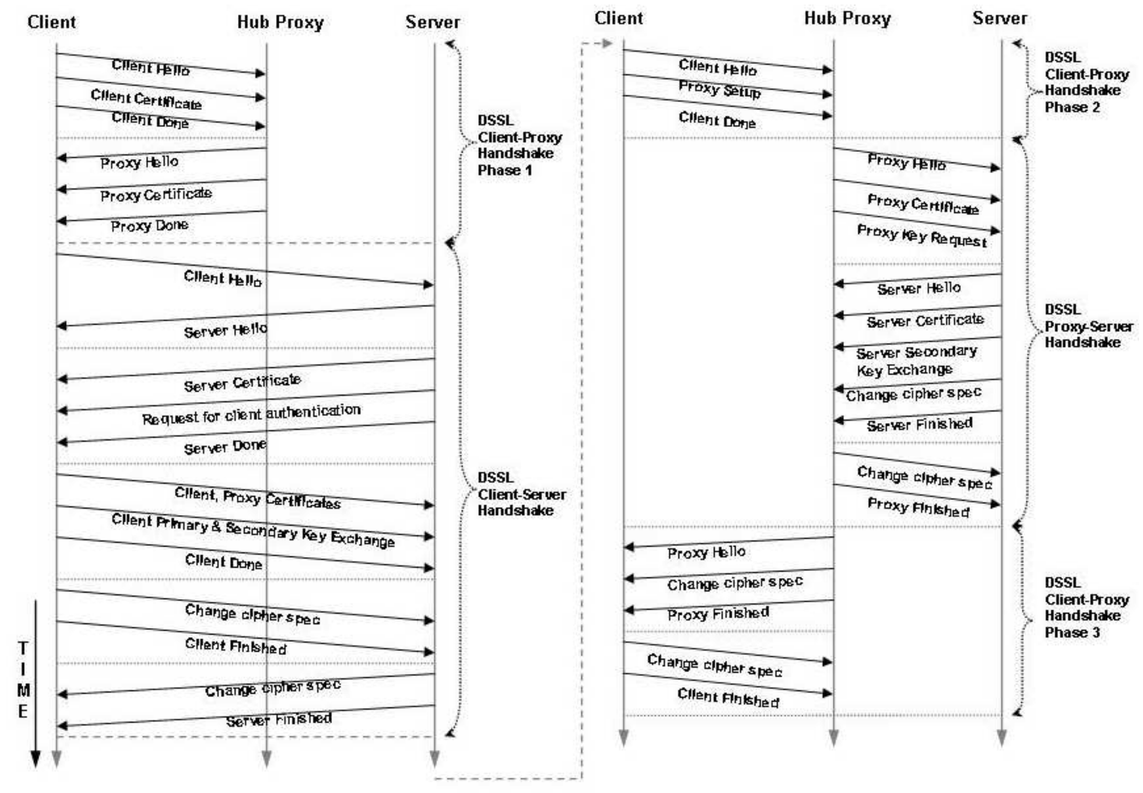


Figure 6.16: Handshake between client, server and proxy in DSSL

exchanges between the client and the HTTP proxy server, and between the proxy server and the web server. As shown in the figure, the DSSL protocol can be divided into several stages.

- **Client-proxy handshake phase 1**: When a remote client wants to establish a secure session with a web server for the very first time, it might not be aware of the security parameters of the HTTP proxy. It therefore establishes a connection to the HTTP proxy and initiates the first stage of the DSSL protocol, the client-proxy handshake phase 1. In this phase the client sends its security certificate to the HTTP proxy, which in turn responds with its own security certificate. The client thus obtains the public key of the HTTP proxy server from the proxy's certificate. The start and end of the communication in each phase is marked by a ”Hello” and a ”Done” message respectively, for each of the participating entities. This is in accordance with the original SSL protocol.

This first stage of the protocol is required to be executed only once between each client-HTTP proxy pair. At future instances, whenever the client wants to establish a secure web session through this proxy, it uses the proxy certificate that it had acquired the first time. This stage does not need to be repeated. The stage will be executed again if the proxy or client certificate expires or is revoked, or if the client communicates with a new proxy server.

- **Client-server handshake**: Once the client has obtained the security certificate of the HTTP proxy, it contacts the web server to exchange security credentials and to establish the session keys for the secure web session. This stage is similar to the SSL protocol, with two exceptions - (i) the client sends both its own certificate and the HTTP proxy certificate to the web server, and (ii) in the key exchange step, the client generates both primary and secondary keys and sends them to the web server.

- **Client-proxy handshake phase 2**: After the client and the web server have established the session keys in the second stage of DSSL, the client again contacts the HTTP proxy and instructs it to obtain the session keys from the web server.

- **Proxy-web server handshake**: The HTTP proxy contacts the web server and sends its certificate to authenticate itself. Upon correct authentication, the web server sends the secondary session key to the HTTP proxy.

- **Client-proxy handshake phase 3**: This final stage of the key establishment protocol is essentially a continuation of the client-proxy handshake phase 2. After the proxy obtains the secondary key from the web server, it contacts the client to confirm that it has received the key. The establishment of the primary and secondary keys between the client, the web server and the proxy is now complete.

Once the DSSL protocol handshake is executed, the HTTP "Get" request from the client HTTP browser is sent to the web server. The client requests are intercepted by the local proxy, who sends spoofed replies to the client browser, while forwarding the request to the web server via the hub HTTP proxy server. The web server, on receiving the request, parses the requested HTML page to obtain the embedded object links, which are collated into a new HTML page. This "object links HTML page" is then encrypted by DSSL using secondary session key to create the proxy SSL record. The web server encrypts the requested HTML page using primary session key to create the primary SSL record. The two records are appended together and sent to the client in an IP packet (fig. 6.15). The hub HTTP proxy intercepts the IP packet, extracts the object links from the proxy SSL record using secondary session key, and pre-fetches the embedded objects. At the same time, the HTTP proxy forwards the IP packet to the client. When the client sends further "Get" requests for the embedded objects, the requests are intercepted and blocked by the client's local proxy. The web server receives only the "Get" requests from the HTTP proxy, encrypted with the secondary session key. In response to these "Get" requests, the web server posts the embedded objects to the client. The web server always encrypts the actual objects using primary session key, so that the hub proxy cannot read the data, even though it had generated the HTTP requests. The hub proxy transfers the encrypted HTTP posts to the client together at one time. Any encrypted data that the HTTP proxy cannot read, it immediately forwards to its destination. Thus the HTTP proxy functionality is preserved in DSSL while maintaining end-to-end security of the HTML page contents.

6.3.2 Performance Evaluation of Dual-Mode SSL

We have analyzed the performance of the DSSL protocol and associated web browsing through simulations on a Opnet Modeler [3] testbed (figure 6.17). The simulation setup consists of a remote client connected to a Internet web server via a satellite link. The connection to the web server is through the satellite NOC or hub. A HTTP proxy server is located at the satellite hub site. All communications between the client and the web server pass through the HTTP proxy. The satellite is in geostationary orbit, with link delay of 130 milli-seconds. The uplink bandwidth is set to 256 kbps, while the downlink bandwidth is 70 Mbps. In all the simulations, the client retrieves a webpage containing multiple embedded objects from the server. The multiple simulations that were conducted can be broadly classified into three sets:

1. Scenarios where the web browsing is unsecured and the HTTP proxy is fully functional.

2. Scenarios where SSL is used for secure web browsing and therefore the HTTP proxy is non-operational.

3. Scenarios where DSSL is used for secure web browsing, allowing the HTTP proxy

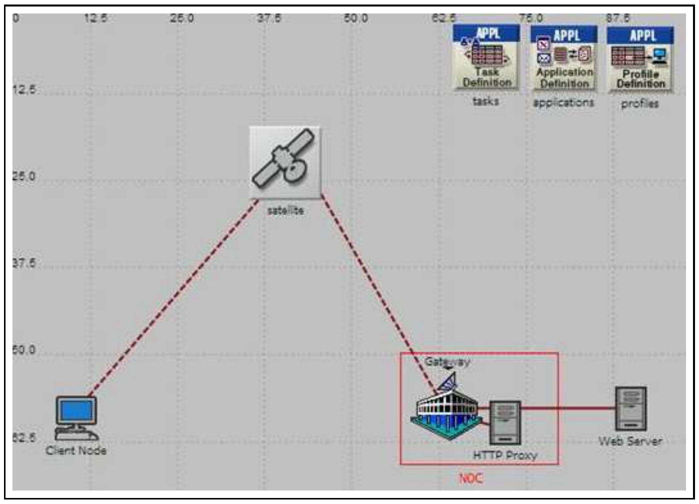
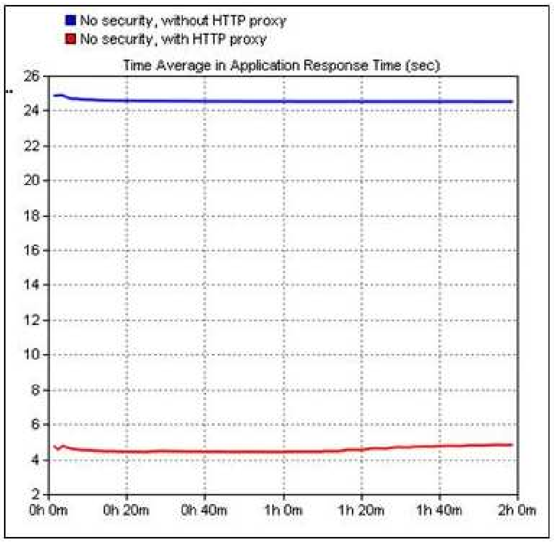


Figure 6.17: Opnet Modeler simulation testbed for the DSSL protocol

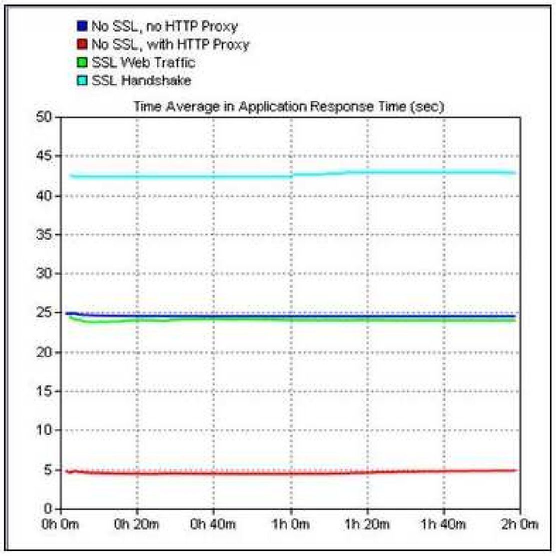
to be functional.

To motivate this discussion, figure 6.18(a) highlights the importance of HTTP proxy for reducing the web browsing delay in satellite networks. The figure compares the time average of overall application response times when the client server retrieves a unsecured webpage with multiple embedded objects, from the server. The response time is a prohibitive 25 seconds without the proxy, but it reduces to only 5 seconds when the HTTP proxy is used. Figure 6.18(b) shows that when SSL is used, the overall response time for retrieving the same webpage increases to the factor when the HTTP proxy is not used. The additional delay due to SSL is split into two graphs - the SSL handshake time takes on the order of 43 seconds, while the secure web browsing takes 25 seconds. This clearly demonstrates that SSL makes the HTTP proxy non-operational.

Figure 6.19 addresses the question whether using enhanced TCP with optimizations can suffice to mitigate the performance negation on web browsing due to lack of HTTP



(a) Reduction in application response time with HTTP proxy server



(b) Comparison of HTTP response times

Figure 6.18: Performance improvement with HTTP proxy server in satellite networks and the detrimental effect of SSL on HTTP proxy performance. *X-axis is the simulation time in minutes; Y-axis is the application response time in seconds.*

proxy acceleration. The motivation is that since the TCP enhancements operate below SSL in the network stack, they are not affected by SSL. However, as the graphs show, while TCP enhancements do contribute to reducing the web browsing response time by a fraction (around 7.5%), the HTTP proxy has a much more pronounced effect - the response time reduction is by nearly 74%. Therefore, it is important that we investigate solutions for HTTP proxy acceleration in the presence of SSL. Finally, when both TCP optimizations and HTTP proxy acceleration are used, the improvement is in the order of 81.5%. Hence, in all subsequent analysis, the experiments were conducted with TCP optimizations in effect.

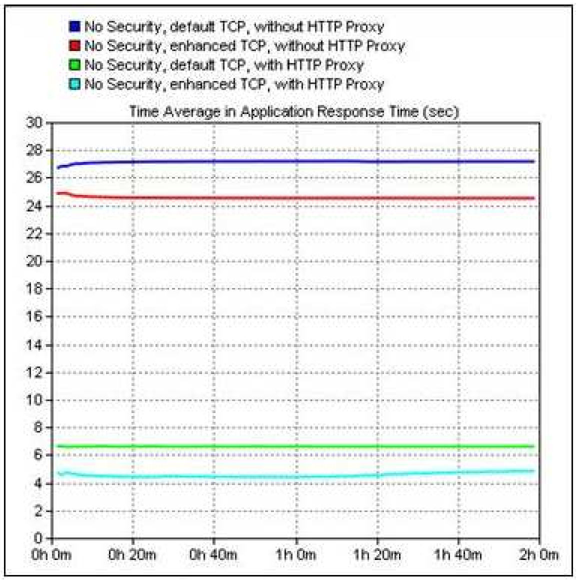
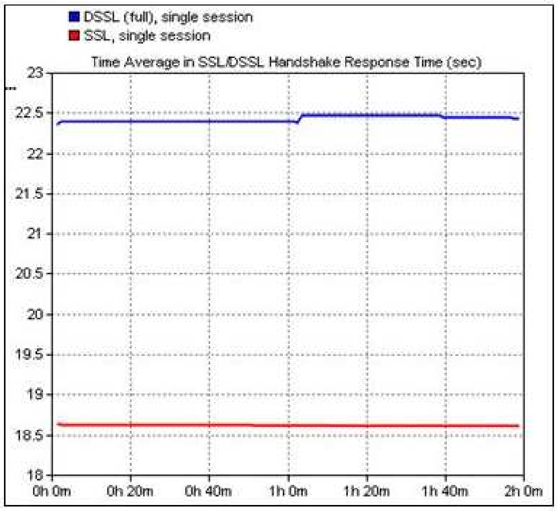


Figure 6.19: A comparison of the effect of TCP enhancements versus HTTP proxy acceleration on web browsing performance in satellite networks. *X-axis is the simulation time in minutes; Y-axis is the application response time in seconds.*

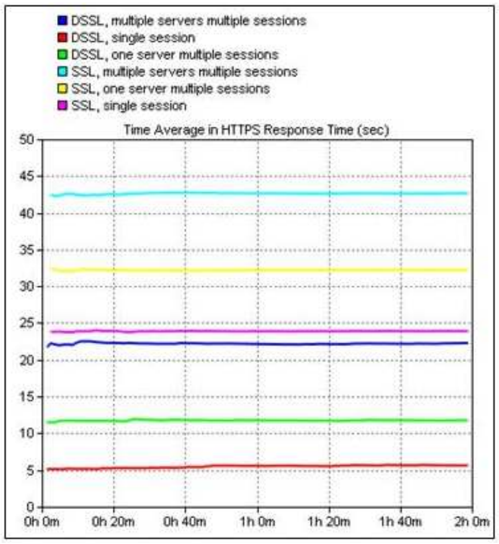
Figure 6.20 gives the application response times when DSSL full mode (figure 6.16) is used to secure the web browsing. In all the simulations with DSSL, we considered three categories of web browsing:

1. The client establishes a single secure session with one web browser. In this case, the DSSL handshake amongst the client, the web server and the HTTP proxy happens only once and the web traffic flows in one session for the rest of the simulation. This is shown in the graphs tagged by ”single session”.

2. The client establishes multiple secure sessions with one web browser. In this case, phases 1 and 2 of DSSL handshake (figure 6.16) happen only once, but phases 3, 4 and 5 are repeated at the beginning of every new session. This allows the



(a) Response times for DSSL handshake



(b) Comparison of HTTP response times with SSL and DSSL

Figure 6.20: DSSL (full) handshake times and comparison of SSL, DSSL web browsing response times.*X-axis is the simulation time in minutes; Y-axis is the application response time in seconds.*

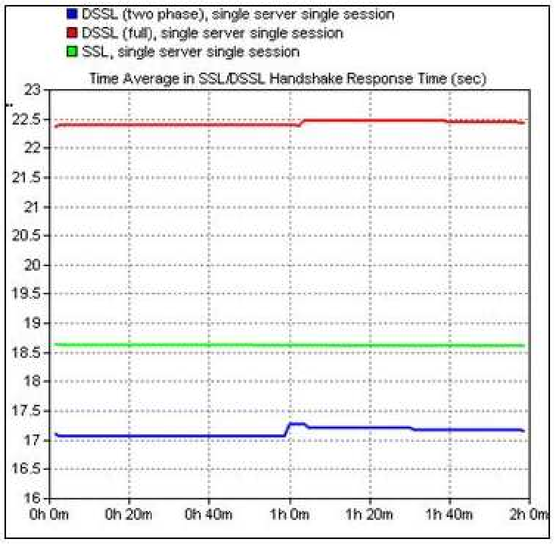
HTTP proxy to obtain from the web server the secure session key for that particular session. This setup is tagged as ”single server multiple sessions” in the graphs.

3. The client establishes individual secure sessions with multiple web servers. In this case, phase 1 of the DSSL handshake happens once at the beginning, while phases 2-5 happen multiple times, once for each web server/session combination. The graphs tagged as “multiple servers multiple sessions” illustrate this setup.

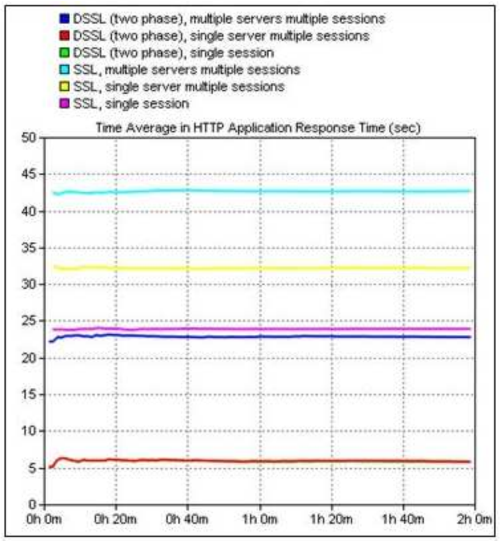
Figure 6.20(a) compares the response times for completing the DSSL handshake against the SSL handshake. The DSSL handshake here follows all the exchanges illustrated in figure 6.16. The graph highlights the delay overhead of DSSL due to the additional steps of the client communicating with the proxy server in phases 1, 3 and 5, and the proxy server contacting the web server in phase 4. The DSSL overhead ranges between 20.3% and 20.74% over the SSL handshake delay. The response time for the secure web traffic is shown in figure 6.20(b). The figure shows the response time-averaged over all the sessions for each scenario. The graph shows that for each category of web browsing enumerated above, the response time for DSSL web browsing is significantly lower than that for SSL web browsing - for example, for one server single session, DSSL web browsing is approximately 5 seconds, compared to 24 seconds for the corresponding SSL case. Figure 6.20(b) also shows that for either DSSL or SSL, the response time is lower for single-server-single-session compared to single-server-multiple-sessions, which again is lower than multiple-servers-multiple-sessions. This is because for single-server-single-session, the DSSL handshake happens only once; while for single-server-multiple-sessions, the DSSL handshake phases 3, 4 and 5 are repeated as many times as the number of web sessions; and for multiple-servers-multiple-sessions, DSSL handshake phases 2 to 5 are repeated the number of times as there are web sessions.

6.3.3 DSSL Two Phase Protocol

Even though figure 6.20 demonstrates that DSSL response times are better than that of SSL, the advantage is largely negated by the higher delay involved in DSSL handshake, as compared to SSL. As a solution to this issue, we propose a condensed version of DSSL, in which the handshake stage consists of only the first two phases from figure 6.16. In the first phase, the client contacts the HTTP proxy and exchanges each other's digital certificate. Subsequently, the client contacts the web server. Once the client and the web



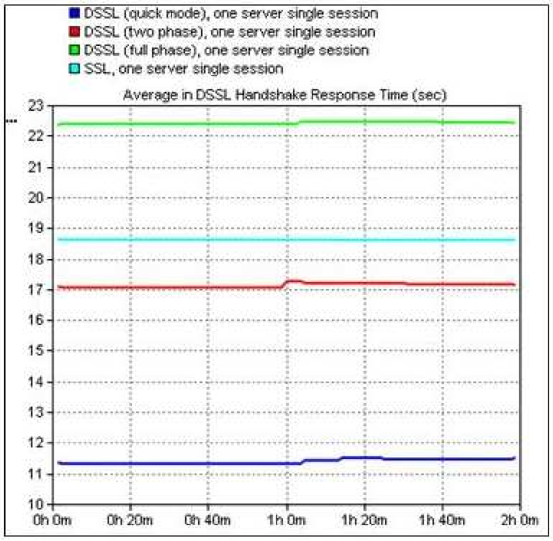
(a) Response times for DSSL handshake (two phase)



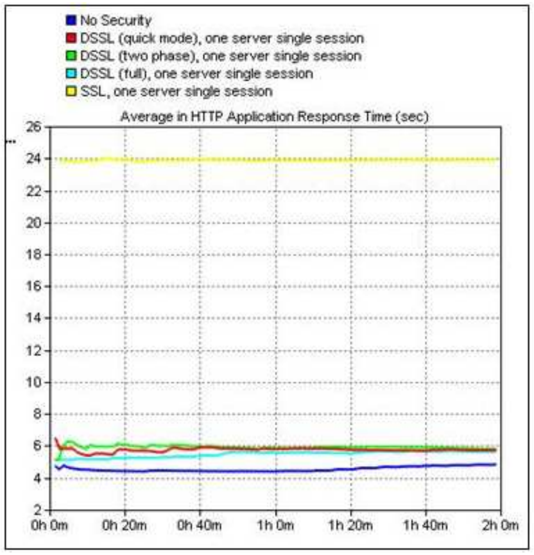
(b) Comparison of HTTP response times with SSL and DSSL (two phase)

Figure 6.21: DSSL two phase handshake times and comparison of SSL, DSSL web browsing response times.*X-axis is the simulation time in minutes; Y-axis is the application response time in seconds.*

server have established the security parameters in phase 2 of DSSL, the client makes the first HTTP "Get" request. This request goes through the HTTP proxy and triggers the proxy to send its certificate to the web server, and a request for the DSSL session secondary keys. The HTTP proxy request is piggy-backed on the client HTTP "Get" request. The server responds to the client request with the base webpage in a HTTP "Post" response. In addition, the server responds to the proxy key request with its own certificate and the DSSL secondary session keys, encrypted with the proxy's public key. This response to the proxy is piggy-backed on the HTTP response to the client. The proxy receives the combined response from the web server and is thus able to retrieve the secondary session keys from the encrypted message, using its private key. Consequently, it is able to decrypt the relevant portions of the HTTP "Post" response and therefore can perform the HTTP acceleration. We refer to this variant of DSSL as *DSSL two phase*.



(a) Comparison of response times for DSSL handshake



(b) Comparison of HTTP response times for single server, single session

Figure 6.22: Comparison of handshake times and web response times for different versions of DSSL and SSL. *X-axis is the simulation time in minutes; Y-axis is the application response time in seconds.*

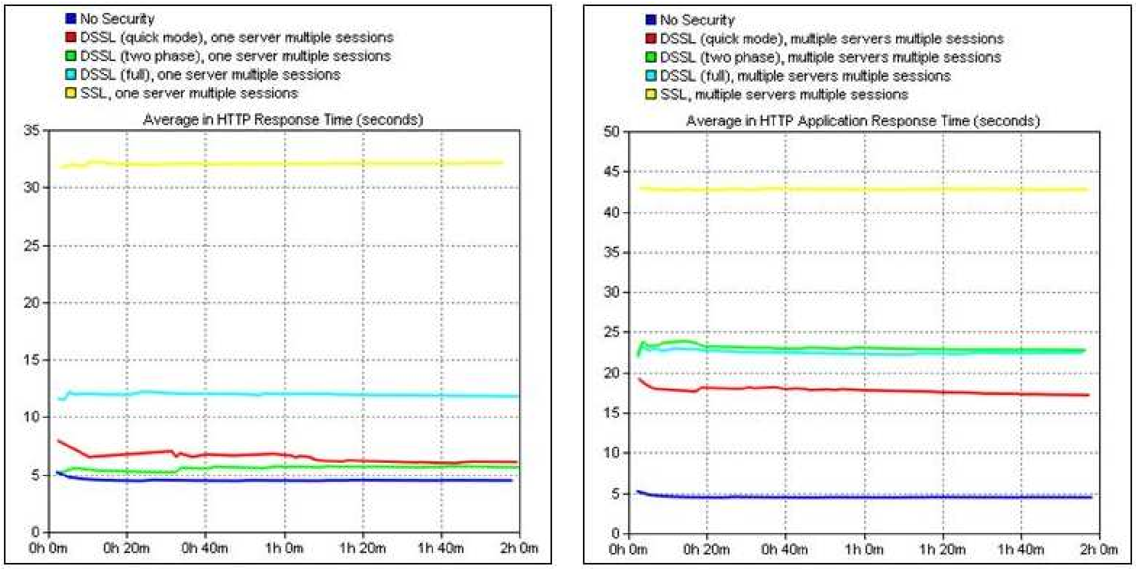
Figure 6.21 gives the application performance graphs for DSSL two phase. Figure 6.21(a) shows that the response time for the DSSL handshake is much lower for DSSL two phase compared to DSSL full mode. The DSSL two phase handshake response time is also comparable to or even lower than that for SSL. The response time for the web browsing traffic for DSSL two phase, compared to SSL, is given in figure 6.21(b). The graphs prove that for each category of web browsing, the response times in case of DSSL two phase are much lower than that for the corresponding SSL case.

6.3.4 DSSL Quick Mode Protocol

The first phase in DSSL two phase is necessary in the situation that the client and the HTTP proxy server do not share any security association beforehand. However, if the two entities are apriori aware of each other's security information (via their digital certificates), then this phase is not needed. Every time the client contacts a web server to initiate a secure web session, it passes to the web server a locally cached copy of the HTTP proxy's certificate. The DSSL protocol can thus be further reduced to just one phase, that of phase 2 in figure 6.16. The DSSL secondary session key is transmitted to the HTTP proxy piggy-backed on the first response from the server, in the method described in section 6.3.3. We refer to this optimization of the DSSL protocol as *DSSL quick mode*.

The optimization in section 6.3.3 or 6.3.4 are proposed to overcome the detrimental effect of the long propagation delay of the satellite channel on the DSSL handshake protocol. This optimization however requires further changes to the HTTP protocol, to allow piggy-backing the DSSL secondary keys on the initial HTTP exchanges between the client, the web server and the proxy server.

Figure 6.22(a) compares the response times of DSSL handshake for its various versions. At simulation time 1 hour and 20 minutes, DSSL quick mode handshake time is 33% less than DSSL two phase (11.5 seconds and 17.2 seconds, respectively). The quick mode time is nearly half (48.8% less) than the handshake time for full version of DSSL (22.465 seconds), and is 38.8% less than that of SSL (18.61 seconds). The web browsing response times for single server, single session, are compared in figure 6.22(b). The times for all cases of DSSL are much lower than that for SSL, and are comparable to unsecured web browsing. The additional delay is primarily due to the security overhead at the different nodes to maintain the secure session and to encrypt/decrypt the traffic.

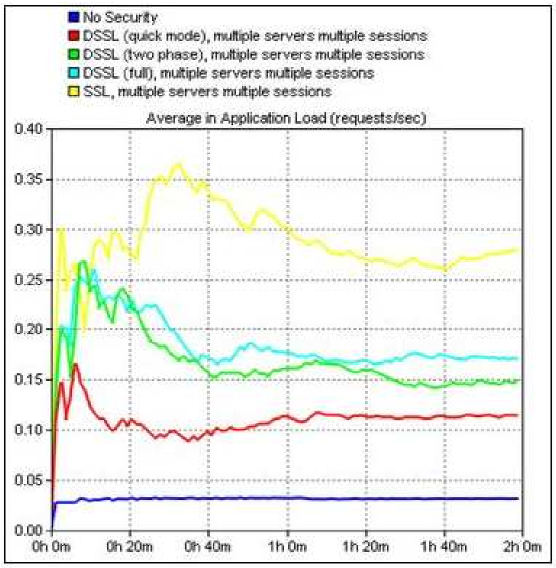


(a) Single server, multiple sessions (b) Multiple servers, multiple sessions

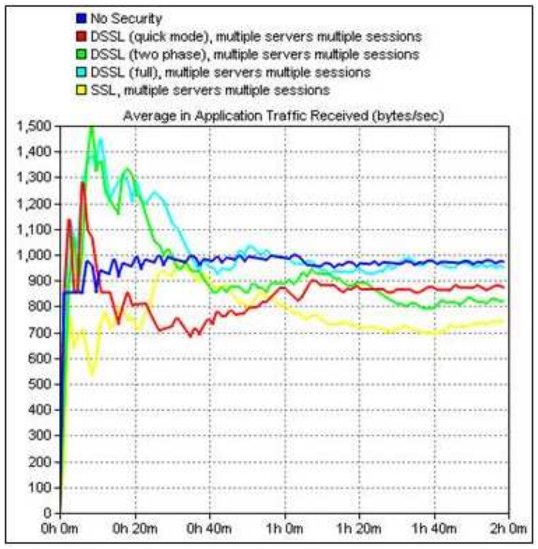
Figure 6.23: HTTP response times for different versions of DSSL and comparison to SSL and unsecured web browsing. *X-axis is the simulation time in minutes; Y-axis is the application response time in seconds.*

Figure 6.23 compares the application response times for multiple secure browsing sessions with one server (figure 6.23(a)) and for secure browsing sessions with multiple servers (figure 6.23(b)). For all the cases, DSSL out-performs SSL.

Figure 6.24 compares the application traffic characteristics at the remote client for multiple-servers-multiple-clients web browsing. The per-second requests are the least increases from the unsecured browsing to DSSL quick mode, two phase, full DSSL and finally SSL, reflecting the increasing load on the network resources for the web traffic (figure 6.25(a)). The TCP load at the client is given by figure 6.25, which shows that the



(a) HTTP application load. *Y-axis is the load in requests/second.*



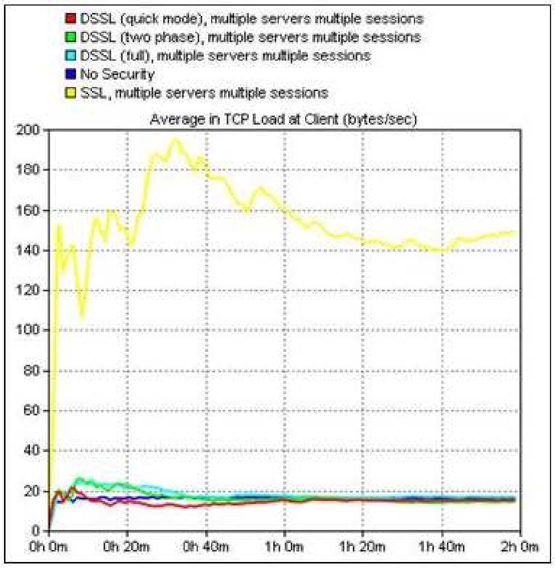
(b) HTTP application traffic received. *Y-axis is the application traffic received in bytes/second.*

Figure 6.24: Comparison of HTTP application load and traffic received by client for multiple servers, multiple sessions. *X-axis is the simulation time in minutes.*

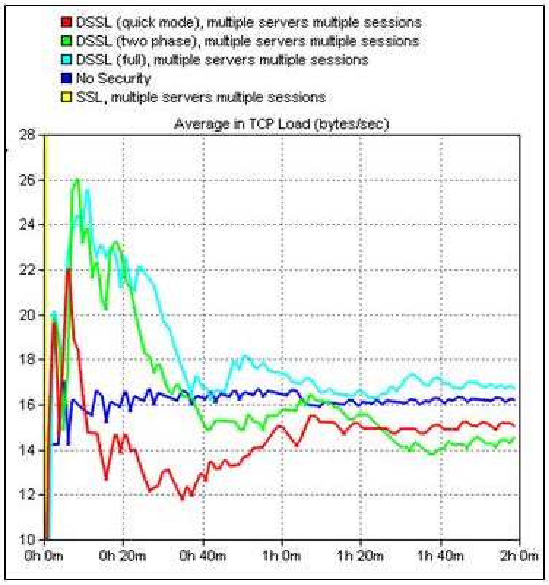
SSL load is much higher compared to DSSL. This is because due to the lack of HTTP acceleration with SSL, more long-duration TCP connections are established between the client and the server to retrieve all embedded web page objects. A sectional closeup of the TCP load in figure 6.25(b) indicates that the TCP load are approximately similar in all DSSL versions, and comparable to that for unsecured browsing, since they all can take advantage of HTTP acceleration.

6.4 Summary

In this chapter, we have described how the high propagation delay and large bandwidth of the satellite link adversely affects the performance of the TCP protocol. We



(a) Client TCP load for DSSL, SSL and unsecured browsing.



(b) Sectional closeup of figure 6.25(a)

Figure 6.25: Comparison of TCP load at client for multiple servers, multiple sessions. *X-axis is the simulation time in minutes; Y-axis is the load in bytes/second.*

have also described how web browsing over the satellite link incurs heavy delay due to HTTP protocol using serial GET requests to retrieve individual objects in a webpage. We have described how satellite networks use TCP performance-enhancing proxy servers at the hub and each remote location to enhance performance through TCP spoofing, and use HTTP-proxy servers at the hub and remote locations to greatly speed up web browsing. We explain how the use of IPSEC for network-layer security break the function of the TCP proxy by encrypting the network headers with keys known only to the end-points. Similarly, using SSL for secure web browsing break HTTP proxy function by encrypting the HTTP segment with keys known only to the end-points.

We have proposed the use of the Layered IPSEC protocol as an alternative to

IPSEC. Layered IPSEC uses different keys to encrypt different regions of the packet header and distributes relevant keys to the proxy servers, so that they can read sections of the header and perform their functions. The key distribution can be done either manually (pre-shared keys) or dynamically using IKE. We have proposed a modification to the IKE protocol so that it can work with Layered IPSEC. Through simulation experiments, we have demonstrated that Layered IPSEC can lead to significant improvement in performance as measured by the end-to-end delay when pre-shared keys are used. However, the experiments also show that when Layered IPSEC is used with the modified IKE protocol, the overhead due to the IKE key distribution neutralizes the advantage of using Layered IPSEC.

We have also proposed the DSSL protocol as an alternative to using SSL for secure web browsing. The DSSL protocol splits the single secure SSL channel into a primary channel (with the encryption keys known only to the end-points) and a secondary channel (with the encryption keys known to the end-points and the HTTP proxy servers). The embedded object links for the requested webpage are placed in the secondary channel by the web server, so that the HTTP proxy can read the links and pre-fetch the objects from the web server, thereby performing its function. Through simulation experiments, we have shown that the DSSL protocol can lead to substantial improvement in the web browsing delay. However, the improvement depends to a great extent on the handhshake method used amongst the client, the web server and the proxy to establish the secure DSSL channels. We have therefore proposed several variations to the DSSL handshake which trade-off the delay performance with the security of the handshake.

6.5 Related Work

Olechna et al [105] have suggested two solutions to the IPSEC problem. In the first approach, the paper proposes moving the the TCP PEP gateways to the endpoints. The TCP optimizations are done on the traffic in the clear, and then the traffic is encrypted using IPSEC. There is no TCP PEP at the satellite hub. This approach improves the performance, but when a packet is lost or received in error, TCP goes into congestion avoidance phase and the transmission is reduced in half. The second proposed approach which deals effectively with this problem is to split the secure connection into two at the satellite gateway. One connection is between the client and the gateway, and the second connection is between the gateway and the Internet server. This allows the gateway to decrypt the IPSEC packet and read the headers and thereby do performance optimizations. This requires trust in the satellite gateway, which can now read all the traffic. This might be unacceptable to users who require strong end-to-end security.

Several modified TCP protocols have been proposed that perform better compared to the original specification in the event of channel errors or delay, or when IPSEC is used. A discussion of these TCP enhancements can be found in [106].

The problem of HTTP proxy performance when SSL is used has been addressed by the industry by breaking up the end-to-end single SSL connection between client and server into multiple SSL connections [107]. In this solution, the client browser creates a secure HTTP connection with the Remote Page Accelerator (RPA) at the client satellite terminal, a second connection is created between the RPA and the Hub Page Accelerator (HPA), and a third connection is between the HPA and the server. The RPA performs all necessary handshaking with the client browser. The HPA can decrypt the SSL traffic from the server and perform the desired object pre-fetching. Taken together, this allows delivery of secure web content with little performance degradation and with little change to the standard protocols. The major drawback to this scheme is that it requires a high level of trust in intermediate nodes. The HPA, which is a third party entity, can read all the sensitive web traffic that passes between the client and the server. This might be unacceptable when absolute end-to-end security is desired.

Chapter 7

Conclusions

7.1 Dissertation Contributions

In this dissertation we have investigated whether the addition of a satellite overlay interconnecting large terrestrial wireless networks, results in:

- improvement in network performance as measured by certain parameters,
- efficient solutions for source authentication in group communication, and
- advantages or drawbacks for secure unicast communication over the satellite link.

Towards this objective, we have made the following contributions.

In chapter 2, we have described toplogies for hybrid satellite/wireless networks and their important applications. We have proved through network modeling and simulation, that the addition of a satellite overlay network to a large wireless network provides significant improvement in end-to-end performance and reliability for different traffic types, network sizes and communicating parties, amongst other parameters. We have developed comprehensive network models for different application scenarios that can be used for further studies on hybrid wireless networks with a satellite overlay. Sections of this work has appeared in [108].

In chapter 3, we have designed a loss network model for hybrid satellite/wireless networks and proposed a new method for performance analysis of such networks, by combining the loss network models for the MAC and PHY layers, with routing, through fixed point iterations. By integrating the proposed fixed point model with the technique of Automatic Differentiation, we have demonstrated a method to compute the sensitivities of the performance metrics with respect to design parameters. We are thus able to perform hybrid network design by finding optimal routing strategies in the hybrid network. We have demonstrated the validity of our model and its computational advantage by performing simulations and comparing with Opnet discrete event simulations. Sections of this work has appeared in [109].

In chapter 4, we have proposed an algorithm for efficient authentication of wireless mobile nodes that have constraints on energy and processing power. The proposed algorithm takes advantage of the wide reach and security of the satellite overlay network by delegating most energy-intensive operations to the satellite node and allowing a source node to send authenticated messages to multiple receiver nodes in a single-hop satellite broadcast transmission. We have also proposed a new method for probabilistic non-repudiation that also makes use of having the satellite node as a proxy for source nodes in the network. We have demonstrated the security properties and correctness of the authentication protocol through analysis, and through simulations have shown the advantage of the proposed protocol for wireless settings, in comparison to traditional source authentication protocols. Part of this work will appear in [110] and parts have also been published in [111].

In chapter 5, we have described a hierarchical multicast routing protocol for efficient group communication in hybrid wireless networks. The proposed protocol has several advantages over traditional multicast routing protocols for wireless ad hoc networks, since it takes into consideration the different characteristics of the terrestrial and satellite links. In particular, the protocol minimizes the amount of control traffic that flows over the satellite links to mitigate the negative impact of the satellite link delay. The proposed protocol localizes the multicast distribution tree only to the terrestrial LANs that have multicast group sources and/or members, while providing for an efficient mechanism to allow LANs to dynamically join or leave the group. The protocol also makes efficient use of the multi-path routing provided by the terrestrial links and the satellite overlay, and integrates mechanisms so that the distribution tree can select high-bandwidth paths based on the application traffic rate. Furthermore, the protocol provides mechanisms to re-connect group members to the network in the event of network partitions, and is robust to link failures. Previously, we have proposed a hierarchical multicast routing protocol for hybrid wired network with an ATM satellite overlay [112]. The wireless multicast protocol proposed is an enhancement of the wired protocol, applied to the more difficult case of terrestrial wireless clusters.

In chapter 5, we have also described a protocol for secure multicast routing in hybrid networks. The proposed secure routing protocol is designed by integrating the authentication algorithm from chapter 4 with the proposed multicast routing protocol. Through analysis, we have demonstrated the security properties and correctness of the proposed secure multicast protocol.

In chapter 6, we have investigated the adverse effects of security protocols IPSEC and SSL on unicast communication over satellite networks. As a solution to the problem, we have proposed the use of layered IPSEC with modified IKE for securing unicast communication while allowing performance optimization algorithms to function simultaneously. Through simulations, we have evaluated the performance of the layered IPSEC protocol in a hybrid satellite network, and demonstrated that the performance compares favorably to standard IPSEC when implemented in hybrid networks. We have also proposed the DSSL protocol to replace SSL for secure HTTP in hybrid satellite networks. Through simulations we have shown that the performance of DSSL protocol is significantly better than SSL in hybrid networks. To mitigate the heavy handshake overhead in DSSL, we have also proposed two variations to the protocol - the DSSL two phase, which reduces the handshake to only two stages of the full DSSL protocol, and the DSSL quick mode, which further reduces the handshake to only one stage. We have shown that both DSSL two phase and DSSL quick mode protocol perform much better compared to SSL. Sections of this work have been mentioned in [111].

7.2 Future Work

Although this dissertation has covered several important problems for communication support in hybrid satellite/wireless communication networks, there remain some issues that need to be addressed. In the following section, we highlight some of these problems for future work on this topic.

- In chapter 2, we have built simulation models in Opnet Modeler to demonstrate the advantages of hybrid networks. In the future, we plan to conduct simulation studies with the Opnet network models to analyze the percentage of network availability as the number of gateway nodes are varied. Also, we would like to investigate the reliability afforded by a given overlay network by varying the number of terrestrial node failures so that the terrestrial forwarding paths become unavailable. We plan to model the simulation scenario where source-destination pairs are using terrestrial forwarding paths, but the forwarding path becomes unavailable due to intermediate node failures, prompting the source to send out new route requests. New routes are available over the overlay network, prompting the gateway nodes to reply to the route request and re-establishing the path. We will investigate how the end-to-end application characteristics are affected by this path re-establishment.

We believe it would be worthwhile to develop algorithms that compute the minimum number of gateway nodes required for full network connectivity for the network models in chapter 2. Our current models use the UAV placement algorithm developed in [4] to calculate the number of gateway nodes required and their placement. The algorithm finds the minimum number of gateway nodes (UAVs) such that at least one node from each cluster would be able to reach one UAV. The algorithm identifies clusters based on physical partitions, and does not limit the number of hops within any cluster. We plan to develop an algorithm to find the minimum number of gateway nodes such that no user is more than k hops away from a gateway node (where k is a system parameter). This would be useful in cases where physical partitions don't exist, so a different parameter is needed to partition the network into clusters served by gateway nodes (and the associated satellite overlay network).

We would like to investigate the dynamic allocation of mobile gateway nodes based on the traffic characteristics in the network. The UAV placement algorithm [4] used in our network model allocates the gateway nodes statically based on the number of clusters. However, if only a small subset of the clusters are involved in the communication at any given time, a smaller number of gateway nodes would suffice. Therefore, we will investigate whether we can achieve comparable performance with a smaller number of mobile gateway nodes, by moving around the gateway nodes to cover only the clusters that are involved in the communication.

- In chapter 3, we would like to investigate other loss network models using additional modules for MAC and routing for both the terrestrial wireless clusters and the satellite overlay. This will include the hidden node problem in the MAC layer. For the satellite channel, more realistic channel error models need to be considered and also the effect of satellite link delay on the performance, especially for Ka-band satellite networks. For the performance metrics, we plan to derive results for delay and buffer over-flow. We also would like to perform scalability analysis for large networks, convergence proofs, and accuracy bounds on various metrics.

- In chapter 5, we have given a description of the multicast routing protocol described for hybrid networks. We would like to model the routing protocol in simulation software and evaluate its performance. We would like to compare the performance to basic MAODV, and other multicast routing protocols proposed for wireless networks. We plan to include various scenarios in the simulation models, including testing the robustness of the protocol with node failures and network partitions and dealing with node mobility.

  We would also like to build a model for the secure multicast routing protocol as described in chapter 5. We would like to evaluate its performance through simulations and quantify the overhead in secure routing by comparing with the base hierarchical multicast routing protocol. It would also be worthwhile to investigate further security additions to the multicast routing protocol to protect against *Active-1-x* and *Active-y-x* attackers, and to deal with the case of users moving from one terrestrial network to another.

Bibliography

[1] P. Gupta and P.R. Kumar. The capacity of wireless networks. *IEEE Transactions on Information Theory*, 46(2):388–404, March 2000.

[2] E. J. Fitzpatrick. Spaceway system summary. *Space Communications*, (13):7–23, 1995.

[3] Opnet Modeler. http://www.opnet.com/products/modeler/home.html.

[4] K. Chandrashekar, M. Raissi-Dekhordi, and J. S. Baras. Providing full connectivity in large ad-hoc networks by dynamic placement of aerial platforms. In *Proc. of IEEE Military Communications Conference 2004 (MILCOM 2004)*, Monterey, USA, October 31 - November 3 2004. IEEE.

[5] P. Gupta and P.R. Kumar. Internets in the sky: the capacity of three dimensional wireless networks. *Communications in Information Systems*, 1(1):33–50, 2001.

[6] T.A. Elbatt and A. Ephremides. Optimization of connection-oriented, mobile, hybrid network systems. *IEEE Journal on Selected Areas in Communications*, 17(2):373–384, February 1999.

[7] O.Dousse, P. Thiran, and M. Hasler. Connectivity in ad-hoc and hybrid networks. In *Proc. IEEE Infocom 2002*, pages 1079–1088, 2002.

[8] B. Ryu, T. Andersen, T. Elbatt, and Y. Zhang. Multi-tier mobile ad hoc networks: architecture, protocols, and performance. In *Military Communications Conference (MILCOM) 2003*, volume 2, pages 1280–1285. IEEE, 2003.

[9] F.-G. Wu, F.-C. Sun, K. Yu, and C.-W. Zheng. Performance evaluation on a double-layered satellite network. *International Journal of Satellite Communications and Networking*, 23:359–371, 2005.

[10] D. Grace, M. H. Capstick, M. Mohoric, J. Horwath, M. B. Pallavicini, and M. Fitch. Integrating users into the wider broadband network via high altitude platforms. *IEEE Wireless Communications*, pages 98–105, October 2005.

[11] B. Evans, M. Werner, E. Lutz, M. Bousquet, G. E. Corazza, G. Maral, R. Rumeau, and E. Ferro. Integration of satellite and terrestrial systems in future multimedia communications. *IEEE Wireless Communications*, pages 72–80, October 2005.

[12] J.G. Jetcheva, Y.-C. Hu, S. Palchaudhuri, A.K. Saha, and D.B. Johnson. Design and evaluation of a metropolitan area multitier wireless ad hoc network architecture. In *Proceedings of the Fifth IEEE Workshop on Mobile Computing Systems and Applications (WMCSA 2003)*. IEEE Computer Society, 2003.

[13] G. Bianchi. Performance analysis of the IEEE 802.11 distributed coordination function. *IEEE Journal of Selected Areas in Communications*, 18(3):535–547, March 2000.

[14] E. N. Gilbert. Capacity of a bursty-noise channel. *Bell System Technical Journal*, 39(9):1253–65, September 1960.

[15] E. O. Elliot. Estimates of error rates for codes on bursty noise channels. *Bell System Technical Journal*, 42(9):1977–97, September 1963.

[16] M. Liu. *Network Performance Modeling, Design and Dimensioning Methodologies*. PhD thesis, University of Maryland College Park, 2000.

[17] M. Bucker, G. Corliss, P. Hovland, and U. N. Boyana Norris. *Automatic Differentiation: Applications, Theory and Implementations*, volume 50 of *Lecture Notes in Computational Science and Engineering*. Springer, Berlin, Germany, 2006.

[18] J. Baras, V. Tabatabaee, G. Papageorgiou, and N. Rentz. Performance metric sensitivity computation for optimization and trade-off analysis in wireless networks. Institute for systems research technical report, University of Maryland College Park, http://hdl.handle.net/1903/7547, 2008.

[19] J. Baras, V. Tabatabaee, G. Papageorgiou, N. Rentz, and Y. Shang. Loss model approximations and sensitivity computations for wireless network design. In *Military Communications Conference (MILCOM) 2007*, Orlando, Florida, October 29-31 2007. IEEE.

[20] J. Slack and M. Rice. Finite state Markov models for error bursts on the ACTS land mobile satellite channel. NASA ACTS Experiment Paper ACTS-96-046, NASA, 1996.

[21] HUGHES DIRECWAY system. http://www.direcway.com.

[22] J. Foerster and J. Liebetreu. FEC performance of concatenated Reed-Solomon and convolutional coding with interleaving. Ieee 802.16 broadband wireless access working group report, IEEE, June, 2000.

[23] N. Abramson. The throughput of packet broadcasting channels. *IEEE Transactions on Communications*, COM-25(1):117–128, January 1977.

[24] N. Abramson. Multiple access in wireless digital networks. *Proceedings of the IEEE*, 82(9):1360–1369, September 1994.

[25] D. R. Smart. *Fixed Point Theorems*. Cambridge University Press, London, U.K., 1974.

[26] University of Chicago Argonne National Laboratory. ADIC resource center. http://www-new.mcs.anl.gov/adic.

[27] S. E. Dreyfus. An appraisal of some shortest path algorithms. *Operations Research*, 17(3):395–412, May 1969.

[28] F. P. Kelly. Loss networks. *The Annals of Applied Probability*, 1(3):319–378, August 1991.

[29] F. P. Kelly. Blocking probabilities in large circuit switched networks. *Advances in Applied Probability*, 18(2):473–505, June 1986.

[30] D. Mitra, J. A. Morrison, and K. G. Ramakrishnan. ATM network design and optimization: A multirate loss network framework. *IEEE/ACM Transactions on Networking*, 4(4), August 1996.

[31] S. Chung, A. Kashper, and K. W. Ross. Computing approximate blocking probabilities for large loss networks with state-dependent routing. *IEEE/ACM Transactions on Networking*, 1(1), February 1993.

[32] A. G. Greenberg and R. Srikant. Computational techniques for accurate performance evaluation of multirate, multihop communication networks. *IEEE Journal on Selected Areas in Communications*, 5(2):266–277, February 1997.

[33] M. Liu and J. S. Baras. Fixed point approximation for multirate multihop loss networks with adaptive routing. *IEEE/ACM Transactions on Networking*, 12(2):361–374, April 2004.

[34] L.L. Dai and V.W.S. Chan. Capacity dimensioning and routing for hybrid satellite and terrestrial systems. *IEEE Journal on Selected Areas in Communications*, 22(2):287–299, February 2004.

[35] H. Krawczyk, M. Bellare, and R. Canetti. *HMAC: Keyed-Hashing for Message Authentication*. IETF RFC 2104, February 1997.

[36] N.I.S.T. Digital signature standard (dss), May 19 1994.

[37] P.R.Zimmermann. *The official PGP user's guide*. MIT Press, May 3 1995.

[38] R. Housley, W. Ford, W. Polk, and D. Solo. *Internet X.509 Public Key Infrastructure Certificate and CRL Profile*. IETF Network Working Group RFC 2459, http://www.ietf.org/rfc/rfc2459.txt, January 1999.

[39] V. Gupta, S. Gupta, and S. Chang. Performance analysis of Elliptic Curve Cryptography for SSL. In *Proceedings of the ACM Wireless Internet Security Workshop (WiSe'02)*, Atlanta, USA, September 28 2002. ACM.

[40] P. Prasithsangaree and P. Krishnamurthy. On a framework for energy-efficient security protocols in wireless networks. *Elsevier Computer Communications*, 27:1716–1729, 2004.

[41] S. Seys and B. Preneel. Power consumption evaluation of efficient digital signature schemes for low power devices. In *Proc. 2005 IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (IEEE WiMOb 2005)*, volume 1, pages 79–86. IEEE, 2005.

[42] W. Freeman and E. Miller. An experimental analysis of cryptographic overhead in performance-critical systems. In *Proc. 7th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOT'99)*, pages 348–357, College Park, MD, USA, October 1999. IEEE.

[43] M. Bohge and W. Trappe. TESLA certificates: an authentication tool for networks of compute-constrained devices. In *Proc. of 6th international symposium on wirless personal multimedia communications (WPMC '03)*, Yokosuka, Kanagawa, Japan, October 2003.

[44] A. Roy-Chowdhury and J.S. Baras. A certificate-based light-weight authentication algorithm for resource-constrained devices. Technical Report CSHCN TR 2005-4, Center for Satellite and Hybrid Communication Networks, University of Maryland College Park, 2005.

[45] A. Perrig, R. Canetti, D. Song, and J. D. Tygar. The TESLA broadcast authentication protocol. *RSA Cryptobytes*, Summer 2002.

[46] A. Perrig, R. Canetti, D. Song, and J. D. Tygar. Efficient and secure source authentication for multicast. In *Proc. Network and Distributed System Security Symposium (NDSS)*, 2001.

[47] M. Naor and M. Yung. Universal one-way hash functions and their cryptographic applications. In *STOC '89: Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 33–43, New York, NY, USA, 1989. ACM.

[48] M. Bohge and W. Trappe. An authentication framework for hierarchical ad hoc sensor networks. In *Proceedings of the 2003 ACM Workshop on Wireless Security (WiSE'03)*, pages 79–87, San Diego, USA, August 2003. ACM.

[49] F. Fabrega, J.C. Herzog, and J.D. Guttman. Strand spaces: Why is a security protocol correct. In *Proc. 1998 IEEE Symposium on Security and Privacy*, Oakland, California, May 3-6 1998. IEEE.

[50] J.D. Guttman and F. Fabrega. Authentication tests. In *Proc. 2000 IEEE Symposium on Security and Privacy*, Oakland, California, May 2000. IEEE.

[51] G. Lowe. A hierarchy of authentication specifications. In *Proc. 1997 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 31–43, 1997.

[52] Secure Hash Standard. http://www.itl.nist.gov/fipspubs/fip180-1.htm.

[53] D.L. Mills. *RFC 1305 - Network Time Protocol (Version 3) Specification, Implementation and Analysis*. Internet Engineering Task Force (IETF), March 1992.

[54] J. Jonsson and B. Kaliski. *Public-Key Cryptography Standards (PKCS) # 1: RSA Cryptography Specifications Version 2.1 (RFC 3447)*. Internet Engineering Task Force (IETF), February 2003.

[55] National Institute of Standards and Technology (NIST). Digital signature standard (DSS). Technical Report FIPS 186-3, Information Technology Laboratory, NIST, March 2006.

[56] X. Ding, D. Mazzocchi, and G. Tsudik. Equipping smart devices with public key signatures. *ACM Trans. Internet Technology*, 7(1):3, 2007.

[57] Compaq iPAQ Pocket PC H3600 series. http://h18002.www1.hp.com/products/quickspecs/10632_div/10632_div.HTML#QuickSpecs.

[58] N.R. Potlapally, S. Ravi, A. Raghunathan, and N.K. Jha. A study of the energy consumption characteristics of cryptographic algorithms and security protocols. *Mobile Computing, IEEE Transactions on*, 5(2):128–143, Feb. 2006.

[59] ARM1176JZ(F)-S processor. http://www.arm.com/products/CPUs/ARM1176.html.

[60] iPod and iPhone battery and power specifications. http://www.ipodbatteryfaq.com/ipodbatteryandpower.html.

[61] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas. Multicast security: A taxonomy and some efficient constructions. *Proceedings of INFOCOMM '99*, March 1999.

[62] R. Gennaro and P. Rohatgi. How to sign digital signatures. In *Advances in Cryptology - CRYPTO '97*, pages 180–197. Springer-Verlag Berlin Heidelberg, 1997.

[63] C.K. Wong and S.S. Lam. Digital signatures for flows and multicasts. In *Proc. IEEE ICNP '98*, Austin, USA, October 1998. IEEE.

[64] P. Rohatgi. A compact and fast hybrid signature scheme for multicast packet authentication. In *Proc. Computer and Communications Security Conference (CCS'99)*, Singapore, 1999. ACM.

[65] R. Anderson, F. Bergadano, B. Crisp, J. Lee, C. Manifavas, and R. Needham. A new family of authentication protocols. *ACM Operating Systems Review*, 32(4):9–20, 1998.

[66] A. Perrig. The BiBa one-time signature and broadcast authentication protocol. *Proceedings of the 8th ACM Conference on Computer and Communications Security - CCS '01*, November 2001.

[67] E. Bommaiah, A. McAuley, R. Talpade, and M.-K. Liu. *AMRoute: Adhoc Multicast Routing Protocol*. IETF Internet Draft, draft-talpade-manet-amroute-00.txt, August 1998.

[68] C.W. Wu, Y.C. Tay, and C.-K. Toh. *Ad hoc multicast routing protocol utilizing increasing id-numbers (AMRIS) functional specification*. IETF Internet Draft, draft-ietf-manet-amris-spec-00.txt, November 1998. Work in Progress.

[69] E.M. Royer and C.E. Perkins. Multicast operation of the ad-hoc on-demand distance vector routing protocol. In *ACM Mobicom*, Seattle, USA, August 1999. ACM.

[70] M. Gerla, S.-J. Lee, and W. Su. *On-demand multicast routing protocol (ODMRP) for ad hoc networks*. IETF Internet Draft, draft-ietf-manet-odmrp-02.txt, January 2000.

[71] Y.-C. Hu, A. Perrig, and D.B. Johnson. Ariadne: A secure on-demand routing protocol for ad hoc networks. In *Proceedings of the Eighth Annual International Conference on Mobile Computing and Networking (MobiCom 2002)*, pages 12–23, Atlanta, USA, September 23-26 2002. ACM.

[72] C. E. Perkins and E. M. Royer. Ad-hoc on-demand distance vector routing. In *The Second IEEE Workshop on Mobile Computing Systems and Applications*, New Orleans, USA, February 1999.

[73] K. Sanzgiri, B. Dahill, B.N. Levine, C. Shields, and E.M. Belding-Royer. A secure routing protocol for ad hoc networks. In *Proceedings of the 10th IEEE International Conference on Network Protocols (ICNP'02)*, 2002.

[74] D.B. Johnson, D.A. Maltz, and Y.-C. Hu. *The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR)*. Internet Draft, draft-ietf-manet-dsr-09.txt, April 15 2003.

[75] M. Guerrero Zapata. Secure ad hoc on-demand distance vector routing. *Mobile Computing and Communications Review*, 6(3):106–107, July 2002.

[76] S. Roy, V. Gopala Addada, S. Setia, and S. Jajodia. Securing MAODV: Attacks and coutermeasures. In *Proceedings of 2nd IEEE Communications Society Conference on Sensor and Ad hoc Communications and Networks (SECON 05)*, Santa Clara, USA, September 2005. IEEE.

[77] Y.-C. Hu, D.B. Johnson, and A. Perrig. SEAD: secure efficient distance vector routing for mobile wireless ad hoc networks. *Ad Hoc Networks*, 1:175–192, 2003.

[78] L. Lamport, R. Shostak, and M. Pease. The byzantine generals problem. *ACM Transactions on Programming Languages*, 4(3):382–401, July 1982.

[79] C. Gui and P. Mohapatra. Scalable multicasting in mobile ad hoc networks. In *Proceedings of IEEE INFOCOM '04*, pages 2119–2129, Hong Kong, China, March 7-11 2004.

[80] L. Ji and M.S. Corson. Differential destination multicast-a manet multicast routing protocol for small groups. In *Proceedings of INFOCOM 2001 - Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 2, pages 1192–1201, Anchorage, USA, April 22-26 2001. IEEE.

[81] Y.-B. Ko, S.-J. Lee, and K.-Y. Lee. A multicast protocol for physically hierarchical ad hoc networks. In *Proc. IEEE VTC*, Jeju, Korea, April 2003.

[82] J.G. Jetcheva and D.B. Johnson. Adaptive demand-driven multicast routing in multi-hop wireless ad hoc networks. In *Proceedings of the 2nd ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pages 33–44, Long Beach, CA, USA, 2001. ACM, ACM Press.

[83] R. Bhatia, L.E. Li, H. Luo, R. Ramjee, and S. Paul. ICAM: Integrated cellular and ad-hoc multicast. Technical Report UCLA-CSD-TR040026, University of California, Los Angeles, June 2004.

[84] S. Yi, P. Naldurg, and R. Kravets. Security-aware ad-hoc routing for wireless networks. Technical Report UIUCDCS-R-2001-2241, UILU-ENG-2001-1748, Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL, USA, August 2001.

[85] P. Papadimitratos and Z. J. Haas. Secure routing for mobile ad hoc networks. In *Proceedings of the SCS Communication Networks and Distributed Systems Modeling Simulation Conference (CNDS 2002)*, San Antonio, TX, January 27-31 2002.

[86] R. Bobba, L. Eschenauer, V. Gligor, and W. Arbaugh. Bootstrapping security associations for routing in mobile ad-hoc networks. In *IEEE Globecom 2003*, San Fransisco, USA, December 1-5 2003.

[87] M. P. Howarth, S. Iyengar, Z. Sun, and H. Cruickshank. Dynamics of key management in secure satellite multicast. *IEEE Journal on Selected Areas in Communications*, 22(2):308–319, February 2004.

[88] C.K. Wong, M. Gouda, and S. S. Lam. Secure group communications using key graphs. *IEEE/ACM Transactions on Networking*, 8:16–30, February 2000.

[89] Y. Zhang. A multilayer IP security protocol for TCP performance enhancement in wireless networks. *IEEE Journal on Selected Areas in Communications*, 22(4):767–776, May 2004.

[90] H. Cruickshank, M.P. Howarth, S. Iyengar, Z. Sun, and L. Claverotte. Securing multicast in DVB-RCS satellite systems. *IEEE Wireless Communications*, pages 38–45, October 2005.

[91] ETSI. *Digital Video Broadcasting (DVB): Interaction Channel for Satellite Distribution Systems*. ETSI EN 301790 v. 1.3.1, March 2003.

[92] L. Duquerroy, S. Josset, O. Alphand, P. Berthou, and T. Gayraud. SatIPSec: an optimized solution for securing multicast and unicast satellite transmissions. In *22nd AIAA International Communications Satellite Systems Conference and Exhibit 2004*, number AIAA-2004-3177, Monterey, California, 9-12 May 2004.

[93] R. Atkinson and S. Kent. *Security Architecture for the Internet Protocol*. IETF RFC 2401, November 1998.

[94] J. Kong, H. Luo, K. Xu, D. L. Gu, M. Gerla, and S. Lu. Adaptive security for multilayer ad-hoc networks. *Special Issue of Wireless Communications and Mobile Computing, John Wiley InterScience Press*, 2002.

[95] J. Border, M. Kojo, J. Griner, G. Montenegro, and Z. Shelby. *Performance enhancing proxies intended to mitigate link-related degradations*. IETF RFC3135, June 2001.

[96] V. Arora, N. Suphasindhu, J. Baras, and D. Dillon. Effective extensions of internet in hybrid satellite-terrestrial networks. Technical Report CSHCN TR 96-2, University of Maryland College Park, 1996.

[97] V. Bharadwaj. Improving TCP performance over high-bandwidth geostationary satellite links. Technical Report ISR TR MS-99-12, University of Maryland College Park, 1999.

[98] N. Ehsan, M. Liu, and R. Ragland. Evaluation of performance enhancing proxies in internet over satellite. *Wiley Int. J. Communication Syst.*, 16:513–534, August 2003.

[99] IETF Transport Layer Security Working Group, http://wp.netscape.com/eng/ssl3/draft302.txt. *The SSL Protocol Version 3.0*, November 1996.

[100] R. Atkinson and S. Kent. *IP Encapsulating Security Payload (ESP)*. IETF RFC 2406, November 1998.

[101] R. Atkinson and S. Kent. *IP Authentication Header*. IETF RFC 2402, November 1998.

[102] M. Karir and J.S. Baras. LES: Layered encryption security. In *Proceedings of the Third International Conference on Networking (ICN'04)*, Guadeloupe, French Caribbean, March 2004.

[103] D. Harkins and D. Carrel. *The Internet Key Exchange (IKE)*. IETF RFC 2409, November 1998.

[104] Y. Song, V.C.M. Leung, and K. Beznosov. Supporting end-to-end security across proxies with multiple-channel SSL. In *Proceedings of the 19th IFIP Information Security Conference (SEC 2004)*, pages 323–337, Toulouse, France, August 23-26 2004. IFIP.

[105] E. Olechna, P. Feighery, and S. Hryckiewicz. Virtual private network issues using satellite based networks. In *Military Communications Conference (MILCOM) 2001*, volume 2, pages 785–789, 2001.

[106] P. Chitre, M. Karir, and M. Hadjitheodosiou. TCP in the IPSEC environment. In *22nd AIAA International Communications Satellite Systems Conference and Exhibit (ICSSC) 2004*, Monterey, California, May 2004. AIAA.

[107] Spacenet Inc., http://www.spacenet.com/technology/advantages/ssl.html. *SSL Accelerator*.

[108] A. Roy-Chowdhury and J. S. Baras. Improving network performance in hybrid wireless networks using a satellite overlay. In *Proc. 13th Ka and Broadband Communications Conference*, Turin, Italy, September 24-26 2007. Istituto Internazionale delle Comunicazioni (IIC).

[109] A. Roy-Chowdhury, , V. Tabatabaee, and J. S. Baras. Performance modeling of hybrid satellite/wireless networks using fixed point approximation and sensitivity analysis of the performance models for network design. In *Proc. 14th Ka and Broadband Communications Conference*, Matera, Italy, September 24-26 2008. Istituto Internazionale delle Comunicazioni (IIC).

[110] A. Roy-Chowdhury and J.S. Baras. A lightweight certificate-based source authentication protocol for group communications in hybrid wireless/satellite networks. In *Proc. IEEE Global Communications Conference (Globecom) 2008*, New Orleans, LA, USA, November 30 - December 4 2008. IEEE.

[111] A. Roy-Chowdhury, J. Baras, M. Hadjitheodosiou, and S. Papademetriou. Security issues in hybrid satellite networks. *IEEE Wireless Communications*, 12(6):50–61, December 2005.

[112] A. Roy-Chowdhury and J.S. Baras. Framework for IP multicast in satellite ATM networks. In *22nd AIAA International Communications Satellite Systems Conference and Exhibit (ICSSC) 2004*, number AIAA-2004-3176, Monterey, California, USA, May 9-12 2004.