Adaptive Security-Aware Scheduling for Packet Switched Networks Using Real-Time Multi-Agent Systems

Ma'en Saleh Saleh

Western Michigan University, maen_saleh@yahoo.com

Follow this and additional works at: http://scholarworks.wmich.edu/dissertations

Part of the OS and Networks Commons, Systems and Communications Commons, and the Systems Architecture Commons

Recommended Citation

http://scholarworks.wmich.edu/dissertations/41

This Dissertation-Open Access is brought to you for free and open access by the Graduate College at ScholarWorks at WMU. It has been accepted for inclusion in Dissertations by an authorized administrator of ScholarWorks at WMU. For more information, please contact maira.bundza@wmich.edu.
ADAPTIVE SECURITY-AWARE SCHEDULING FOR PACKET SWITCHED NETWORKS USING REAL-TIME MULTI-AGENT SYSTEMS

by
Ma’en Saleh Saleh

A Dissertation
Submitted to the
Faculty of the Graduate College
in partial fulfillment of the
requirements for the
Degree of Doctor of Philosophy
Department of Electrical and Computer Engineering
Advisor: Liang Dong, Ph.D.

Western Michigan University
Kalamazoo, Michigan
June 2012
WE HEREBY APPROVE THE DISSERTATION SUBMITTED BY

Ma'en Saleh Saleh

ENTITLED Adaptive Security-Aware Scheduling for Packet Switched Networks Using Real-Time Multi-Agent Systems

AS PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

DEGREE OF Doctor of Philosophy

Electrical and Computer Engineering

(Department)

Electrical and Computer Engineering

(Program)

APPROVED

Dean of The Graduate College

Date June 2012
ADAPTIVE SECURITY-AWARE SCHEDULING FOR PACKET SWITCHED NETWORKS USING REAL-TIME MULTI-AGENT SYSTEMS

Ma’en Saleh Saleh, Ph.D.
Western Michigan University, 2012

Conventional real-time scheduling algorithms are in care of timing constraints; they don’t pay any attention to enhance or optimize the real-time packet’s security performance. In this work, we propose an adaptive security-aware scheduling with congestion control mechanism for packet switching networks using real-time agent-based systems. The proposed system combines the functionality of real-time scheduling with the security service enhancement, where the real-time scheduling unit uses the differentiated-earliest-deadline-first (Diff-EDF) scheduler, while the security service enhancement scheme adopts a congestion control mechanism based on a resource estimation methodology.

The security service enhancement unit was designed based on two models: single-layer and weighted multi-layer design models. For single-layer, the design provides an enhancement for a single security service: confidentiality, integrity, or authentication, while the weighted multi-layer design provides an enhancement for multiple security services with different weights on a real-time network with multi-processor end nodes. The proposed system provides the required QoS guarantees for different classes of real-time data flows (video, audio), while adaptively enhances the
packet’s security service levels according to a feedback from the congestion control model, which efficiently utilizes the buffering system at the edge network, and thus protects the network from being congested by heavy traffic load.

Our agent-based system eliminates the overhead of the security association phase performed by the internet protocol security (IPsec). Such elimination had been achieved by overloading the priority code point (PCP) fields of the IEEE 802.1Q tagged frame format for the single-layer scheme, while repeated single-layer and overloading the PCP and the virtual-LAN identifier (VID) fields of the IEEE 802.1Q were the adopted methodologies by the weighted multi-layer security design model.

By using the Diff-EDF scheduler, the proposed system minimizes the flows miss rates and the flows average total delays compared to the earliest-deadline-first (EDF) and the first-come-first-served (FCFS) schedulers. From the other hand, our adaptive security enhancement scheme minimizes the buffer consumption, the average total packet delays, and the pending packets at the end users compared to the IPsec protocol. It was also compared to an implemented feedback-IPsec, where our adaptive system eliminated the repeated security associations performed by the feedback-IPsec, hence less overhead and increases the chances to meet the flows QoS requirements.
Copyright by
Ma’en Saleh Saleh
2012
ACKNOWLEDGMENTS

First of all, I would like to thank god for providing me the strength and patience to fulfill my goals during my whole life. It’s his blessings that I am here today. My great thank and appreciation to my supervisor, Dr. Liang Dong, who guided me during all my research phases. Thank you for your continuous support, encouragement, and valuable recommendations. I would like also to thank my committee members Dr. Ikhlas Abdel-Qader, Dr. Janos Grantner, and Dr. Kapseong Ro for their great effort in supervising me to finish this dissertation.

I would like to thank my parents who always supported me during my study. Thank you for your continuous prayers, which was the key behind my success. Finally, I would like to thank my beloved wife, Sara’a, who shared me all my happy and sad moments. Thank you for your patience, support, and encouragement. For you and my daughters, Salma and Renad, I am dedicating this dissertation.

Ma’en Saleh Saleh
# TABLE OF CONTENTS

ACKNOWLEDGMENTS ........................................................................................................ ii
LIST OF TABLES .................................................................................................................. vii
LIST OF FIGURES ................................................................................................................ viii

CHAPTER

1. INTRODUCTION ........................................................................................................... 1
   1.1 Background and General Overview ................................................................. 1
   1.2 Problem Statement ............................................................................................. 2
   1.3 Research Goals ................................................................................................... 6
   1.4 Dissertation Outline ........................................................................................... 8

2. LITERATURE REVIEW .............................................................................................. 11
   2.1 General Overview ................................................................................................. 11
   2.2 Real-time Applications ....................................................................................... 12
      2.2.1 Properties of Real-time Applications .......................................................... 13
      2.2.2 Classification of Real-time Applications ..................................................... 15
   2.3 Real-time Networks ............................................................................................. 18
      2.3.1 Quality of Service for Real-time Networks (QoS) ......................................... 19
      2.3.2 Real-time Network Performance Metrics (NPMs) ...................................... 21
   2.4 Buffer Estimation Techniques for Real-time Networks ....................................... 24
   2.5 Real-time Network Scheduling .......................................................................... 26
      2.5.1 Types of Network Scheduling Algorithms .................................................. 27
<table>
<thead>
<tr>
<th>CHAPTER</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.5.2 Network Scheduling using Multi-agent Systems</td>
</tr>
<tr>
<td>2.6 Real-time Network Security</td>
</tr>
<tr>
<td>2.6.1 Real-time Network Security Services</td>
</tr>
<tr>
<td>2.6.2 Network Security-aware Scheduling</td>
</tr>
<tr>
<td>3. SYSTEM DESIGN MODEL USING REAL-TIME AGENT-BASED METHODOLOGY</td>
</tr>
<tr>
<td>3.1 Introduction</td>
</tr>
<tr>
<td>3.2 Object-oriented Agent-based Design Model</td>
</tr>
<tr>
<td>3.2.1 Source Agent</td>
</tr>
<tr>
<td>3.2.2 Coordinator Agent</td>
</tr>
<tr>
<td>3.2.3 Diff-EDF Scheduler Agent</td>
</tr>
<tr>
<td>3.2.4 Queue Agent</td>
</tr>
<tr>
<td>3.2.5 Server Agent</td>
</tr>
<tr>
<td>3.2.6 Destination Agent</td>
</tr>
<tr>
<td>3.3 Security Service Design Models</td>
</tr>
<tr>
<td>3.3.1 Single-layer Security Service Design Model</td>
</tr>
<tr>
<td>3.3.2 Weighted Multi-layer Security Service Design Model</td>
</tr>
<tr>
<td>3.4 Evaluating System Parameters</td>
</tr>
<tr>
<td>3.4.1 Evaluating Ethernet Packet Size</td>
</tr>
<tr>
<td>3.4.2 Evaluating the Average Aggregate Bandwidth</td>
</tr>
</tbody>
</table>
CHAPTER

4. IMPLEMENTATION OF REAL-TIME SECURIT-AWARE SCHEDULER FOR PACKET SWITCHED NETWORKS ....................... 73
   4.1 Introduction ................................................................................... 73
   4.2 Brownian Motion Queuing Model for Workload Process Implementation .............................................................................. 75
   4.3 System Methodology ..................................................................... 78
      4.3.1 Real-time Multi-agent Timing Protocol for Scheduling ..... 79
      4.3.2 Real-time Multi-agent Security Enhancement Timing Protocol for Single and Weighted Multi-layer Models...... 83
      4.3.3 Network Congestion Feedback Mechanism ....................... 87

5. IEEE 802.1Q FRAME FORMAT FOR ADAPTIVE SECURITY-AWARE NETWORKS ................................................................. 100
   5.1 Introduction ................................................................................... 100
   5.2 Static Security Protocols and the Implementation of Feedback IPsec Protocol for Packet Switched Networks ...................... 101
   5.3 Overloading IEEE 802.1Q Frame Format for Single and Weighted Multi-Layer Security Design Models ......................... 104
      5.3.1 IEEE 802.1Q for Single Layer Security Model ................. 105
      5.3.2 IEEE 802.1Q for Weighted Multi-layer Security Model ... 107

6. SYSTEM SIMULATION AND NUMERICAL RESULTS ...................... 112
# Table of Contents–Continued

## CHAPTER

6.1 Introduction ................................................................................... 112

6.2 Initializing Simulation Parameters ................................................ 113

6.3 The Effect of Destination’s Buffering System on the Overall Network Performance.............................................................................. 115

6.4 Adaptive Security-aware Scheduler Vs Static IPsec Protocol ..... 123

6.5 Network Performance from Edge Router Agents’ Perspectives... 128

6.6 Adaptive Security-aware Scheduler Vs Feedback IPsec............... 131

7. CONCLUSIONS AND FUTURE WORK............................................... 136

7.1 Conclusions ................................................................................... 136

7.2 Research Contributions .................................................................... 138

7.3 Future Work .................................................................................. 139

BIBLIOGRAPHY..................................................................................... 142
LIST OF TABLES

2.1: Network Security Threats and Security Services ............................................ 40
3.1: Confidentiality Algorithms .............................................................................. 63
3.2: Integrity Algorithms ........................................................................................ 65
3.3: Authentication Algorithms .............................................................................. 65
5.1: IEEE 802.1Q PCP Fields ................................................................................ 106
5.2: Overloading IEEE 802.1Q PCP Fields ........................................................... 107
5.3: Overloaded PCP Bits for Confidentiality ......................................................... 110
5.4: Overloaded VID Bits for Integrity ................................................................. 110
5.5: Overloaded VID Bits for Authentication ......................................................... 111
6.1: Average Pending Packets at Destination for Confidentiality ....................... 127
6.2: Average Pending Packets at Destination for Integrity .................................... 127
6.3: Average Pending Packets at Destination for Authentication ......................... 128
6.4: Average Pending Packets at Destination (Weighted Multi-layer) ................. 128
LIST OF FIGURES

1.1: Network Development ..................................................................................... 3

2.1: Real-time Generator Models ........................................................................... 15

2.2: IEEE 802.1Q Priority Code Point (PCP) ......................................................... 21

2.3: Phases of Hacking Process ............................................................................. 38

2.4: Security Services Algorithms ......................................................................... 41

3.1: Object-oriented Agent-based Design Phases ............................................... 53

3.2: Environment Topology .................................................................................. 54

3.3: Multi-agent System Model ............................................................................. 55

3.4: IEEE 802.3 Ethernet Frame Format ............................................................... 69

3.5: Video Packet Size Distribution ...................................................................... 70

4.1: Waterfall Software Engineering Model ......................................................... 74

4.2: System Timing Diagram ................................................................................ 89

4.3: Initializing System Parameters ...................................................................... 91

4.4: Generating Simulation Results ..................................................................... 92

4.5: Creation of Results Data Base ...................................................................... 93

4.6: Packet Class .................................................................................................. 94

4.7: Creating Independent Exponential Segments ............................................... 94

4.8: Creating Independent Uniform Segments ..................................................... 95
List of Figures–Continued

4.9: Generating Source Packets ................................................................. 95
4.10: Generating the Packet Effective Deadline ........................................... 96
4.11: Packet Queuing Process .................................................................... 96
4.12: Serving Real-time Packets ................................................................. 98
5.1: IEEE 802.1Q Tag Insertion ................................................................. 105
5.2: IEEE 802.1Q Registered Fields ........................................................... 107
5.3: 802.1Q Overloaded Bits for Weighted Multi-layer Model ..................... 109
6.1: Destination Buffer Effect on Confidentiality Security Level Enhancement ........................................................................................................ 116
6.2: Destination Buffer Effect on Integrity Security Level Enhancement ....... 116
6.3: Destination Buffer Effect on Authentication Security Level Enhancement ........................................................................................................ 117
6.4: Arrival-rate of Packets at the Destination ............................................ 118
6.5: Buffer Effect on Security Level Enhancement (Weighted Multi-layer $\psi_1$) ... 119
6.6: Buffer Effect on Security Level Enhancement (Weighted Multi-layer $\psi_2$) ... 119
6.7: Buffer Effect on Average Packets Delays (Single-confidentiality) .......... 120
6.8: Buffer Effect on Average Packets Delays (Single-integrity) ................. 121
6.9: Buffer Effect on Average Packets Delays (Single-authentication) ......... 121
6.10: Buffer Effect on Average Packets Delays (Weighted Multi-layer $\psi_1$) .... 122
6.11: Buffer Effect on Average Packets Delays (Weighted Multi-layer $ψ_2$) 122
6.12: Average Buffer Consumption at Destination for Confidentiality 124
6.13: Average Buffer Consumption at Destination for Integrity 124
6.14: Average Buffer Consumption at Destination for Authentication 125
6.15: Average Buffer Consumption (Weighted Multi-layer $ψ_1$) 125
6.16: Average Buffer Consumption (Weighted Multi-layer $ψ_2$) 126
6.17: Miss Ratio Metric at Server Agent 130
6.18: Average Total Packet Delays Metric at Queue Agent 130
6.20: Feedback IPsec Security Level Changes for Integrity 133
6.21: Feedback IPsec Security Level Changes for Authentication 133
6.22: Feedback IPsec Security Level Changes (Weighted Multi-layer $ψ_1$) 134
6.23: Feedback IPsec Security Level Changes (Weighted Multi-layer $ψ_2$) 134
CHAPTER 1
INTRODUCTION

1.1 Background and General Overview

The earliest type of packet network applications was governmental based. Such applications were not sensitive to any variations in packet losses or delays. Accordingly, the service adopted by the service provider (internet) was the best effort service, where all data packet streams were treated equally by the service provider without denying any data traffic admission. The only limitation that affects such provided service is the availability of the network’s resources, where additional system delays will be added.

Different factors played a key role in the transition of the provided service by the internet. One factor was based on the fact that different classes of network applications with different requested services started to share and congest the internet. Real-time video and audio streams are examples of such classes. According to the previous data classes, real-time data losses and system delays became more critical [1]. Another factor was based on quality issues, where the internet became a commercial entity that needs to provide its customers with the best quality of service (QoS) guarantees [2][3].

According to the type of requested service by the network application, QoS could be in different forms such as delivery, capacity, reliability, mean time between failures (MTBF), mean time to restore a service (MTRS), or any combination of such metrics [4]. To provide real-time network applications with such QoS guarantees,
different network technologies were developed such as differentiated services (Diff-Serv) and multiprotocol label switching (MPLS) [5]. Such technologies provide the required QoS guarantees by applying the appropriate real-time scheduling algorithm.

Real-time scheduling algorithm selects the next appropriate real-time packet to be served among a number of arrived packets from different data classes to the scheduler [6]. The process of choosing the appropriate scheduling algorithm is mainly controlled by the type of flowing data streams. According to the best effort traffic (text), the first come first served (FCFS) scheduling algorithm shows high efficiency in providing the best services to its applications, while for real-time traffics (video, audio), the priority scheduling algorithms such as earliest deadline first (EDF) and differentiated earliest deadline first (Diff-EDF) are more efficient in guaranteeing the required QoS requirements to such data flows [7][8].

**1.2 Problem Statement**

Nowadays, real-time data packet sources are in care of providing security services to their real-time applications [9][10], making them robust against different security threats specially in local-area network (LAN), where most of the hacking processes occurred at the network’s edge [11]. The development of network technologies is shown in Fig. 1.1.

In order to provide such security services on the network’s data streams, different security protocols were implemented such as the secure sockets layer protocol (SSL), the transport layer security protocol (TLS), and the internet protocol security (IPsec). With the current security protocols, any dynamic change in the
network can not affect the pre-negotiated security level. Therefore, network performance issues are not taken into account and the QoS may not be guaranteed for different classes of real-time data streams. Such results may lead into a catastrophe especially for those hard real-time network applications [12].

**Figure 1.1: Network Development.**

While providing such security services to its real-time network applications, service provider should keep a balance between guaranteeing such security services and preserving the overall performance of the network. The overall performance of a real-time network could be measured by different network performance metrics (NPMs) such as miss rate, total average packets delay, functionality, jitter, and throughput [13]. A key factor that affects such NPMs, and thus controls the overall performance of the network is the best utilization of the network’s queuing system, which regulates the total amount of traffic load in the network and thus, limits the maximum throughput in the network [14]. Accordingly, different network-based
algorithms were implemented based on network buffer estimation models such as routing, scheduling, maintenance, load balancing, and security [15].

Different methodologies were implemented to analyze and measure the overall performance of the network such as off-line monitoring, agent-oriented systems, and live monitoring. Such monitoring techniques were based on queuing theory analysis models [16]. Conventional simulation techniques were inefficient in modeling and analyzing complicated heterogeneous environments such as dynamic real-time networks with QoS guarantees and security aspects. In order to overcome such limitations, real-time agent based simulation systems were implemented, where the whole environment is modeled by interactive entities that are cooperating together within a time-critical constrained protocol to accomplish the main system’s tasks [17].

In designing any multi-agent system, two main pre-phases should be defined: collaboration and interaction. Collaboration is the process of establishing different levels of cooperation between agents, while interaction is the protocol of rules and constraints that control the different transactions performed by agents. According to the environment at which the multi-agent system was deployed, two main agent-based architectures were implemented: software and artificial intelligence.

In this research, we propose an adaptive security-aware scheduling with congestion control mechanism for packet switching networks using real-time agent-based systems. The novelty of our research could be presented from the following perspectives:
1) Our work implements an object-oriented agent-based system that combines the functionality of real-time scheduling with the security service enhancement for packet switched networks, where the real-time scheduling unit uses the differentiated-earliest-deadline-first (Diff-EDF) scheduler, while the security service enhancement scheme adopts a congestion control mechanism based on resource estimation methodology.

2) The security service enhancement unit was designed based on two models: single-layer and weighted multi-layer design models. For single-layer, the design provides an enhancement for a single security service: confidentiality, integrity, or authentication, while the weighted multi-layer design provides an enhancement for multiple security services with different weights on a network with multi-processor end nodes.

3) The proposed system provides the required QoS guarantees for different classes of real-time data flows (video, audio), while adaptively enhances the packet’s security service levels according to a feedback from the control congestion model, which efficiently utilizes the buffering system at the edge network, and thus protects the network from being congested by heavy traffic load.

4) Our agent-based system eliminates the overhead of the security association phase performed by the IPsec protocol. Such elimination had been achieved by overloading the priority code point (PCP) fields of the IEEE 802.1Q tagged frame format for the single-layer scheme, while repeated single-layer and overloading both
the PCP and the VID fields of the IEEE 802.1Q tagged frame format fields were the adopted methodologies by the weighted multi-layer security design model.

By using the Diff-EDF scheduler, the proposed system minimizes the flows miss rates and the flows average total delays compared to the earliest-deadline-first (EDF) and the first-come-first-served (FCFS) schedulers. From the other hand, our adaptive security enhancement scheme minimizes the buffer consumption, the average total packet delays, and the pending packets at the end users compared to the IPsec protocol. It was also compared to an implemented feedback-IPsec, where our adaptive system eliminated the repeated security associations performed by the feedback-IPsec, hence less overhead and increases the chances to meet the flows QoS requirements.

1.3 Research Goals

The main goal of our research is to apply the object-oriented agent-based methodology to propose a new real-time security awareness scheduler, which provides the real-time data packet flows with guaranteed QoS requirements, while adaptively enhances the flows’ security service levels in a packet switched networks. While providing such guarantees (QoS & security), the proposed system preserves the overall performance of the network, such that no network congestion occurs. To achieve this general goal, the following specific objectives were highlighted:

1- Evaluating the efficiency of using the Diff-EDF real-time scheduling algorithm at the scheduler agent over the well known FCFS and EDF scheduling algorithms. Network performance metrics (NPMs) will be in
terms of the flow’s miss rate at the server agent and the flow’s average total delays at the edge router’s queue agent.

2- Evaluating the efficiency of our proposed algorithm at higher levels of data packets arrival rates at the edge router, where NPMs were measured for a dynamic network topology, such that the number of secure data channels (source/destination pairs) was varied by a constant step in the single simulated iteration.

3- Evaluating the efficiency of our proposed algorithm over the static IPsec protocol for both single and weighted multi-layer security design models. The measured NPMs will be in terms of packets’ total average delays at the destination’s queue agent and the utilization of the destination’s buffering system.

4- Defining the functionalities and communication schemes for each network entity in the heterogeneous environment, such that the overall networking system could be designed and modeled using a real-time multi-agent system, which has the capability to enforce the required timing constraints on both requests and actions performed by the interacted agents.

5- Evaluating the efficiency of redesigning the IEEE 802.1Q Ethernet tagged frame for both single and weighted multi-layer security design models. In order to perform that, our proposed system was compared to an implemented feedback-IPsec protocol. Simulation results show the efficiency of our proposed system in eliminating the repeated security
association phase performed by the feedback-IPsec protocol for each security level change, and thus increases the chances of guaranteeing the requested QoS requirements for different classes of data streams.

6- Implementing a congestion control mechanism based on a resource estimation methodology for the local-area end stations. Such mechanism will be used in adjusting the real-time packets’ security levels, such that no congestion occurs in the network with both QoS and security requirements are achieved.

1.4 Dissertation Outline

Our dissertation is structured in seven chapters including the introduction one. In this section we provide the overall outline of our adaptive security-aware scheduling algorithm using agent-based systems in a real-time packet switched network.

Chapter two presents an exhaustive literature review that covers the recent related work to our research topics. According to the literature, we begin by reviewing the properties, types, and applications of real-time systems. For such real-time systems, we provide the methodologies that guarantee providing their flows within the requested QoS. Being a key factor of enhancing the overall performance of the network, the literature provides different methods for estimating the availability of the network’s buffering system. As a method of providing the required QoS guarantees to the different classes of data flows in the network, real-time scheduling has been reviewed. In this chapter we also provides an extensive literature about
different security topics such as security threats, security services, security protocols, and secure scheduling methods. Being the most efficient method for modeling and analyzing heterogeneous environments, our literature outlines the real-time agent-based systems.

In chapter three, we present the process of using the object-oriented real-time agent-based methodology to design and model our proposed system; the chapter provides the design process for each real-time agent. It also provides the design process of two real-time security models: single-layer and weighted multi-layer security models. Chapter four presents the implementation process of our real-time security-aware scheduler for packet switched networks. Such process includes implementing the workload process using the Brownian motion queuing model; it also provides the timing protocol implementation for both scheduling and security design models (single-layer & weighted multi-layer). The implementation of the live feedback mechanism was also reviewed. Finally, this chapter provides implementation of our proposed agent system using the .Net object oriented programming platform.

Chapter five introduces the common static network security protocols. According to the limitations of using such security protocols, this chapter provides a design model for an adaptive feedback-IPsec protocol, where the security levels for the data packets are adaptively upgraded. The proposed mechanism of overloading the IEEE 802.1Q frame format was reviewed as a method of solving the limitations of using the feedback-IPsec protocol.
The system’s simulation and numerical results was provided in chapter six. The chapter begins by initializing the simulation parameters. Simulation results examine the performance of our proposed agent-based system from different perspectives. For each simulation experiment, two parameters were identified: network performance metrics to be measured (NPMs) and the real-time agent at which the experiment was carried out. Finally, chapter seven introduces our research’s conclusions and contributions. We also provide blueprint directions for our future work.
CHAPTER 2
LITERATURE REVIEW

2.1 General Overview

In the area of networks and data communication, a huge amount of research has been performed to provide network flows from different classes with different levels of QoS guarantees. According to the type of data stream flowing through the network, QoS could be in different metric forms. Different categories of security threats attack different types of flowing data streams in the network; accordingly, data traffic generators are in care of applying security services to their data streams. Nowadays, researchers are studying the effect of applying such guarantees on the overall performance of the network. They are also trying to implement network technologies that provide both QoS and security guarantees to their data traffics, while still preserving the overall performance of the network.

In this chapter, we provide an intensive literature review about our system’s related topics. Since our research environment is a packet switched network, we outline the properties, types, and applications of real-time systems. The literature covers different methodologies that had been used to provide real-time applications with the required QoS guarantees. The proposed system makes a balance between providing the required guarantees to the network’s applications and the overall performance of the network. In order to achieve that, the network’s buffering system is efficiently utilized. As a key factor of enhancing the overall performance of the
network, the literature provides different methods for estimating the availability of the network’s buffering system.

As a method of providing the requested QoS guarantees to different classes of data flows in a real-time network, real-time scheduling has been reviewed in this literature. In this chapter we provide an extensive literature about different security protocols that had been adopted to provide the required security services to real-time network applications. Secure scheduling mechanisms at different environments will also be reviewed. Real-time agent-based system was the best method for modeling and analyzing our heterogeneous environment. It controls the limitations of using conventional simulation based systems; accordingly, the literature provides an overview of using such methodology in real-time heterogeneous networks.

2.2 Real-time Applications

In a real-time environment, the entire system should have the capability to enforce the required timing constraints on its sub-tasks [18]. Such constraints could be reflected by the associated relative deadline timing parameter. The real-time system should have a mechanism to check the validity of its functionality. The validation process could be achieved by applying two main correctness parameters: logical and temporal. The logical correctness checks for generating correct system outputs, while temporal correctness deals with the system clock. It checks whether system outputs had been generated at the pre-defined instances of time or not [19].

According to the type of real-time data traffic and its requested QoS requirements, the real-time system implements the appropriate scheduling algorithm.
that serves such traffic and guarantees its requested QoS requirements. In a passive real-time system, where the required time to serve each real-time traffic is pre-defined, the entire system uses the traffics’ specifications to decided if a schedule is exist for such data streams or not [20], and thus it performs a prior validity check for the previous correctness parameters (logical & temporal). From the other hand, active real-time system doesn’t have any prior expectation about system’s behavior [21], and thus an individual correctness mechanism should be performed.

2.2.1 Properties of Real-time Applications

Nowadays, different classes of real-time applications share and congest the same integrated real-time network; accordingly, integrated networks should have the capability to provide different types of services for its real-time data flows. Real-time applications could be in different forms such as audio streams, video streams, multimedia applications, real-time signal processing applications, and real-time control applications. The implementation of the serving real-time scheduler depends on the traffic’s generator model, where a specific model is defined for each real-time data generator. Such model specifies the characteristics of the generated real-time data traffics such as the size of the flow’s data units, the traffic’s inter-arrival time, the traffic’s associated deadline, and the sending rate.

Accordingly, three main real-time generator models were defined [22]: the fixed data rate model (FDR), the variable data rate model (VDR), and the fixed data rate with variable size model (FDVS). In the FDR model, the generator generates equally-size real-time data units periodically such as real-time control and data
processing systems for hard real-time medical applications [23]. VDR model generates equally-size data units asynchronously, where different gaps isolate the stream of data at different instances of time such as the discrete real-time audio systems. In such systems, the data traffic interrupts the scheduler aperiodically [24]. According to the FDVS model, different classes of data traffics with different characteristics are integrated in a hybrid model, where the generator generates variable-size data units at a fixed data rate; accordingly, a synchronization mechanism was implemented to regulate the process of serving such real-time data units [25]. The classification of real-time system generator models is shown in Fig. 2.1.

Due to its strict timing constraints, real-time system should have the following properties:

1- Efficient response time for external discrete interrupt events.
2- Flexible to adopt dynamic changes in the system’s environment.
3- Reliability, which could be achieved through deploying the logical and temporal correctness mechanisms.
4- High processing speed that insures providing the required QoS guarantees to different classes of data traffics.
5- Overload stability, where critical tasks will be given higher priority over other tasks to guarantee their QoS requirements.
6- The ability to be modeled and analyzed using simulation based systems or multi-agent systems for monitoring issues.
2.2.2 Classification of Real-time Applications

According to the level of stringency in the timing constraints for real-time tasks, real-time application could be classified into hard or soft real-time application [26][27]. According to hard real-time applications, each individual task should complete its functionalities within its specified deadline, where missing the task’s deadline leads into a catastrophe. In such systems, the data generator provides the network’s provider with the flow’s information. Such information will be used in the miss rate prediction process, which evaluates the capability of serving the generator’s flows within the requested QoS requirements; hence, synchronous FDR generator model is efficient in such applications, where both arrival rate and task’s size are precisely known in prior to the system.
According to the statistical analysis for the traffics’ characteristics, different hard real-time generator models were implemented such as traffic shaper model [28], peak theory model [29], and LBAP model [30]. Such models reduce the overhead and complexity of predicting the system’s behavior and exchanging flow’s information, and thus enhancing the overall performance of the system.

Hard real-time schedulers were applied in different applications such as fast computing machines, where a real-time scheduling algorithm was implemented to provide the optimal number of parallel processors needed for a multi-threading hard real-time environment [31][32]. Hard real-time systems were employed in diagnostic and therapeutic instruments such as magnetic resonance tomography (MRT) and remote (robotic) surgery equipments. Such applications provide patients with the required critical-time health care services [33][34]. A huge research was performed to implement hard real-time applications on network technologies, especially at the edge of the network, where different methodologies were applied on layer-2 devices of the OSI model (switch, hub) to protect the end nodes from being congested by heavy traffic loads. Such methodologies include the internet traffic management practices (ITMP) [35][36] and the hardware-modification based schedulers [37][38]. The previous methodologies require designing specific network resources for hard real-time applications. Such problem was solved by implementing a generic hard real-time protocol suit, which deals with the traffic, regardless the network’s equipments [39][40].
From the other hand, soft real-time applications are less sensitive to variations in QoS parameters; it can accommodate a pre-defined miss rate, while the whole system still considered reliable. Such applications are specifically designed for VDR generator models, where asynchronous traffics flowing in the system. Discrete real-time audio systems are examples of such soft real-time applications. In order to guarantee the QoS requirements for such systems, two models should be implemented: audio signal recognition model and soft real-time scheduling model.

According to the recognition model, the challenge was to detect the isolating gaps, where a discontinuity appears in the audio stream. Since the audio signal could be represented as a stationary signal, different statistical Markov models were proposed to recognize the whole audio stream (talk and pause) such as the unidirectional Markov model audio detectors [41][42][43], the bidirectional Markov model audio detectors [44][45][46], and the hidden Markov chain model audio detectors [47] [48]. The previous models show high efficiency in recognizing continues signals over the paused signals, and thus a new algorithm was implemented to efficiently detect the discontinuity gaps in the audio signal. Such algorithm called the tri-state audio detector [49][50][51].

The most efficient schedulers for soft real-time systems are the priority based schedulers such as the earliest deadline first (EDF) scheduler, where the task that is closer to expire will be given higher priority over other tasks [28]. Modified versions of EDF scheduler were implemented such as the dynamic queue deadline first (DQDF) scheduler, which integrates the functionality of the EDF scheduler with the
dynamic queuing model on a single processor environment. DQDF provides an efficient utilization for the system resources with a minimized processing overhead [52]. Another proposed scheduler was the adaptive weighted fair queue (AWFQ) scheduler. In such scheduler, the tasks will be placed in queues with different properties. According to the status of the network and the flow’s QoS requirements, the priorities of the queues are dynamically change, such that the overall performance of the network will be enhanced [53].

2.3 Real-time Networks

The earliest types of data streams were not sensitive to any timing constraints; hence, the earliest version of the internet was in care of providing the network applications with the best effort services, where the only guarantee is to serve the arrived tasks in the order they were received. Such provided services started to be inefficient, especially when different categories of data traffics with different requested time-critical services begin to share the same integrated network.

In order to handle such time-critical services, network technologies started to implement embedded real-time protocols that provide the real-time data traffics with guaranteed QoS requirements. The process of providing guaranteed QoS requirements is mainly controlled by the overall performance of the network; hence, real-time network technologies were implemented based on protecting the network from being congested by heavy traffic load, and thus the chances for guaranteed QoS requirements would increase.
2.3.1 Quality of Service for Real-time Networks (QoS)

According to data communication, QoS could be defined as the ability of real-time network technologies to guarantee an appropriate delivery for different real-time data streams within their requested timing constraints. In order to fulfill such task, network technologies depend on the process of prioritizing arrived data streams, such that the stream that is more critical to variations in delays and data misses will be given higher priority than other data streams [54].

QoS could be represented in different metric forms such as delivery, capacity, and reliability. According to the generator’s sensitivity factor, delivery could be in one of two different forms: miss rate or total average delays. Miss rate could be in terms of the number of tasks lose their relative deadlines to the total number of arrived tasks, while task’s delay could be defined as the task’s waiting time in the scheduler queue ready to be served [55]. Capacity could be defined in terms of the aggregate bandwidth provided to the different classes of data streams. According to the international telecommunication union (ITU), reliability could be defined in one of two forms: mean-time between failures (MTBF) or mean time to restore a service (MTRS) [56]. Such QoS parameters could be in any of the previous forms; they could also be defined as a combined function of the previous metrics [57][58].

According to the type of flowing data traffics, the load of real-time data traffics, and network’s performance considerations, different models were adopted to implement the QoS guarantees in the system [59]. The earliest models were based on the implementation of threshold functions such as the quality-threshold and the
quantity-threshold models. Quality-threshold function could be driven by a single or a set of QoS metrics, such that none of the real-time tasks can exceed the pre-defined QoS metric’s value [60][61]; otherwise, the task will be considered as an expired one. Quantity-threshold function could be defined by determining a specific number of tasks that can exceed the pre-defined QoS values without affecting the overall reliability of the system [62]. According to the previous threshold models, the system doesn’t differentiate between the data streams, it treats all the traffics equally; hence, such models could be more efficient when dealing with best effort traffics, or when the system doesn’t accommodate heavy traffic loads.

The revolution in networking and data communication fields pushed the researchers to think of different QoS models, especially when different categories of data flows started to share and congest the same integrated network; accordingly, differentiated QoS models were adopted [63]. Such models were implemented by modifying the IEEE protocols for both Ethernet frame format and IP packet format. According to the Ethernet frame format, priority code point (PCP) fields were added in a new modified IEEE 802.1Q tagged Ethernet frame format [64]; such fields prioritize different classes of data flows in an integrated network as shown in Fig. 2.2. From the other hand, differentiated service (DiffServ) fields were added in a new 802.1P packet format, such that different levels of QoS guarantees were implemented at layer-3 of the OSI model [65].

Differentiation models are not efficient when dealing with heterogeneous environments such as real-time networks with stringent QoS and security
requirements. For such environments, network performance issues should be taken into consideration, such that no network congestion occurs. Differentiation models could be implemented as priority markers for real-time data streams. The new QoS models are based on system negotiations between data generators and the service provider, where each data source sends its traffic’s information to the provider; accordingly, the provider predicts its capability of serving such traffics within the requested QoS requirements, and notifies each source. Such models were implemented by applying the appropriate real-time interactive scheduling algorithm, which is mainly based on applying a resource estimation methodology that ensures the best utilization of the network’s buffering system [66].

Figure 2.2: IEEE 802.1Q Priority Code Point (PCP).

2.3.2 Real-time Network Performance Metrics (NPMs)

The overall performance of the network plays a key role in specifying the type of QoS guarantees that could be provided by the network technology. In real-time
packet switched networks, data packet generators perform a negotiation process with the system provider on the requested QoS requirements. Such negotiations are controlled by the service level agreement protocol (SLA) [67], which adaptively controls the process of guaranteeing the QoS requests for the real-time flows according to the network’s performance level.

The overall performance of the real-time network could be measured by different network performance metrics (NPMs) such as miss rate, average total packet delays, functionality, jitter, and throughput [68]. As we can see, the first two NPMs (miss rate and total average packets delay) are common between the QoS requirements and the overall performance of the network. According to the QoS requirements, the previous metrics are predefined by the data packet generator as requested services, while they will be measured using network monitoring techniques for the network performance case.

The functionality metric measures the efficiency of network elements at different layers of the OSI model in performing their tasks. Jitter can be defined as the variation in the arrival times for the real-time tasks at the destination side. Such variations are caused by different delay factors such as network congestion delays, delays by buffer limitations at both edge router and end stations, and delays by priority-based scheduling algorithms [69]. Throughput could be defined as the average rate of transmitted data over the communication data link; it is measured by (bits/second). This metric reflects the overall utilization of the data link, which affects the overall performance of the network [70].
In order to measure the previous metrics, different queuing-theory based monitoring techniques were implemented such as conventional simulation based systems (offline monitoring), agent-oriented systems, and live monitoring methods. Conventional simulation based systems are mainly based on capturing a set of packets from different flowing data streams. Once the data was collected, analysis processing mechanisms will be performed on the collected data to evaluate two main NPMs: utilization and throughput. Different monitoring projects were built based on such methodology such as global coral reef monitoring network (GCRMN) [71] and web-based internet/intranet network traffic monitoring and analysis systems (WebTrafMon) [72].

Live monitoring techniques are mainly efficient for time-critical applications, where a direct response should be taken according to the status of the network. The implementation of such methodology is carried out through the usage of specific control packets; according to live monitoring, the provider sends specific control packets (ICMP echo request packet) to the pre-defined network’s component, and then analyzes the arrived response packets (ICMP echo response packet). The performance metrics to be evaluated using such techniques include: the connectivity, the miss rate, and the total average packet delay [73]. One of the projects that implement such technique is the ping end-to-end reporting, which was used to implement end to end performance tests through the system’s data links [74]. Another active monitoring project is the national internet measurement infrastructure (NIMI), which was used to check the reliability of the internet clouds and paths [75].
Agent oriented monitoring systems are checking the same NPMs of the offline monitoring models. Such models solve the limitations of using offline models in large heterogeneous environments such as wide-area networks (WANs), where both QoS and security requirements should be provided. In this model, network’s components are modeled by real-time agents that are cooperating together to accomplish the main system’s tasks. The agent system inherits object oriented capabilities to handle the complexity of such heterogeneous environments [76]. One of the most famous work on such monitoring models is the measurement and analysis on the wide area internet (MAWI) [77], which was implemented to trace public traffic streams through the internet (WAN).

2.4 Buffer Estimation Techniques for Real-time Networks

Nowadays, researchers pay a significant attention on the factors that affects the overall performance of networks, especially those real-time networks that need to provide different QoS guarantees to their real-time applications. The best utilization for the network’s buffering system is a key factor that regulates the network’s traffic, controls the maximum throughput in the network, protects the network from being congested, and improves the overall performance of the network. As a result, such efficient utilization will increase the chances of guaranteeing the flows’ QoS requirements; accordingly, efficient communication-based algorithms such as real-time routing, real-time scheduling, network maintenance, load balancing, and network security were implemented based on such efficient buffer utilization [78].
In order to achieve an efficient utilization for the network’s buffering system, the service provider implements an appropriate buffer estimation methodology that works at different network’s layers. According to real-time network’s type and the requested needs from the real-time data streams, different buffer estimation models were implemented [79].

One of the models was based on the statistical analysis, where two Poisson distribution functions were implemented: one for modeling customers’ buffer requests, while the other one for modeling the average holding time for the information in the system’s buffer. The model calculates the optimal number of buffers needed to accommodate the arrived real-time tasks without congesting the network, and thus maintaining high network performance [80]. Another model was implemented to evaluate the minimum limit of buffers needed to accommodate two main types of data: internally generated data and received data from other nodes. The research strategy was implemented at each network node in a homogenous network, such that a maximum throughput is maintained, while the network will be protected from being congested by heavy traffic load [81].

In a TCP-IP packet switched network, a buffer estimation model was developed to provide an efficient utilization for the buffering system in a real-time multimedia network; such model uses the flow’s information such as transmission rate, average round trip time (RTT) for the stream’s packets, and the data packet size to evaluate the minimum number of available buffers at both end users and edge router, such that QoS requirements will be guaranteed with minimum average
packets’ delays [82]. In the area of packet optical networks, a buffer estimation model was implemented to evaluate the minimum buffer requirements at each optical node in the network. The model combines a proposed genetic algorithm with the shortest path first (SPF) routing algorithm, such that the best route for the real-time optical data packet is determined without overriding the internal data buffer of the optical node [83].

Queuing theory was employed to evaluate the average queue size for an internet gateway server in a real-time network; the model was based on an asynchronous live monitoring for the server’s queue. By analyzing the collected information from live monitoring, the service level agreement protocol (SLA) specifies the optimal QoS guarantees that can be offered by the service provider, such that a minimum miss rate values are achieved [84]. Later, a work was performed to examine the capability of using tiny routers with bounded buffers in a real-time network. The proposed algorithm implements a buffer estimation mechanism for the router’s internal queues, such that a peak throughput is maintained with a minimum number of dropped packets [85].

2.5 Real-time Network Scheduling

In order to provide different real-time data streams with guaranteed QoS requirements, the real-time service provider implements the appropriate real-time scheduling algorithm to be applied on the arrived data flows. Such scheduling algorithm decides the order of serving a number of arrived data tasks from different flows to the scheduler’s queue [86]. Besides guaranteeing QoS requirements for real-
time data flows in an integrated network, schedulers are the key behind the implementation of real-time operating systems, real-time multi-processing systems, real-time data-base transaction management, and pipeline management systems [87].

### 2.5.1 Types of Network Scheduling Algorithms

Different factors control the process of choosing the appropriate scheduling algorithm that should be applied to provide the best services to arrived data streams. Such factors include the type of flowing data traffic, the processing capabilities of the service provider, the type of requested QoS requirements, and network’s available resources. In order to model and design any real-time scheduler, different characteristics should be taken into consideration such as priority, preemption, rate controllability, and bandwidth conservation [88]. According to priority characteristics, schedulers are classified into dynamic or fixed priority sc. In a dynamic based scheduler, the priorities of queued data tasks at the scheduler are dynamically changing according to the specifications of new arrived data tasks, such that QoS parameters among all streams will be satisfied. From the other hand, in a fixed priority scheduler, data streams will be given their static priorities in the SLA phase, without any future updates [89].

Preemptive schedulers are based on dynamic priority schedulers, where a new higher-priority arrived data task will cause the system to preempt the under-served lower priority task and start serving such arrived task. Such schedulers are mainly used for hard real-time systems with strength timing requirements [90]. Rate controllable scheduler depends on the service level management phase (SLM), which
specifies a fixed sending rate for each real-time stream in the negotiation process, such that the QoS guarantees will be met for all data flows, while non-rate controllable scheduler adaptively enhances the sending rates for the data streams according to the network status, such that none of the traffics misses its QoS [91].

Bandwidth conservative scheduler keeps all the links with the end nodes in an active mode, even when no packets are currently scheduled for a certain destination, while non-bandwidth conservative systems only activates the link that is currently connected with the under-served node, and terminates all other links; such system reduces the number of reserved resources in the network, whereas additional overheads are added due to the links’ reestablishing processes [92]. According to the previous scheduling characteristics, different scheduling algorithms were implemented such as first come first served (FCFS), earliest deadline first (EDF), weighted fair queue (WFQ), and multi-level schedulers.

The FCFS scheduler was implemented to serve the earliest virgins of real-time data traffics (best effort traffics). Since best effort traffics are not sensitive to any variations in packet losses or delays (QoS requirements), they were treated equally by the service provider, and thus the FCFS was the best choice for serving such types of data streams. According to FCFS, arrived data packets will be queued in the scheduler’s buffer according to their arrival-time, such that the packet that arrived first will be at the top of the queue ready to be served next; hence, the FCFS scheduler is considered as the simplest scheduler. The only requirement for its
implementation is to order the packets in the scheduler’s queue based on their arrival time [93].

In order to examine the efficiency of such scheduler, FCFS was deployed in different environments such as asynchronous best-effort networks, integrated real-time networks, and real-time networks with multi-processing technology [94]. For asynchronous best-effort networks, the PB-FCFS scheduler was proposed by integrating the FCFS algorithm with a backfilling strategy. By implementing a resource recycling process, such scheduler improves the utilization of the network’s buffering system, and thus enhances the overall network’s performance [95].

In a real-time integrated network, different classes of data flows with different QoS requirements share and congest the same network. In such environment, the FCFS scheduler shows low efficiency in guaranteeing such QoS requests, especially for both heavy load and hard real-time streams [96]. A reasonable reliability was achieved when deploying the FCFS scheduler with real-time multi-processing networks, where the only limitation was the type of flowing real-time data traffics. Simulation results show that it would be more reliable if the system is dealing with soft real-time applications rather than hard real-time applications [97].

Weighted fair queue (WFQ) scheduler was implemented to solve the limitations of using the FCFS scheduler in an integrated real-time network. According to WFQ, each data flow reserves a virtual sub-queue at the scheduler side, such that the scheduler assigns each sub-queue with an associated weight. The queues will be served in a round-robin mechanism according to their weights; hence, the starvation
problem of the FCFS will be solved, which totally leads to increase the chances of guarantee the flow’s QoS requirements. According to its proportional fairness, this model provides a minimized end-to-end transmission delay limits and efficiently utilizes systems with limited bandwidth [98].

According to the network’s technology, different WFQ models were implemented. Generalized processor sharing (GPS) model was adopted for clustered network, where flowing data units are in terms of tasks (jobs). For such model, different streams from different sources are flowing to the scheduler simultaneously, where each flow consists of a batch of divisible tasks that reserve a certain sub-queue in the scheduler. According to delay bounds, the scheduler gives dynamic weights to sub-queues that can be change based on the queue’s capacity. Simulation results show the efficiency of using such scheduler in terms of bandwidth utilization and end-to-end delays [99, 100].

GPS scheduler is non-efficient when dealing with packet switched networks, since packets are indivisible data unit, and thus the packet weighted fair queue (PWFQ) scheduler was implemented, which deals with packets rather than tasks [101][102]. When the time slice for the underserved queue finishes, PWFQ doesn’t terminate the packet’s servicing process, and thus the allowable sending rate (bandwidth) for such session will be exceeded. In order to solve such limitation, the worst-case fair weighted queue (WF^2Q) scheduler was implemented [103][104]. To model and design such scheduler, two main phases were developed: the eligibility phase and the scheduling phase. According to the eligibility phase, the packet is
checked if it can be scheduled within the current time slice or not, while in the scheduling phase, eligible packets are scheduled according to the PWFQ scheduling algorithm.

A major problem with the previous packet scheduling models is the big variations in the bandwidth utilization for the online session, especially when a pre-offline session becomes online. For such scenario, the sending rate of the online session is sharply dropping, while the pre-offline one retains its maximum bandwidth in one shot. To regulate such sudden changes in the session’s bandwidth, the slow-start weighted fair queue ($S^2$WFQ) scheduler was implemented [105], which regulates the process of reserving the bandwidth of the new online session; it assigns the bandwidth to the session in an exponential upgrading, until it gains the whole specified bandwidth [106].

The EDF scheduler was implemented as the most efficient algorithm for hard real-time applications; it is a priority-based algorithm with the packet’s relative deadline being priority key, such that the packet that is closer to expire will be given higher priority than other arrived packets to the scheduler’s queue. Different models of EDF scheduler were implemented to guarantee the flows’ QoS requirements at different environments. Standard EDF scheduler (SEDF) was implemented to serve real-time data streams in an integrated network [107]; it shows an optimal efficiency when dealing with equally-likely data traffics, where SEDF provides the same miss rate QoS metric for all data traffics; hence, it was not efficient when dealing with data traffics with different QoS requirements [108]. In order to solve such limitation, a
modified version of SEDF with live monitoring strategy was developed, where all data flows are initially operated with the same levels for each QoS metric. Once the scheduler receives a notification from the monitoring policy about any stream being close to its QoS requirements, it adaptively modifies the priority scheme for the streams, such that all QoS will be met [109].

The previous EDF models were efficient when dealing with limited number of data generators, such that no heavy load traffic will affect the process of guaranteeing the requested QoS metrics. In order to solve such limitation, a modified model of the EDF scheduler with a pre-negotiation phase was adopted. In such model, both system provider and data generators negotiate on the type of services that can be guaranteed [110][111]. In order to achieve an efficient scheduling, the algorithm takes into consideration different system parameters such as traffic characteristics (relative deadlines, arrival times, service times), the available bandwidth, the number of data sources, and the scheduler’s available resources (computational power, available buffers). According to this scheduler, the negotiation phase determines if a schedule is available for each data stream within the requested QoS or not.

The pre-negotiated model is efficient when dealing with static network topologies; for dynamic topologies such as wireless networks, the scheduling algorithm should have the capability to handle any new updates in the network’s topology; accordingly, a modified version of pre-negotiated model was implemented, where a feedback about the status of the network is sent to the scheduler to reinitiate
the pre-negotiation phase; according to the received notification, the scheduler adjusts scheduling parameters to handle the topology changes [112][113].

Different multi-level scheduling models were implemented. A tri-state multi-layer scheduler was implemented to model the asynchronous data switch, where three different scheduling mechanisms were adopted to serve different classes of data traffics (video, audio, text): FCFS at the bottom layer, fair queue (FQ) at the intermediate layer, and EDF scheduler for the upper layer [114]. For open wireless technology, a multi-level scheduler was implemented based on the cyclic exclusive scheduling (SE) to provide the requested services for data frames at the data-link layer of the OSI model [115].

For real-time integrated networks, a multi-level scheduler was adopted to provide the requested QoS guarantees for both hard and soft real-time data traffics. Hard real-time traffics were forwarded to the upper level of the scheduler, where the EDF algorithm is working at that level, while a WFQ scheduling algorithm was implemented at the downstream layer to serve those soft real-time traffics [116]. In a multi-processing network system, a multi-level EDF scheduler was implanted to regulate the execution process of different data tasks among the multi-processing system. The outer level determines which processor will be chosen to serve the task, where the capacity of the processor’s internal queues would be the priority key, while the inner scheduler adopts the EDF algorithm to serve the local tasks. According to this design, the system provides high throughput, less miss rate, and best utilization for the multi-processing system [117].
2.5.2 Network Scheduling using Multi-agent Systems

Simulation based systems were used to analyze and model different types of network’s technologies. Unfortunately, conventional simulation techniques were inefficient when dealing with complex real-time heterogeneous environments such as real-time networks with QoS and security requirements. In order to control such limitations, multi-agent network simulation based systems were proposed [118][119][120].

Multi-agent system could be defined as a set of entities (hardware and software) that are communicating together through a well defined protocol to perform a specific task. In designing any multi-agent system, two main pre-phases should be defined: collaboration and interaction. Collaboration is the process of establishing different levels of cooperation between agents [121], while interaction is the protocol of rules and constraints that controls the transactions performed by agents [122]. According to the environment at which the multi-agent system is deployed, two main multi-agent architectures were implemented: software and artificial intelligence (AI) architectures. Different AI architecture models were implemented such as reactive, deliberative, and hybrid models, where the main common design parameter for such AI models is the decoupling, such that agents’ transactions are independent; hence, they could be expresses as multi-threading systems [123].

From the other hand, software models are object-oriented based models, and thus they are capable of inheriting the required object-oriented based terminologies and methodologies, which provide the agent system with huge capabilities for
modeling and designing different real-time applications [124]. The real-time agent system should have the capability to enforce the required timing constraints on both requests and actions performed by the interacted agents; accordingly, different real-time applications were implemented using software agent-based systems such as real-time scheduling [125].

A quantitative multi-agent real-time scheduler was implemented to model the real-time scheduling problem in both static and dynamic network environments. The proposed model analyzes a function that defines the quantitative relationships between real-time multi-agent tasks’ constraints and entities’ constraints, such that the real-time tasks will be served within their critically-timing constraints and agent’s performance constraints. The static networking environment was handled by the earliest release time first scheduling algorithm (ERFS), while the dynamic one was handled using the real-time dynamic scheduling algorithm (RTDS) [126].

In homogeneous networks, a real-time multi-agent system was proposed to schedule time-critical tasks at different homogenous distributed network’s topologies. According to the topology’s directed acyclic graph (DAC), three main agent’s centralized scheduling models were implemented: basic scheduling model, forward-backward scheduling model (FB), and partial forward-backward scheduling model (PFB). The three-model system shows high efficiency in serving such critically-timing tasks within the requested QoS metrics [127].

The heterogeneity of distributed networks adds different limitations on the process of developing efficient frameworks for self-correction networks such as
limitations of scheduling critically-timing tasks, limitations of implementing efficient solutions for heavy traffic loads, and limitations of solving large-scale network corruptions. In order to solve such limitations, an economical-based multi-agent scheduling architecture for self-correction networks was proposed. Simulation results show the efficiency of such approach in solving the previous network’s limitations without affecting the overall network’s performance, where measured performance metrics were in terms of buffer utilization and round-trip time delays [128].

An agent-based grid scheduling (ABGS) architecture was proposed to provide an efficient real-time scheduling for time-critical tasks on a grid network; three agents were defined for the ABGS system: source agent for task’s submission, local scheduler agent for allocating resources, and global scheduler for regulating traffics between user and local scheduler agents. The QoS requirements for the tasks were achieved by integrating the ABGS with the service level agreement protocol (SLA); such integration provides an efficient task allocation, natural load-balancing, dynamic reserving/releasing of system resources, and guaranteeing QoS for sensitive time-critical applications [129].

The main objective of a real-time digital advertisement application is to dynamically deliver an appropriate data-content to a specific destination within the pre-defined timing constraints. In order to serve such complex scheduling and dynamic environment, a distributed multi-agent system was proposed to handle the dynamic system’s behavior and the non-deterministic transactions between system’s entities. The system was modeled and designed using the coordinated and intelligent
rational (CIR) agent model, which provides a variety of inherited features from object oriented programming (OOP) methodology, and thus it has the capabilities of handling the dynamicity of such environments [130].

In a large scale real-time integrated network, a multi-agent scheduling system was proposed to provide guaranteed QoS requirements for different classes of data streams. According to the network management decomposition algorithms, three types of system’s sub-tasks were defined: priority-ranked tasks, equally-priority tasks, and unlikely priority tasks. Priority-ranked tasks were handled according to their inter-dependences. Equally-priority tasks were handled in parallel, while unlikely-priority tasks were handled based on their priorities. The multi-agent system shows a high efficiency in dealing with the dynamicity of such large scale environments, where the performance metrics were in terms of utilization, throughput, decomposition overhead, agent migration time, agent code burden, and the total execution of real-time sub-tasks [131].

2.6 Real-time Network Security

The rapid evolution in data communication and networking fields makes the use of computerized systems more efficient for most commercial applications. In such environments, end users share different local resources, and query for different external data applications. As a result, most data sources are in care of providing the required security services to their data applications, making them robust against different security threats, especially for those real-time data applications, where the cost of hacking such processes is extremely high; accordingly, different security
models were modeled to implement such security services at different data communication environments such as real-time packet switched networks, real-time wireless networks, real-time clusters environment, and handheld devices.

2.6.1 Real-time Network Security Services

According to real-time hacking process, hackers try to create a weakness in a specific network’s resource that leads to open a gate for different security threats, and thus harming the functionality of the whole system. Fig. 2.3 summarizes the phases of the hacking process. The process begins by identifying system’s resources and their functionalities; once the hacker identifies the system, he tries to access the system from a weak interactive gate, where no authorization is required, and then he tries to get higher levels of privileges to access secure data connections. Finally, the decision is made by the hacker whether to attain an authorized future access to the system or to stop at that level and denying the functionality provided by the attacked network’s resource [132].

![Figure 2.3: Phases of Hacking Process.](image-url)
According to the main purposes of the hacking process, the type and location of the hacked data, and the implemented hacking methodology, real-time network security threats could be in different forms such as snooping (sniffing), spoofing (aliasing), alteration (tampering with data), and denial of service [133]. Snooping could be defined as the unauthorized monitoring and interception of the data stream that flows through the network. While monitoring traffic information, the sniffer decrypts the entire packet’s data (payload) [134]. Spoofing (aliasing) is a fake representation performed by a hacking entity. In such process, the hacker starts to send its stream of packets with a faked source MAC address that represents a MAC address to a well-identified network node, and thus the original identity of the hacking source will be hidden [135].

Alteration threat is an unauthorized attempt to modify the payload of the data packet at the hop-nodes (routers), which are standing in the path between the data packet generator and the end user. Different methods were implemented to perform the alteration process such as changing the access privileges of the path-routers or changing the original identity of one path-router to the edge router’s identity [136]. Denial of service threat is the process of making the network’s server down, or a specific networking service to be unavailable; according to this threat, the hacking entity tries to consume the overall system resources through flooding the server with a continuous stream of hacking packets; such stream consumes the available buffering system at the server and congest the network with a heavy traffic load. As a result, the
In order to protect the network’s streams from the previous security threats, different security services were implemented [138]. Table 2.1 shows the security services needed to overcome each of the previous security threats. Each security service is implemented using different security algorithms, where the differentiation between such security algorithms is mainly based on two parameters: throughput and computational overhead, such that using high-level security algorithm decreases the system’s throughput and adds more computational overhead to the system.[139].

Table 2.1: Network Security Threats and Security Services.

<table>
<thead>
<tr>
<th>Network Security Threats</th>
<th>Network Security Services</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sniffing</td>
<td>Confidentiality using Cryptographic Algorithms</td>
</tr>
<tr>
<td>Spoofing</td>
<td>Secure Authentication</td>
</tr>
<tr>
<td>Alteration</td>
<td>Integrity using Hash-Functions</td>
</tr>
<tr>
<td>Denial of Service</td>
<td>Traffic Filtering &amp; Bandwidth throttling</td>
</tr>
</tbody>
</table>

Different factors control the process of choosing the appropriate security algorithm such as the availability of the network’s buffering system at both routers and end nodes, the computational speed capabilities of end nodes, the requested QoS requirements by data generators, the amount of load flowing in the overall performance of the network will be affected, and thus traffics’ QoS guarantees will not be met [137].
network, the security requirements defined by upper level applications, the deployed real-time security protocol, and the type of network technologies used to serve data streams’ requests [140]. Fig. 2.4 shows the associated security algorithms for different security services, where the security level increases in an up-down order [12].

![Diagram of Security Services Algorithms](image)

**Figure 2.4: Security Services Algorithms.**
Security algorithms were implemented for different data communication areas; in gigabit networks, a new proposed cryptographic protocol for confidentiality security service was implemented. The designed protocol integrates three different cryptographic enhancement models: confidentiality design model, parallelization, and algorithm independent hardware support, such that the overall performance of the proposed cryptographic protocol will be improved. Simulation results show the capability of such protocol to handle any dynamic change in the hardware structure of the gigabit network without affecting the process of guaranteeing the flows’ QoS requirements [141].

For wireless sensor networks (WSNs), a research study shows the efficiency of using the SEA confidentiality security algorithm over AES and RC6 algorithms in terms of resource requirements, processing time, and bandwidth utilization. Such performance characteristics make the SEA security algorithm efficient for different WSN topologies [142]. In a VoIP packet switched networks, a proposed confidentiality security methodology was implemented based on both AES and CBC cryptographic algorithms to provide an automatic synchronization for audio stream’s ciphers. The proposed methodology ensures both QoS and security requirements for the audio stream in terms of packet loses and confidentiality level; it also decreases the overall system overhead by neglecting the process of initializing the IP packet’s preamble field [143]. For real-time packet homogenous network, a study was performed to select the appropriate cryptographic algorithm from a suit of six algorithms (AES, RC2, RC6, Blowfish, DES, 3DES). The selection process was
based on the efficiency of the security algorithm when dealing with different network settings such as variations in packet size, processing speed, classes of data packets, and power consumption [144].

According to spoofing security threat, different authentication algorithms were implemented for different network technologies. A research was implemented to provide a secure authentication with a minimum key generation overhead in a heterogeneous wireless sensor network. Such proposed methodology was based on generating a modified version of HMAC-SHA-1 algorithm based on a new key generation process, which generates a small pool of random key chain using keyed-hash function generator [145]. In IP satellite networks, a multi-layer IP security protocol (ML-IPsec) was proposed to solve the limitations of adopting the standard IPsec protocol in the TCP performance enhancement proxy (TCP PEP) methodology. The proposed protocol modifies the authentication algorithms provided by the IPsec protocol, making them capable of providing the intermediate routers with a secure and controlled access to the flowing IP packets, while maintaining a high level of end-to-end security without affecting the overall performance of the network [146].

For life critical applications, a research was performed to apply secure authentication mechanisms on the sensitive data carried by wireless body area networks (WBANs). The research methodology was based on implementing a multi-mode authentication protocol using the AES security algorithm. According to the stream’s sensitivity level, three different modes were defined, such that one of the following authentication algorithms was applied for each mode: CBC-MAC-AES for
hard real-time stream (high-sensitivity), CCM-AES for soft real-time stream (intermediate-sensitivity), and CTR-AES for best effort stream (low-sensitivity). The security mode selection was performed at the application layer, such that the mode selection guarantees providing the optimal security level without affecting the overall performance of the network [147]. In hybrid wireless-Seattleite networks, a research was implemented to reduce the overhead generated by satellite nodes upon providing the required authentication levels for wireless packets. The process was achieved by proposing a broadcasting symmetric-key protocol based on TESLA certificates, where satellites nodes broadcast the source’s data keys through the hybrid network. The protocol uses the HMAC-MD5 and HMAC-SHA-1 algorithms for authenticating the data traffics [148].

Different hash-function algorithms were implemented to provide the integrity security service for different network environments. A research was performed to apply integrity security service on peer-to-peer networks (P2P). The procedure was performed by establishing secure data channels using the defined data commands provided by the trusted computing group (TCG), where the built-in SHA-1 integrity algorithm was invoked to provide security data transfer over the pre-established channels in a commercial P2P network [149]. In high speed wireless networks, an integrity security service protocol was proposed by implementing high speed architecture for modeling the MD5 and the SHA-1 hash-functions. The protocol was implemented using the embedded VHDL programming language, and was installed on the FPGA devices of the high speed WSNs. When compared to other integrity
hash-functions, the proposed protocol shows high efficiency in enhancing the overall performance of the network in terms of bandwidth utilization, total average delays, and operating frequency [150]. For real-time packet switched networks, a new protocol was proposed to apply integrity security services on real-time data packet streams according to the status of the network. The proposed protocol was built based on modifying the static IPsec protocol, where a selection controller was implemented to select between three integrity hash-functions: MD5, SHA1, and RIPEMD160; such protocol preserves the overall performance of the network through implementing a network’s status feedback mechanism using live performance monitoring technique [151].

2.6.2 Network Security-aware Scheduling

Conventional scheduling algorithms are in care of providing the required QoS requirements to different classes of network applications without paying any attention to enhance or optimize their security requirements. In order to achieve both QoS and security requirements, a level of cooperation between the scheduling unit and the security enhancement unit should take place. Accordingly, different models of secure scheduling architectures were implemented to provide the required QoS and security guarantees for different network’s technologies, especially on cluster networks, where most security threats occurred at LAN environments.

A security-aware heuristic architecture (SAREC) was implemented to apply the required confidentiality, integrity, and authentication security services on soft real-time tasks in clusters. The SAREC system was integrated to the EDF priority
scheduling algorithm forming the SAEDF security-aware scheduling algorithm. According to a well-defined security overhead model for each security service, SAEDF enhances the real-time tasks’ security levels, while still providing them with the required QoS guarantees; the proposed algorithm shows that the average security level for the system would be better enhanced than if we used conventional schedulers such as the first come first serve (FCFS) scheduler and the least laxity first (LLF) scheduler [12].

According to the type of the LAN’s computing platform (homogeneous or heterogeneous), a research was performed to implement a platform dependent security-aware scheduling algorithms. In this research two resource allocation security-aware scheduling algorithms were implemented: task allocation for parallel applications with deadline and security constraints (TAPADS) for homogeneous environment and secure heterogeneity-aware resource allocation for parallel jobs (SHARP) for heterogeneous environment. Such two algorithms were implemented to provide an optimal resource allocation that guarantees both enhancing different security levels for real-time tasks and providing the required QoS guarantees to the network applications [152]. A utilization based secure-scheduling approach was implemented to provide a high success rate for soft real-time applications on grid networks, while still providing such applications with the minimum required authentication levels. By using a load-balancing methodology, the proposed algorithm increases the system’s throughput level, such that the overall performance of the system and the real-time tasks’ security requirements will be preserved [153].
Two-phase security-aware scheduling strategy (TPSS) was implemented on real-time heterogeneous networks; such proposed algorithm provides both timing constraints and security services to its real-time applications. The first phase is operated by a load sensitive algorithm called DSRF. With heavy load traffic, DSRF decreases the security level of data flows, such that a high success rate is achieved within the guaranteed QoS requirements. For light traffic load, the algorithm increases the security level of the data traffics without affecting the flows’ QoS requirements. From the other hand, the second phase is controlled by the FMSL algorithm, which decreases the variance between the tasks’ security levels, such that the system could be expressed in an average security level that is closed to most flows’ security levels [154].

An adaptive security improvement strategy using dynamic window methodology (ASIDW) was integrated with the EDF priority scheduling algorithm in a heterogeneous network environment. The proposed algorithm provides the required QoS requirements to asynchronous real-time tasks, while adaptively enhances their security levels, such that the real-time tasks were initially accepted with the minimum security requirements. According to the available slack at each processing node, the dynamic window methodology reduces the computational overhead generated from enhancing the tasks’ security requirements; it shows better success rates and security enhancement than both SAEDF and TPSS security-aware scheduling algorithms [155].
A proposed space-time genetic Algorithm (STGA) was implemented to provide the minimum security requirements for real-time tasks in a heterogeneous network. In such environment, each network node is operated by a specific security algorithm, and thus the security level provided by each route is different. According to STGA, three security modes were defined: secure, risky, and f-risky. According to the secure mode, STGA forwards the data in a route, such that each node in the route provides a security level equals or higher than the flow’s minimum security requirements. For risky mode, the system chooses any available node to be in the route, such that a high throughput is achieved without caring to the security level enhancement process. From the other hand, f-risky mode accepts random available nodes in the route, such that the security probability risk doesn’t exceed the f threshold value [156].

A security-aware task allocation (SATA) algorithm was implemented to provide a trusted secure route within guaranteed QoS requirements for real-time streams in a heterogeneous network. SATA implements an initial discovery strategy to identify different specifications for each node in the system such as the node’s computational speed, the degree of security deficiency on each node, the size of the security enhancement buffer at each node, and the overhead of the node’s security model. According to the previous nodes’ parameters, SATA determines the route to be chosen, such that the following performance metrics will be enhanced: slowdown ratio, node utilization, security risk free probability, and end-to-end delays [157].
One of the most famous IP security protocols for packet switched network is the IPsec protocol. IPsec works at layer-3 of the OSI model and provides the IP packets with different security services such as authentication, integrity, and confidentiality. According to IPsec, the security level provided by the security association (SA) phase is fixed, and thus IPsec doesn’t pay any attention to the flows’ QoS requirements. Accordingly, QoS-capable-IPsec protocol was implemented as a modified version of the IPsec protocol; such proposed protocol handles the QoS requirements of the IP data flows and provides them with guaranteed security requirements. QoS-capable-IPsec integrates a priority scheduling algorithm with the IPsec protocol, such that the output of the scheduler is connected to the input of the IPsec protocol. The scheduler prioritizes the packets according to the required QoS requirements and adopts the optimal security level for each packet, and then it forwards the packet to a specific encryption unit in the IPsec structure, where the packet will be encrypted using the pre-determined security level by the scheduler [158].

Although the previous secure scheduling algorithms provides both QoS and security requirements for their flows, they have the following limitations: 1) The security enhancement process was performed by a central unit (edge-router) instead of the data generator, and thus more overhead on the system; 2) For any security level enhancement, security association phase should take place, which means more overhead and less chances to provide guaranteed QoS requirements; 3) For heavy traffic load, none of the schemes implements a congestion control mechanism, and
thus more chances to miss the required network performance metrics (NPMs); and 4) No feedback mechanism to real-time data generators based on the network’s status was implemented, and thus no traffic admission will be denied, which totally leads to congest the network.
CHAPTER 3
SYSTEM DESIGN MODEL USING REAL-TIME AGENT-BASED METHODOLOGY

3.1 Introduction

In order to model and analyze data communication environments, different queuing theory based methodologies were implemented such as off-line simulation techniques, online interactive systems, and multi-agent based systems. The earliest network technologies were depending on the capabilities of conventional simulation techniques to analyze and model the overall performance of the network. Such conventional techniques were suitable for those uncomplicated environments (best-effort networks), where no traffics’ QoS guarantees needed to be provided by the service provider.

Our research environment is a dynamic heterogeneous real-time network that provides both QoS guarantees and security requirements to its real-time data flows. The entire system also monitors the overall performance of the network and protects the network from being congested by heavy traffic load. According to such complicated specifications, conventional simulation techniques were inefficient to be used in analyzing and modeling such heterogeneous environment. In order to overcome such limitations, real-time multi-agent simulation systems were implemented, such that the entire system is modeled by interactive entities (hardware & software) that are cooperating together within a time-critical constrained protocol to accomplish the main tasks of the system.
The multi-agent simulation system used in our research is an object-oriented based model. Such model provides a mechanism to inherit the required object-oriented methodologies needed to model and design our complicated interaction schemes between the interactive agents. It also allows the agents to be synchronized with time-critical events, which makes the system applicable to simulate our real-time heterogonous environment.

In designing any agent-based system, three main design phases should be defined:

1- Problem decomposition: In this phase, the main environment will be decomposed into a group of interactive entities, where each entity performs specific sub-tasks of the whole system process.

2- Entity Modeling: According to this phase, each pre-decomposed entity will be modeled by an object-oriented agent. The modeling process defines the entity’s main functionalities, behaviors, data bases (input data), and the format of generated output data.

3- Communication protocol: This phase defines both the route of data transfer and the interaction processes between the communicated agents. The interactions between agents are defined as a set of asynchrony subroutines (triggers), which control the main functionalities of the cooperated agents. They are in terms of status notification, status modification, transfer of system’s parameters, and change agent’s
behavior. The object-oriented agent-based system design phases are shown in Fig. 3.1.

![Figure 3.1: Object-oriented Agent-based Design Phases.](image)

3.2 Object-oriented Agent-based Design Model

The topology of our heterogonous environment is shown in Fig. 3.2, where $N$ real-time data sources are communicated with $N$ destinations (end users) through secure data channels in a packet switched network. The destinations are connected to the default gateway (edge-router) in a star topology. The core entities of the multi-agent system were installed at the edge level of the network (edge-router) for many reasons:
1- Most of the network-hacking processes such as spoofing, alteration, sniffing, and denial of service occur at the local-area environment (LAN), which is directly connected to the edge router.

2- The location of the edge router makes it capable of identifying the MAC address of the end user (destination); such identification will be used by the agent-system in different stages such as scheduling, buffer estimation, and the security enhancement of the IP real-time data packet.

3- Since the edge router is the last hop in the IP route, it has the capability to identify the source’s IP address. Such identification will be used in the security feedback mechanism from the edge-router to the real-time sources.

Figure 3.2: Environment Topology.
According to the object-oriented agent-based design phases, our research environment was decomposed into six interactive entities from both hardware and software categories. The hardware entities are: source, destination, and buffer queue entities. From the other side, the software entities are: coordinator, server, and scheduler entities. Four of the previous entities were designed and implemented at the edge router, which are: coordinator, buffer queue, server, and scheduler. In the modeling phase, we have modeled each entity by an active agent through specifying its main functionalities and behaviors. Fig. 3.3 shows the interaction and data transfer schemes between the cooperative agents.

Figure 3.3: Multi-agent System Model. Solid Line: Data Transfer; Dashed Line: Interaction.
3.2.1 Source Agent

This agent is the real-time data packet generator. It generates a real-time data stream from one of two real-time data types: real-time video or audio. The generated real-time data flow \( f \) has the following specifications:

1- Each real-time data packet has a maximum size \( P_s = 1.46 \) KB (1500 bytes), which is the maximum size of the Ethernet packet frame.

2- The source agent sends the real-time packets through the communication links with a rate of \( \lambda_f \).

3- An exponential distribution with a mean \( 1 / \lambda_f \) was used to generate the real-time packet’s inter-arrival time.

4- A uniform distribution was used to generate the associated relative deadline \( D_f \) for each data flow.

5- A quality of service (QoS) requirements is specified for each \( f \) in terms of the flow’s deadline miss rate \( \Phi_f \).

6- According to the implemented security design model (single-layer or weighted multi-layer), a single security service level or multiple of security service levels will be carried by the IP packet as we will see in section 3.3.

The source agent interacts with the coordinator agent by sending requests to serve its real-time data flow within guaranteed QoS requirements.

3.2.2 Coordinator Agent

The coordinator agent is a software agent. This agent is the core of the edge-router sub-agents. It communicates with all other agents to regulate their
functionalities. The coordinator does not have a global view of the entire system. Instead, its location at the edge router (the default gateway of the LAN) makes it capable of interacting with the source agent with known IP address and the destination agent with known MAC address.

As shown from Fig. 3.3, the coordinator agent monitors the whole system behavior through the interactions (asynchronous control signals). All other agents are interacting together through sending such signals to the coordinator, which in turn redirect the signal to the specific real-time agent. The coordinator agent interacts with the edge router agents (scheduler, buffer queue, and server) to guarantee the process of delivering the real-time packets to their destinations within guaranteed QoS requirements.

The coordinator agent is considered as a huge processing unit and data base system. It evaluates the required system parameters and forwards them to the agents to accomplish their own functionalities. This agent reserves the overall performance of the network through implementing an entire feedback mechanism about the network’s status. According to such feedback, the coordinator interacts with the source agent by sending asynchronous control notification messages. Based on the response from the source agent, it readjusts system parameters and forwards them again to the interactive agents. This agent decides when and how to inform other agents to change their behaviors, such that the network will not be congested and the data flows will be served within the requested QoS requirement ($\Phi_f$).
3.2.3 Diff-EDF Scheduler Agent

In our research, we have modeled the scheduler entity with an interactive differentiated earliest deadline (Diff-EDF) scheduler agent. As all other real-time schedulers, Diff-EDF scheduler enforces the required timing constraints on the real-time packets to provide the requested QoS by the source agent. The Diff-EDF is one of the real-time priority scheduling algorithms that had been implemented based on the earliest deadline first (EDF) scheduling algorithm, which uses the flow’s relative deadline ($D_f$) as the scheduling priority key.

The Diff-EDF scheduler implements a shadow function on the flows’ relative deadline. It defines a new deadline called the flow’s effective deadline ($D_{ef}$). The effective deadline will be evaluated according to the arrived flows’ specifications, where the coordinator agent evaluates the shadow parameter ($C_f$) that will be used by the scheduler in the deadline adjustment process. The new generated deadline will be used as the new priority key in the scheduling process.

The diff-EDF algorithm shows high efficiency in serving real-time data traffics with hard critical-time requirements. In performing such effective deadline prioritizing process, higher priorities are assigned to the stream with smaller deadline miss rate ($\Phi_f$), which leads to provide high QoS guarantees to the real-time data streams.

3.2.4 Queue Agent

This agent is another hardware edge-router sub-agent. The buffer queue agent is in contact with both the coordinator through the interaction signals and the Diff-
EDF scheduler through the data transfer interconnection. This agent has a maximum capacity (bounded buffer). It also has a pre-negotiated consumption limit with the coordinator agent, where the buffer queue agent can’t consume more than a percentage of its maximum capacity.

The buffer queue agent performs two main processes: the queuing (storing) process and the de-queuing (fetching) process. In the queuing process, the queue agent places the arriving packets from the scheduler agent in its buffer according to their effective deadlines. This process is in response to a request from the scheduler to the queue through the coordinator agent. In the de-queuing process, the queue agent fetches the packet that is closest to expire (with the smallest effective deadline) and sends it to the scheduler. The scheduler consequently passes the packet to the server. This process is in response to a request from the coordinator agent upon receiving an idle status asynchronous message from the server agent.

A feedback mechanism from the queue agent to the coordinator agent is implemented, through which the queue notifies the coordinator of its buffer usage. If the buffer usage exceeds the pre-negotiated limit, the queue agent sends a feedback to the coordinator. In response, the coordinator notifies the corresponding source agent to adjust its traffic’s specifications such as the sending rate and the QoS requirements. Accordingly, the coordinator agent adjusts its system parameters and sends them again to the interactive agents. Such mechanism avoids dropping the pending real-time packets at the queue side, and thus increasing the chances of guaranteeing the requested QoS requirements.
3.2.5 Server Agent

This agent is last agent in the hierarchy of the edge-router sub-agents. It’s the one that is responsible of serving the real-time data packets that were chosen and sent by the scheduler. It determines whether to serve or drop a packet based on the packet’s remaining time till expiration. If the packet is not expired, the server sends it to the specific destination according to the MAC address with an exponentially distributed service time. Otherwise, it drops the real-time data packet. The service time for the real-time data packets was implemented using an exponential distribution with a mean \( \mu_f \), where \( \mu_f \) is the service rate, such that:

\[
\mu_f = \frac{8 \times B_w}{P_s} \quad (3.1)
\]

Where \( B_w \) is the average aggregate bandwidth needed for both types of real-time traffics (video and audio).

While serving the real-time data packets, the server keeps track of the serving statistics such as the QoS parameter (miss rate), number of served packets for each destination agent, and the time differences between serving each destination’s packets. Such information will be sent to the coordinator agent periodically (every time period \( T \) specified by the coordinator agent). The coordinator agent uses such information in the packet’s security enhancement process.

A feedback mechanism from the server agent to the coordinator agent is implemented, through which the server notifies the coordinator of its miss rate values. If the server notices a miss rate values near the required miss rate, it sends a feedback notification message to the coordinator. In response, the coordinator notifies the
corresponding source agent to adjust its data stream characteristics. Accordingly, the coordinator agent adjusts its system parameters and sends the new QoS values to the server agent.

3.2.6 Destination Agent

This agent is a hardware agent that modeled the physical end user (end host) at the LAN environment. According to our research, the destination agent performs a first-come first-served (FCFS) scheduling algorithm on the receiving packets from the server. The destination agent has two main parameters:

1- The processing speed rate \((P_f)\) for traffic flow \(f\). This parameter will be sent to the coordinator at the network initiating phase.

2- The size of its available buffers \((B_f)\), which will be used for accommodating arrived packets of traffic flow \(f\) from the server agent.

At every time period \(T\) that is specified by the coordinator at the network initiating process, the destination agent sends its available buffers \((B_f)\) parameter to the coordinator. Such parameter along with the processing rate parameter will be used by the coordinator in the packet’s security enhancement process based on the best utilization of the destination’s buffers. The process protects the network from being congested by heavy traffic load, and thus reserving the overall performance of the network.

3.3 Security Service Design Models

Nowadays, real-time data packet generators apply security services to their data applications to combat the different network security threats. The current service
providers (network technologies) are in care of guaranteeing the flows’ QoS requirements without enhancing the packets’ security requirements. Besides providing the requested QoS requirements, our multi-agent system adaptively enhances the security requirements of the real-time data packets making them robust against different LAN security threats, where LAN is the most preferred environment for network security hackers.

The enhancement process is mainly based on the status of the network, where it guarantees that no congestion process will be caused by such security enhancement mechanism. This could be achieved by operating an efficient resource estimation mechanism to work side by side with the security enhancement unit. According to the type of security threats in the environment, the end user processing speed capabilities, the network performance metrics, and the format of the IP packet carrying the payload, two main security service models were designed in this work: single-layer and weighted multi-layer security service design models.

3.3.1 Single-layer Security Service Design Model

According to the single-layer design, the system was modeled to treat a single security service threat. Such systems will be efficient for environments with limited resources such as limited processing speed capabilities, or limited available network buffering system. It could also be efficient for static and specific purpose environments, where a single specific security threat attacks such environments. This design model applies one of the following security services on the real-time data packets: confidentiality, integrity, or authentication.
Confidentiality security service was implemented to face the sniffing (snooping) security threat, which is the unauthorized monitoring and interception for the flowed data packet stream. For confidentiality single-layer security service design model (Single-conf.), a confidentiality security service level with a range from 1 to 8 was applied to each real-time data packet. Such security level indicates one of the eight cryptographic algorithms used by the source agent to apply the confidentiality security service on the real-time packet. Index 1 indicates the weakest cryptographic algorithm, while index 8 indicates the strongest cryptographic algorithm.

Table 3.1 shows such cryptographic algorithms, which are based on a study performed on 175 MHz processor machine [12].

<table>
<thead>
<tr>
<th>Index ($j$)</th>
<th>Algorithm</th>
<th>$S^c_j$</th>
<th>$\mu^c_j$ (KB/ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SEAL</td>
<td>0.08</td>
<td>168.75</td>
</tr>
<tr>
<td>2</td>
<td>RC4</td>
<td>0.14</td>
<td>96.43</td>
</tr>
<tr>
<td>3</td>
<td>Blowfish</td>
<td>0.36</td>
<td>37.5</td>
</tr>
<tr>
<td>4</td>
<td>Knufu/Khafre</td>
<td>0.40</td>
<td>33.75</td>
</tr>
<tr>
<td>5</td>
<td>RC5</td>
<td>0.46</td>
<td>29.35</td>
</tr>
<tr>
<td>6</td>
<td>Rijndael</td>
<td>0.64</td>
<td>21.09</td>
</tr>
<tr>
<td>7</td>
<td>DES</td>
<td>0.90</td>
<td>15</td>
</tr>
<tr>
<td>8</td>
<td>IDEA</td>
<td>1.00</td>
<td>13.5</td>
</tr>
</tbody>
</table>

According to Table 3.1, $\mu^c_j$ is the data rate in KB/ms that can be enhanced using the $j^{th}$ cryptographic security algorithm. $S^c_j$ is a number between 0.08 and 1, which indicates the efficiency of the security algorithm with respect to the strongest algorithm (IDEA), such that:

$$S^c_j = 13.5 / \mu^c_j$$

(3.2)

As we can see from Table 3.1, the differentiation between the security...
algorithms is in the size of data that can be encrypted within the unit time (data rate). The strongest algorithm provides the highest security level, but the size of data encrypted using such algorithm will be the least. There is no absolute best cryptographic algorithm to be applied on the real-time data traffic. The best algorithm to be used is mainly depends on the status of the network, the requested QoS, the power speed capabilities, the measured NPMs, and the characteristics of the data streams such as arrival-time, service-time, and the relative deadline.

Integrity security service was implemented to combat the alteration security threat, which is the unauthorized attempt to change the real-time packet’s payload, while crossing through the packet’s route (hob-nodes). Integrity security service was implemented using a variety of hash functions. For our integrity single-layer security service design model (Single-intg.), an integrity security service level with a range from 1 to 7 was applied to each real-time data packet. Such security level indicates one of the hash functions used by the source agent to apply the integrity security service on the real-time packet, with index 1 indicates the weakest integrity algorithm and index 7 indicates the strongest one as shown in table 3.2.

According to table 3.2 with a processing speed equals to 175 MHz, $\mu_{j}$ is the data rate in KB/ms that can be enhanced using the $j^{th}$ hash function for integrity security service. $S_{j}^{g}$ is a number between 0.18 and 1, which indicates the efficiency of the security algorithm with respect to the strongest algorithm (Tiger), such that:

$$S_{j}^{g} = 8.5 / \mu_{j}^{g}$$

(3.3)
Table 3.2: Integrity Algorithms.

<table>
<thead>
<tr>
<th>Index (j)</th>
<th>Algorithm</th>
<th>$S_j^g$</th>
<th>$\mu_j^g$ (KB/ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>MD4</td>
<td>0.18</td>
<td>46.4</td>
</tr>
<tr>
<td>2</td>
<td>MD5</td>
<td>0.26</td>
<td>33.2</td>
</tr>
<tr>
<td>3</td>
<td>RIPEMD</td>
<td>0.36</td>
<td>23.3</td>
</tr>
<tr>
<td>4</td>
<td>RIPEMD-128</td>
<td>0.45</td>
<td>18.9</td>
</tr>
<tr>
<td>5</td>
<td>SHA-1</td>
<td>0.63</td>
<td>13.4</td>
</tr>
<tr>
<td>6</td>
<td>RIPEMD-160</td>
<td>0.77</td>
<td>11.1</td>
</tr>
<tr>
<td>7</td>
<td>Tiger</td>
<td>1.00</td>
<td>8.5</td>
</tr>
</tbody>
</table>

Authentication security service was applied on the data packet to face the spoofing security threat in the LAN environment. Spoofing (aliasing) is a fake representation performed by a hacking entity, which starts to send its stream of packets with a fake source MAC address. Such MAC address returned to a well identified network node, and thus the original identity of the hacking source will be hidden. According to the single-layer design model, an authentication model was designed (Single-authn.) that applies an authentication security service level to each real-time data packet. The authentication level ranges between 1 and 3, which indicates one of the three encryption algorithms used by the source agent to apply the authentication security service on the real-time packet as shown in table 3.3.

Table 3.3: Authentication Algorithms.

<table>
<thead>
<tr>
<th>Index (j)</th>
<th>Algorithm</th>
<th>$S_j^a$</th>
<th>$\mu_j^a$ (KB/ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>HMAC-MD5</td>
<td>0.55</td>
<td>16.1</td>
</tr>
<tr>
<td>2</td>
<td>HMAC-SHA-1</td>
<td>0.91</td>
<td>9.1</td>
</tr>
<tr>
<td>3</td>
<td>CBC-MAC-AES</td>
<td>1</td>
<td>8.8</td>
</tr>
</tbody>
</table>

According to table 3.3, index 1 indicates the weakest authentication algorithm, while index 3 indicates the strongest one. $\mu_j^a$ is the data rate in KB/ms that can be
enhanced using the $j^{th}$ authentication security service algorithm on a 175 MHz processor machine. $S^a_j$ is a number between 0.55 and 1, which indicates the efficiency of the security algorithm with respect to the strongest algorithm (CBC-MAC-AES), such that:

$$S^a_j = 8.8/\mu^a_j \quad (3.4)$$

### 3.3.2 Weighted Multi-layer Security Service Design Model

According to the weighted multi-layer design, the system was modeled to be robust against the three most common security threats in the LAN environment (sniffing, alteration, and spoofing) with multi-processor end nodes. Such design will be efficient for networks with high security requirements, where data hacking process leads to a catastrophe. It will also be required in general-purpose networks, where the chances for different types of hacking processes will be high.

The weighted multi-layer security service design adds additional processing load on the network, where end users are dealing now with different security services rather than single one as in the single-layer design. Accordingly, the network system should be able to provide the required resources needed to handle such security requirements such as the buffering system at the end nodes, the bandwidth of the secure channels, and the processing speed capabilities of the end stations.

Nowadays, most of network technologies provide a static security level for each security service during the network serving process; hence, there should be a balance between the requested QoS requirements of the data traffics and the level of each security service provided, such that no traffic will lose its QoS due to the high
security requirements in the system, and the overall performance of the network will be reserved without any network congestion.

In such model, the IP packet should be designed to carry different security levels, one for each security service (confidentiality, integrity, and authentication). The security level to be chosen for each security service depends on two main factors: the control congestion mechanism and the pre-defined weights for each security service. The control congestion mechanism depends on the resource estimation mechanism for the end stations (available buffers and processing speed), while the weights of the security services are defined by a security threshold serving vector $\psi$, where $\psi = (\psi^c, \psi^g, \psi^a)$.

Each vector’s element reflects the percentage of the available destination’s shared buffer that could be used by each processor for the confidentiality, integrity, and authentication security service respectively, and thus the security algorithm (security level) that will be adopted for each security service. The threshold vector will be defined by the source agent at the network initiating process, and will be passed to the control agent that uses such vector values in the security enhancement process such that:

$$\sum_{i \in \{c, g, a\}} \psi^i = 1$$  \hspace{1cm} (3.5)

### 3.4 Evaluating System Parameters

According to agent-based methodology, each agent should have some knowledge about the whole environment, which helps the agent to perform its main sub-tasks.
The environment could be identified by its system parameters; accordingly, the first phase in the simulation process for any environment is to initialize system parameters. In our agent-based system, the coordinator agent is considered as a global database engine that collects system parameters, and passes them through agents. Such parameters include Ethernet packet size, system aggregate bandwidth, and traffics characteristics that include packets arrival-rate, packets service time, and type of packets.

3.4.1 Evaluating Ethernet Packet Size

The IEEE 802.3 standard defines the maximum size of the Ethernet packet’s payload to be 1500 bytes. This doesn’t mean that the real-time packets flow through the network with their maximum capacity. Packets could be found in the network with different sizes ranging from 65 bytes (header fields (19 bytes) + minimum payload (46 bytes)) up to 1519 bytes (header fields (19 bytes) + maximum payload (1500 bytes)). The IEEE 802.3 Ethernet frame format is shown in Fig. 3.4. The differentiation in the packet’s size returns to different factors such as the type of packet (control packet or data packet), the size of data to be sent through the network, and network resource limitations such as bandwidth and processing capabilities.

The size of the real-time data packet is an important environment’s parameter that affects our system agents’ functionalities. It affects the functionality of the server agent, where the packet size is a key parameter in the packet’s service time. The queue agent will also be affected by the packet size, where the packet size determines the required number of available buffers needed to accommodate such data traffic.
The packet size affects the encryption process time at the source agent, and thus it affects the decryption process time at the destination agent.

![IEEE 802.3 Ethernet Frame Format](image)

**Figure 3.4: IEEE 802.3 Ethernet Frame Format.**

Our security enhancement process at the coordinator agent depends on the packet’s size, where the system chooses the best security level to adopt, such that no congestion occurs in the network. Such congestion could be happened in the case of using large sizes of data packets with limited processing capabilities; as a result, the size of the Ethernet packet affects the overall system performance in terms of total end-to-end delay and utilization, which reduces the chances of guaranteeing the QoS requirements of real-time data traffics, and leads to a catastrophe when dealing with hard real-time data streams.

In our research we have used the standard IEEE 802.3, which has a packet size of 1500 bytes. For security enhancement implementation process, we have used the IEEE 802.1Q format, which modifies the IEEE 802.3 Ethernet frame format by adding a field with a size of 4 bytes between the source MAC address field and the type/length field. The choice of using the maximum range of packet size was based on a work performed on a real-time video network. The work illustrates the
distribution of packet sizes during the entire serving process. Simulation results show that less than 1% of the packets are within the lowest packet size range (65-128 bytes), while about 50% of the packets are in the high range (1025-1518) as shown in Fig. 3.5.

Figure 3.5: Video Packet Size Distribution.

3.4.2 Evaluating the Average Aggregate Bandwidth

The bandwidth is defined as the average rate of transmitted data over the communication data link, which is measured by bits/second. This metric (bandwidth) reflects the overall utilization of the data link, and thus it affects the overall performance of the network. For real-time networks with multi-data terminals, the network utilization is reflected by the aggregate bandwidth, which is the sum of the data rates that are delivered to all terminals in a network.
In a real-time general purpose network with different real-time classes flowing through the data links, the service provider should have the capability to provide the required bandwidth that accommodates such different real-time data classes. At any instance of time different classes of data traffics consume the available bandwidth, such that the consumption’s percentage differs from one class to another. One of the most important factors that specify such consumption ratio is the QoS requirements for the real-time data traffics.

According to any real-time packet network technology, the real-time scheduler defines the sequence of packets to be served, such that the QoS requirements for the different classes are achieved. According to the relative deadline rates for the data streams, the video class has the lowest deadline rate followed by the audio traffic. Best effort traffics such as text have the highest relative deadline rates. Based on such deadline distributions, the video packets consume much more bandwidth than other classes (audio, best-effort), where the scheduler giving such traffic higher priority than others to guarantee the required QoS requirements for the data flows.

Assuming that the consumed bandwidth by different data classes in a general purpose network is defined by the bandwidth vector $B = (B^1, B^2, \ldots, B^n)$, where $B^1$, $B^2$, $\ldots$, $B^n$ are the bandwidth consumption rates for each data class such as video, audio, and text. The average aggregate bandwidth needed to serve the different data classes could be defined as $B_w$, where:
\[ B_w = \frac{\sum_{i=1}^{n} (B^i \times (\lambda_i / \sum_{j=1}^{n} \lambda_j))}{n} \]  

(3.6)

Where \( \lambda_i \) is the sending rate for traffic \( i \). According to our research, we only deal with video conference traffic (video, audio). The sending rate from each data source is the same; accordingly, we define our research average aggregate bandwidth to be:

\[ B_w = (B^v + B^a) / 2 \]  

(3.7)

\( B^v \) and \( B^a \) are the bandwidth consumption rates for video and audio data classes respectively. The average aggregate bandwidth \( (B_w) \) plays a key role in guaranteeing the QoS requirements for the real-time data traffics. It directly affects the functionality of the server agent, which is responsible of completing the process of serving the real-time packet. The aggregate bandwidth is one of real-time flow’s service rate \( (\mu_i) \) parameters as shown in equation 3.1.

According to equation 3.1, the system with higher average aggregate bandwidth \( (B_w) \) and lower packet size \( (P_s) \) will be more efficient in the process of serving the real-time data packet and delivering it to the destination within the lowest delay overhead, and thus the chance of guaranteeing the traffic’s QoS requirements will be high.
CHAPTER 4

IMPLEMENTATION OF REAL-TIME SECURITY-AWARE SCHEDULER FOR PACKET SWITCHED NETWORKS

4.1 Introduction

In this chapter we present the process of implementing and developing our real-time software multi-agent simulation system, which simulates a real-time security aware scheduler for packet switched networks. The system was implemented using the .NET platform, which is a high level object oriented modeling, designing, and implementation platform that has capability of simulating different complicated real-time data communication and networking applications such as real-time scheduling, routing, load balancing, network maintenance, and network security applications.

According to software engineering models, we have used the waterfall software engineering model, where the progress of developing the final software product looks like flowing downwards as a waterfall, and passing through the different software engineering models in a sequential mode [waterfall] as shown in Fig. 4.1. The development phase in such model requires a pre fully-identified design, where no iterative development (back and forth between phases) takes place as in the spiral software engineering model.

Based on the previous requirements for the design model, we have modeled and designed our system using a real-time software multi-agent system. According to the followed design model three main characteristics are exist: agent-based, software,
and real-time. Using agent-based system solves the complexity and limitations of using conventional simulation based systems, where the environment will be represented by a number of interactive entities that are cooperated to accomplish the main tasks of the system. Being more specific, we have used a software agent system. Software agent systems are object-oriented agents, which makes them capable of inheriting the required object-oriented based terminologies and methodologies, and thus providing the agent system with huge capabilities that accurately model the entire environment. Going deeper, we have adopted a real-time software agent system, where each agent should synchronize its interactions with the required timing constraints.

![Waterfall Software Engineering Model](image)

**Figure 4.1: Waterfall Software Engineering Model.**

The previous three design characteristics allow the developer to go systematically and smoothly from the design phase to the implementation phase,
which could be adopted using different high-level object oriented programming languages.

4.2 Brownian Motion Queuing Model for Workload Process Implementation

Static queuing theory models were adopted to implement different network and communication applications such as network scheduling, network processing schemes, network routing, and load-balancing techniques. Static queuing theory models work efficiently under certain environments such as networks with best-effort traffics (non real-time), non-prioritized networks, non-preemptive network technologies, networks with predefined traffics’ specifications, networks with non heavy load traffics, networks with no dynamics or mobility, and networks with non complicated traffics’ demands such as time-critical QoS and security requirements.

Accordingly, our security-aware scheduling problem can’t be modeled using the conventional static queuing theory models, where the environment is a real-time heterogeneous network with priority base scheduling problem, adaptive security-aware scheme, time-critical QoS requirements, buffer estimation methodology, dynamic topology, and heavy traffic load. Based on such specifications, the selection of the next real-time packet to be served by the scheduler depends on the current packet under served, the specifications of arrived data packets, the status of the end station to be used for serving the next data packet, QoS requirements, and network performance metrics.

In our model, different real-time data packet generators send their real-time traffics to the scheduler to be served; such real-time traffics have different flows’
specifications such as relative deadline, requested QoS (miss rate), and arrival-time. According to such specifications, the new arrived packets to the scheduler may change the priority of the pre-arrived and queued packets in the scheduler; they may also cause the preemption of the underserved packet (when an arrived packet is closer to expire than the underserved packet). As a result, we can’t have a list of priority-ordered packets that should be put at the top of the static queuing system to be served with the same pattern.

To control such limitations of static queuing theory models, our security-aware scheduling workload process was modeled and implemented using Brownian motion queuing theory model [159]. Brownian motion (also known as Weiner process) is a stochastic and sophisticated random number generator distribution process \( W(t,w) \), which is a stationary random process with both time-independent random increments and continues paths. The parameters of the heterogeneous environment and the flows’ specifications will be reflected through the workload process motion drift parameter \( (\theta) \), which drives the process of generating weighted random instances from different classes of real-time network streams, such that:

\[
\theta = \frac{2(1 - I)}{\sum_{j=1}^{N} \lambda_j (I_j^{2} \sigma_{i,j}^{2} + \sigma_{2,j}^{2})}
\]  

As we can see from equation 4.1, the motion drift parameter \( (\theta) \) was evaluated according to the real-time environment’s parameters:
1- \( N \): The number of real-time data packet sources (generators), where each real-time source is connected to a specific destination in the LAN environment and sends one type of real-time traffics (video or audio).

2- \( \lambda_f \): The sending rate for real-time traffic \( f \).

3- \( I_f \): The intensity of the real-time traffic \( f \), such that:

\[
I_f = \frac{\lambda_f}{\mu_f}
\]  
(4.2)

Where \( \mu_f \) is the service rate of the real-time traffic \( f \), which can be evaluated using equation 3.1. Accordingly, the total intensity of all flows that arrived to the edge router is defined as \( I \), such that:

\[
I = \sum_{i=1}^{N} I_f
\]  
(4.3)

The portion of intensity gained by flow \( f \) is defined by the flow’s parameter \( \alpha_f \), such that:

\[
\alpha_f = \frac{I_f}{I}
\]  
(4.4)

4- \( \sigma_{1f} \): the standard deviation of the inter-arrival time for real-time flow \( f \).

5- \( \sigma_{2f} \): the standard deviation of the service time for real-time flow \( f \).

According to our multi-agent design model, the coordinator agent models the real-time security-aware scheduling process with an exponential Brownian distribution workload process with a mean equals to the inverse of the motion drift parameter (\( \theta^{-1} \)). Such motion drift parameter will control the functionalities of the different real-time agents in the multi-agent system. As we will see later, the drift
motion parameter allows the coordinator agent to decide whether to accept the requests to serve the real-time sources’ flows or not. It will also be a key parameter in the shadow function, which will be performed by the schedule agent to evaluate the new effective deadlines for the real-time flows. From the other side, the previous shadowing will affect the processes of both buffer queue agent and server agent, where the queue agent queues the real-time data packet based on the effective deadline, while the server agent serves the real-time packet based on the validity of the effective deadline (expiration time).

As a result, the process of evaluating the appropriate motion drift parameter will be a key factor behind guaranteeing the required QoS requirements of the different real-time flows, preserving the overall performance of the real-time heterogeneous network, upgrading the security service levels of the real-time data packets, and provide a best utilization of the network’s resources at different layers of the OSI model, which leads to protect the network from being congested by heavy traffic loads.

4.3 System Methodology

Our security-aware scheduling algorithm was designed using real-time object-oriented multi-agent systems. According to the multi-agent design, the real-time agents interact together through a well defined protocol, which defines the real-time interactions (asynchronous control messages) and the data transfer schemes between the real-time agents. In this section we provide an intensive description of the real-time protocol that controls the timing transactions of the multi-agent system in a
packet switched network. We also provide the implementation of our security enhancement methodology for both single and weighted multi-layer design models. Finally, we provide a feedback mechanism model that monitors the status of the network, protects the network from being congested by heavy traffic loads, and guarantees providing the required QoS requirements to the different real-time data classes.

4.3.1 Real-time Multi-agent Timing Protocol for Scheduling

Our real-time multi-agent timing protocol begins when each source agent requests a schedule for its real-time data stream within guaranteed QoS requirements. Such requests will be in terms of interconnections (asynchronous control messages) that are directed from each source agent to the coordinator agent, where each request carries the specifications of each source’s real-time traffic flow. Upon receiving such requests, the coordinator agent starts to evaluate the system parameters needed to model the security-aware scheduling workload process. The coordinator models the workload process as a general Brownian motion queuing theory model with a negative motion drift parameter (-θ) as shown before in equation 4.1.

After evaluating the motion drift parameter, the coordinator agent starts to evaluate the shadow scheduling parameter (C_f), which will be used by the Diff-EDF scheduler to evaluate the effective deadline (D_{ef}) of the real-time data flow. With the smallest deadline miss rate among all the arrived real-time data flows being Φ_{min}, the coordinator agent obtains the parameter C_f, such that:
\[ C_f = \theta^{-1} \log \left( \frac{\Phi_f}{\Phi_{\text{min}}} \right) \] (4.5)

\( \Phi_f \) is the required deadline miss rate of traffic \( f \), which was sent by the source agent as one of the traffic’s parameters. Such parameter reflects the level of QoS requested by the source agent and will be sent at the pre-negotiation phase between the source and coordinator agents. After evaluating such shadowing parameter \( (C_f) \), the coordinator agent evaluates the feasibility of serving such sources’ requests by estimating each real-time flow’s deadline miss rate \( (\hat{\Phi}_f) \) such that:

\[ \hat{\Phi}_f = \exp \left( -\theta \left( D_{\text{avg}} - C_f \right) \right) \] (4.6)

Where \( D_{\text{avg}} \) is the average effective deadline for all arrived real-time data flows, such that:

\[ D_{\text{avg}} = \sum_{f=1}^{N} I_f \left( \Phi_f + C_f \right) \] (4.7)

In evaluating the feasibility of serving the sources’ requests, the coordinator agent checks whether the estimated deadline miss rate for each real-time data flow \( (\hat{\Phi}_f) \) meets the requested flow’s QoS requirements \( (\Phi_f) \), that is, if \( (\hat{\Phi}_f < \Phi_f) \), the coordinator agent performs the following transactions:

1- Interacting with the source agent by sending it an asynchronous acceptance message to serve its real-time data stream.
2- Interacting with the Diff-EDF scheduler agent by passing it the effective deadline shadow parameter ($C_f$).

3- Interacting with the server agent by passing it the traffic’s deadline miss rate parameter ($\Phi_f$).

Once the source agent receives the coordinator’s acceptance message, it starts to send its real-time data packet stream to the Diff-EDF scheduler agent. When it receives the real-time data packet, the Diff-EDF scheduler a shadow function on the packet’s relative deadline to obtain the new scheduling priority key, that is the packet’s effective deadline ($D_{ef}$), such that:

$$D_{ef} = D_f + C_f$$

(4.8)

The previous shadowing process increases the chances of guaranteeing the QoS requirements of the real-time data flows. According to equation 4.5, the shadowing parameter ($C_f$) will be higher for video flows than audio flows, where the deadline rates for video flows are lower than audio ones. Among video flows, the flow with the lowest deadline miss rate will get the smallest value for $C_f$, which will be 0, where log (1) = 0. Accordingly, the video flow with the lowest deadline miss rates will get lower effective deadline, and thus higher scheduling priority.

Once the packet’s effective deadline is evaluated, the Diff-EDF scheduler agent forwards the packet to the buffer queue agent. It also sends a request to the coordinator agent to generate a queuing control signal. The coordinator generates such queue signal (asynchronous control message) and forwards it to the queue agent,
which queues the packet according to its effective deadline. According to our multi-agent design, all control requests should be sent through the coordinator agent, which explains the previous interactions between scheduler, queue, and coordinator agents.

The functionality of the server agent is to complete the process of serving the real-time data packets. Once the sever agent completes serving a current real-time data packet, it interacts with the coordinator agent by sending an idle status message. Accordingly, the coordinator agent responds by interacting with the queue agent through a fetching control signal. The queue agent responds to such signal by retrieving the real-time packet from the top of the queue; such packet has the smallest effective deadline, and thus it’s the closest to expire. The queue agent then forwards the fetched packet to the Diff-EDF scheduler, which passes it to the server agent.

Once it receives the real-time packet from the scheduler agent, the server agent performs the following transactions:

1- Modifying its current status from idle to busy, and informing the coordinator agent with its new status.

2- Serving the unexpired data packet (doesn’t exceed its deadline) according to its service time or dropping the expired data packet.

3- Forwarding the served real-time data packet to its destination in the LAN environment according to its MAC address.

4- Keeping track of two main counters:

   a- \( n_f \): the number of packets served for the destination of traffic flow \( f \).
b- \( t_f \): the cumulative sum of time differences of served packets for the
destination of traffic \( f \), such that:

\[
\sum_{i=1}^{n_f} (t_i - t_{i-1})
\]

(4.9)

### 4.3.2 Real-time Multi-agent Security Enhancement Timing Protocol for Single
and Weighted Multi-layer Models

Over every time period \( T \) that is specified by the coordinator agent at the
network initiating process, the coordinator agent interacts with both the server and the
destination agents requesting for the server’s counter information (\( n_f \) and \( t_f \)) and the
destination’s resource information (\( B_f \) and \( P_f \)). Upon receiving such information, the
coordinator agent evaluates the mean inter-arrival time (\( 1/\xi_f \)) for the packets of real-
time traffic flow \( f \) delivered to their destination, such that:

\[
1/D_f = t_f/(n_f - 1)
\]

(4.10)

According to the single-layer security service design model, the coordinator
agent stores in its data base the required information for enhancing the packet’s
security level of the security service \( x \), where \( x \in \{ \text{confidentiality (c), integrity (g), or}
authentication (a) \} \) as shown in table 3.1, table 3.2, and table 3.3 respectively. Based
on that, the coordinator agent determines the length of buffer \( (L^x_{f}) \) that is needed to
enhance \( n_f \) real-time packets using the \( f^{th} \) security algorithm of the \( x \) security
service, such that:

\[
L^x_{f} = \rho_f \xi_f
\]

(4.11)
where $\rho_f^x$ is the total processing time for the packets of traffic flow $f$. Such processing time takes into account two main processing delays:

1- $D_{f, \text{equal-priority}}$: Delay of solving the problem of two or more equally prioritized data packets.

2- $D_{f, \text{priority}}$: Delay of the preemption process; such delay occurs when an arrived packet is closer to expire than the remaining time of the currently under-process packet.

Accordingly to the previous delays, we have:

$$\rho_f^x = D_{f, \text{equal-priority}} + D_{f, \text{preemption}} + \tau_{fj}^x$$

(4.12)

Where $\tau_{fj}^x$ is the time required to decrypt a real-time data packet with a size $P_s$ equals to 1500 bytes (1.46 KB) using the $j^{th}$ security service, such that:

$$\tau_{fj}^x = P_s/\left(\mu_j^x \beta\right)$$

(4.13)

Where $\beta$ is the processing speed rate factor that is used to handle end users (destinations) with different processing speed capabilities, such that:

$$\beta = P_j / 175 MHz$$

(4.14)

The coordinator then compares the length of available buffers at the destination agent with the length of buffers needed to enhance $n_f$ real-time data packets using different $x$ security service algorithms. The coordinator agent adopts the strongest security service algorithm, such that no congestion occurs in the network. According to that, the coordinator tries to keep a balance between enhancing the
security level of the real-time data flows and preserving the overall performance of the network.

Given that the length of available buffers at the destination of traffic $f$ is $B_f$, and the length of buffers needed to enhance $n_f$ real-time data packets to security level $z^x$ (security level $z$ of the $x$ security service) is $L_{fz}^x$. The coordinator enhances/reduces the security requirements to level $z^x$, or stays at the same security level $z^x$, such that:

$$L_{fz}^x \leq B_f < L_{f(z+1)}^x$$  \hfill (4.15)

According to the weighted multi-layer security design model, the coordinator agent uses its security data base to determine the buffer length vector $(L_{c}^x, L_{g}^x, L_{a}^x)$ that is needed to enhance $n_f$ real-time data packets using the three security service algorithms (confidentiality ($c$), integrity ($g$), and authentication ($a$)). The process of evaluating the value of each vector’s entry ($L_{i}^x$) is obtained using equations (4.11, 4.12, 4.13, and 4.14). According to the security threshold function ($\psi$), the coordinator agent evaluates the portion of the destination’s available buffer ($B^x_f$) that can be used by each security service algorithm on each processor, such that:

$$B^x_f = \psi^x_f \times B_f$$ \hfill (4.16)

According to equation 3.5, the whole destination’s available shared buffer ($B_f$) could be defined as the following:

$$B_f = \sum_{i\in\{c,g,a\}} B^i_f$$ \hfill (4.17)
The coordinator agent compares the length of destination’s available buffers reserved for each security service \((B^x_f)\) with the required length of buffers to enhance \(n_f\) packets using the three security services’ algorithms \((L^x_f)\). According to the comparison process, it adopts the appropriate security level for each security service, such that QoS, NPM, and buffer utilization are achieved in the network. Given that the length of available buffers at the destination of traffic \(f\) is \(B_f\), the length of buffers needed to enhance \(n_f\) packets to security level \(z^x\) is \(L^x_f\), and the threshold value for such \(x\) security service is \(\psi^x_f\). The coordinator enhances/reduces security to level \(z^x\), or stays at the same security level \(z^x\) such that:

\[
L^x_{f=} \leq \psi^x_f B_f < L^x_{f(=+1)}
\]  

Once the decision on security level is made, the coordinator agent notifies the source through an asynchronous control message. According to single-layer model, no notification will be sent if the decision was to stay at the same security level. The same criteria will be followed by the weighted multi-layer design, where no security-level notification will be sent if the decision was to stay at the same three security levels. Upon receiving the security-level notification, the source agent applies the security algorithms that lead to the new adopted security levels. Accordingly, the new generated real-time data streams by the source agent will carry the modified security levels, which were optimized by the coordinator, such that no congestion will occur in the network.
4.3.3 Network Congestion Feedback Mechanism

In guaranteeing the QoS requirements for different classes of real-time data flows, our proposed algorithm doesn’t only depend on the preliminary miss rate prediction process. The pre-prediction process doesn’t take into consideration any dynamic future topology changes or any changes in the performance of network’s components (functionality). Such pre-prediction process gives an initial acceptance notification to the source agent to serve its real-time flow, which will be based on the current status of the network.

In order to keep the system updated with the current status of the network that affects the process of guaranteeing the QoS requirements for the real-time streams, our proposed system implements a feedback mechanism that was based on a live network performance monitoring strategy. Such feedback mechanism was implemented between the server and the coordinator agents in both single and weighted multi-layer security design models. As we said before: if the source’s request was accepted, the coordinator agent passes the requested QoS parameter ($\Phi_f$) to the server agent, which will be used by the server in implementing the feedback mechanism.

The server agent monitors its miss rate statistical counter ($n_f$), which reflects the number of packets that was served for the destination of traffic $f$. When the server agent notices a miss rate rate near the requested miss rate limits ($\Phi_f$), it interacts with the coordinator agent through an asynchronous interconnection control message that
considered as a high priority interruption message, such that it preempted the current task of the coordinator agent to implement the message’s subroutine.

According to the received feedback message, the coordinator agent interacts with the source agent by notifying it to adjust its system parameters such as increasing the deadline miss rate limits or decreasing the flow’s sending rate. According to its specifications, the source agent requests a service with the new traffic characteristics and sends such request to the coordinator agent. From the other side, the coordinator adjusts the system parameters and notifies both the Diff-EDF scheduler agent and the server agent with the new effective deadline shadow parameter \( C_f \) and the new QoS requirements \( \Phi_f \) respectively.

Such feedback mechanism not only guarantees the QoS requirements of the real-time traffics but also preserves the overall performance of the network, where real-time traffics that go beyond their QoS limits will not continue flowing in the network, and thus decreasing the overall load in the system, which totally leads to protect the network from the congestion process. The overall timing protocol for our security-aware scheduling system is shown in Fig. 4.2.

The process of designing our system using real-time software agent-based system allows the network’s developer to adopt different high-level object oriented programming languages for the implementation phase. In our development phase, we have used the C# .Net platform as a high level implementation programming tool,
where the huge inherited object oriented real-time capabilities provided by such platform makes it capable of implementing both hardware and software real-time agents. In section 3.2 we have reviewed the main functionalities of each real-time agent, while in this section we will provide the process of implementing such functionalities using the C#.Net OOP tool.

The coordinator agent initializes the real-time network’s parameters, monitors the functionalities of other real-time agents, acts as a global data base that provides the agents with the required data to accomplish their tasks, and provides the network’s analyst with the required simulation results and reports that measures the network’s performance metrics. In implementing such functionalities, the coordinator agent was developed as a graphical user interface (GUI) window form as shown in Fig. 4.3.

According to Fig. 4.3, our system allows the user to initialize the different network’s parameters such as number of real-time sources/destinations in the network, minimum/maximum QoS requiems for real-time video/audio traffics, minimum video/audio traffic rates, minimum destination processing speed, and minimum initial available buffers at the destination side. Such initialization process allows the user to analyze different network topologies, which makes our system efficient for real-time networks with dynamic topology changes.

The simulate button at the GUI form begins the whole simulation process, such that the coordinator agent starts distributing the pre-initialized data to their
specific agents, real-time agents start performing their interactions, and data results will be collected by the coordinator agent.

![Figure 4.3: Initializing System Parameters.](image)

The results button at the GUI form interrupts the real-time agents to send their simulation data results to the coordinator agents, which in turns evaluates the required data reports such as the average miss rate, the average packet’s deadline, the average security level, the average consumption of destination’s buffer, and the average pending packets at the destination side. Such results will be displayed on the GUI form in different list-boxes as shown in Fig. 4.4, where the formula for filling such list-box is given by:

```csharp
listBox.Items.Add("NPM_Name" + NPM_Value.ToString());
```

(4.19)
Our simulation provides a way for offline monitoring technique, where simulation results and reports provided by the coordinator agent could be stored in a database, such that the analyst engineer will have the ability to extract the network performance metrics from the results, and thus analyzing the performance of the current network’s topology and specifying the network’s parameters that should be adjusted to enhance the overall performance of the real-time network within the guaranteed QoS requirements. The offline monitoring is offered by applying the save results button on the main GUI form, where the handler for such button is shown in Fig. 4.5.
The source agent is the agent that generates one of two main real-time data flows (video, audio). Each real-time flow will be specified by its packets’ characteristics such as type, inter-arrival time (ArrT), service time (ServT), security level (SecL), source ID (SrID), and relative deadline (DeadL). In developing such data flows, the source agent implements each data flow an object oriented class called packets, where the attributes of the class are the real-time packet’s parameters as shown in Fig. 4.6.

The inter-arrival time and the service time were implemented using an exponential distribution function, such that $\lambda_f^{-1}$ and $\mu_f^{-1}$ are the inter-arrival time and service time means respectively. The relative deadline was implemented using a uniform distribution, where the boundaries of the uniform distribution are the minimum and maximum QoS requirements that are specified at the GUI main form by the network administrator. The exponential and uniform distributions are shown in Fig. 4.7 and Fig. 4.8 respectively.

```csharp
public void save ()
{
    StreamWriter sw = new StreamWriter("Data Base Full Path", true);
    int count = listBox.Items.Count;
    string strItems = "",

    for (int counter = 0; counter < count; counter++)
    {
        strItems = listBox.Items[counter].ToString();
        sw.WriteLine(strItems);
    }
    sw.Close();
}
```

**Figure 4.5: Creation of Results Data Base.**
The segments generated by the exponential and uniform distributions for each real-time flow were based on the random function generator function. The seed of the random function was evaluated using the IEEEremainder method that is defined in the Math class for different high level platforms; such implementation ensures the independency of the created segments among the real-time flows.
Figure 4.8: Creating Independent Uniform Segments.

According to the previous parameters, the source agent defines an array of type packets for each real-time data stream as shown in Fig. 4.9.

```csharp
public double GeneratingUnifSegment(double MinQoS, double MaxQoS, int RndSeed)
{
    Random Rnd;
    Rnd = new Random(RndSeed);
    double UnifSegment;
    double UniqueIncrement;
    UniqueIncrement = Rnd.NextDouble();
    UnifSegment = MinQoS + (MaxQoS - MinQoS) * UniqueIncrement;
    return x;
}
```

Figure 4.9: Generating Source Packets.

The scheduler agent performs the shadow function on the arrived data packets using equation 4.8. The implementation of such process was based on adding a new parameter to the packets class called the effective deadline (EffDeadL). The process of generating the queue of shadowed packets is shown in Fig. 4.10.

```csharp
for (int counter = 0; counter < NumberOfSources; counter++)
{
    type = counter % 2;
    SecL = Lowest;
    SrID = counter;
    for (int count = 0; count < \lambda_f; count++)
    {
        FlowSeed = GenerateSeed[FlowID];
        InterArr = GenerateExpSegment(\lambda_f^{-1}, FlowSeed);
        ServT = ServiceTime(\mu_f, FlowSeed);
        DeadL = GeneratingUnifSegment(MinQoS, MaxQoS, FlowSeed);
        ArrTime = ArrTime + InterArr;
        packetArr[count] = new Packets(type, ArrTime, DeadL, ServT, SecL, SrID);
    }
    FlowID = FlowID + 1;
}
```
The buffer queue agent performs two main functionalities: packet queuing and the packet fetching. According to the packet queuing process, the buffer queue agent will queue the real-time data packet according to its effective deadline. The implementation of the queuing is shown in Fig. 4.11.

```csharp
public void Shadowing (Packets packetArr, double ShadowParameter, int λᵣ) {
    for (int count = 0; count < λᵣ; count++)
    {
        double EffDeadL;
        EffDeadL = packetArr[count].DeadL + ShadowParameter;
        packetArr [count] = new Packets (type, ArrTime, DeadL, ServT, SecL, SrID, EffDeadL);
    }
}
```

**Figure 4.10: Generating the Packet Effective Deadline.**

The buffer agent was implemented by a dynamic software memory structure called the list-array, where different operations could be implemented using such
queue-structure such as add, remove, search, sort, clear, clone, and copy. In the packet fetching process, the buffer queue agent retrieves the packet that is located at the head of the queue (the closest to expire) and sends it to the server agent to be served. The fetching process could be given by:

\[
\text{PacketList.RemoveAt (Index0);}
\]

(4.20)

The server agent completes the serving process of the real-time packet by either serving the unexpired packet or dropping the expired packet (exceeded its deadline). One of the core system parameters that will be modified by the server agent is the time parameter. The simulation defines a global time parameter, such that the server agent modifies it every time it serves a real-time packet; such modification will be according to the packet’s service time attribute. From the other side, The queue agent modifies the remaining deadlines of the queued data packets according to the service time of the underserved packet. The implementation of the servering process is shown in Fig. 4.12.

The destination agent responds to the request of the coordinator agent and sends its buffer information every time period T. The buffer at the destination agent is implemented by a list-array dynamic queue memory. While serving and decrypting the arrived data packets, the destination agent doesn’t modify the global time parameter since the global time reflects the process of serving the real-time data packets before sending them through the communication data links. In order to implement the resource estimation mechanism, the destination agent performs thebuffersize handler that is given by:
int buffersize = InitialBufferSize – ArrivedPacketList.Count  \hspace{1cm} (4.21)

```java
double GlobalTime = InitialSimulationTime;
public void ServePacket(Packets packet)
{
    if (packet.DeadL > (GlobalTime - packet.ArrTime))
    {
        packet.DeadL = packet.DeadL - packet.ServT;
        GlobalTime = GlobalTime + packet.ServT;
        ModifyDeadlines(double packet.ServT)
    }
    else
    {
    }
}
public void ModifyDeadlines (double service)
{
    for (int count = 0; count < QueuedPackets; count++)
    {
        packetList[count].DeadL = packetList[count].DeadL - service;
    }
}
```

**Figure 4.12: Serving Real-time Packets.**

For both single and weighted multi-layer security design models, the coordinator agent implements the confidentiality, integration, and authentication security data bases using the array data structure. The coordinator agent uses the requested information from both server and the destination agents to evaluate the optimal security level to be adopted by the source agent, such that no congestion occurs in the network and the overall performance of the network is preserved. Since the weighted multi-layer design model is more general, we provide the security service upgrading process using the multi-layer security level design model as shown in Fig. 4.13.
```java
int GlobalConfCounter, GlobalIntgCounter, GlobalAuthCounter = 0;
double [] ConfSecArray = new double [8] {168.75, 96.43, 37.5, 33.75, 29.35, 21.09, 15.0, 13.5 };
double [] IntgSecArray = new double [7] { 46.4, 33.2, 23.3, 18.9, 13.4, 11.1, 8.5 };
double [] AuthSecArray = new double [3] { 16.1, 9.1, 8.8 };
double [] ThresholdArray = new double [3] {CThreshold, IntgThreshold, AThreshold};

public void SecUpgrade (int BuffSize, int ServPackets, double SumArrival )
{
    int ConfCounter, IntgCounter, AuthCounter = 0;
    double ArrivalRate = SumArrival / (ServPackets - 1);
    ArrayList ConfSecList, IntgSecList, AuthSecList;
    for (int qs = 0; qs < 8; qs++)
        int [] ConfBuffSec[qs] = (int)(ConfSecTimeArray[qs] / arrivalrate);
    for (int qs = 0; qs < 7; qs++)
        int [] IntgBuffSec[qs] = (int)(IntgSecTimeArray[qs] / arrivalrate);
    for (int qs = 0; qs < 3; qs++)
        int [] AuthBuffSec[qs] = (int)(AuthSecTimeArray[qs] / arrivalrate);

    for (int qx = 0; qx < 8; qx++)
        if (ConfBuffSec[qx] < (int) (ThresholdArray[1] * BuffSize))
            { // Your code here
                ConfSecList.Add(ConfBuffSec[qx]);
                ConfSecCounter = ConfSecCounter + 1;
            }
    for (int qx = 0; qx < 7; qx++)
        if (IntgBuffSec[qx] < (int) (ThresholdArray[2] * BuffSize))
            { // Your code here
                IntgSecList.Add(IntgBuffSec[qx]);
                IntgSecCounter = IntgSecCounter + 1;
            }
    for (int qx = 0; qx < 3; qx++)
        if (AuthBuffSec[qx] < (int) (ThresholdArray[3] * BuffSize))
            { // Your code here
                AuthSecList.Add(AuthBuffSec[qx]);
                AuthSecCounter = AuthSecCounter + 1;
                if ((ConfSecCounter != GlobalConfCounter) || (IntgSecCounter != GlobalIntgCounter) ||
                    (AuthSecCounter != GlobalAuthCounter))
                    ModifySec (ConfSecCounter, IntgSecCounter, AuthSecCounter);
            }
}
```

Figure 4.13: Weighted Multi-layer Security Level Upgrading Unit.
5.1 Introduction

Nowadays, the evolution of data communication plays a key role in the process of transition from analog commercial applications to digital ones. Digital data communication depends on the fact that computerized systems interact together by sharing both LAN and WAN networks’ resources. The interacted parties request the network provider for certain levels of reliable data transfer, which could be achieved by implementing security services on such data transactions making them robust against network security hacking threats.

In this chapter, we will provide the common static network security protocols. Such protocols add different types of security services with static security levels on the real-time data applications at different environments. We will examine the ability of using such protocols for our security-aware scheduling problem, where both QoS guarantees and security requirements should be provided with the overall performance of the network being preserved, such that no congestion occurs in the network.

According to the limitations of using the static security protocols, we provide a design model for an adaptive IPsec protocol, where the security levels for the data packets are adaptively upgraded. The security enhancement process will be based on a feedback mechanism from the edge router to the IPsec protocol at the source side.
Such asynchronous feedback mechanism takes into consideration the status of the network resources at the destination side. The advantages and limitations of using such feedback-IPsec protocol will be also presented.

In order to solve the limitations of using the feedback-IPsec protocol, we propose a mechanism that eliminates the pre-negotiation phase (security association) between the real-time data generator and the end station. The proposed mechanism is based on overloading the IEEE 802.1Q frame format to be used for security issues. This chapter provides different implementations for the overloading process, such that it serves both single-layer and weighted multi-layer security design models. The proposed mechanism was deployed with our real-time multi-agent model for real-time video and audio packet switched networks with both QoS and adaptive security guarantees provided.

5.2 Static Security Protocols and the Implementation of Feedback IPsec Protocol for Packet Switched Networks

According to the type of requested security services, the network’s technology, the real-time traffic’s type, and performance limitations, different security protocols were implemented to provide different levels of security services to their network’s applications. Such protocols include the secure socket layer (SSL) protocol, the secure/multi-purpose internet mail extensions (S/MIME) protocol, and the IP security suit (IPsec) protocol. SSL protocol is an application layer security protocol that is used for protecting different types of internet applications, especially those that are related to electronic commerce [160].
S/MIME is another application layer security protocol, which is used for applying the required security services (authentication, encryption, and integrity) on different electronic based messaging applications [161]. Such previous security protocols are working above the transport layer of the OSI model (upper layers). Based on that, each network application should be redesigned to be compatible with such protocols. In order to solve such limitation, IPsec protocol suit was implemented; it works at the network layer of the OSI model (layer- 3), and thus it provides the required security services to all IP based applications [162].

IPsec protocol was implemented based on three main protocols: the authentication header (AH), the encapsulation security payload (ESP), and the security association protocol (SA). AH and ESP security protocols are used to apply different types of security services on the real-time data packets such as authentication and integrity by the AH protocol, while the ESP protocol adds the confidentiality service besides those security services provided by the AH protocol. The security association protocol is used by both previous protocols; it defines the security parameters required to establish the secure data channel at the initiation phase such as security service’s algorithm, initialization data, encryption data keys, security modes. It also governs the functionalities of the AH and ESP security protocols [163].

As a result, the IPsec protocol seems to be suitable for our development since we deal with real-time data packets. The problem is that our development depends on adaptively enhancing the security level of the real-time data packets according to the
status of the network, and thus we need to switch from one security level to another by applying different security algorithms on the data packets. According to the IPsec protocol, every time we change the security algorithm on the data packets, we have to terminate the secure session created by the security association protocol, and then renegotiate again on the new security parameters to be used for encryption/decryption processes. This leads to an additional overhead, and thus affecting the overall performance of the network.

According to a study performed on two 206 MHz processor machines, it was found that it requires about 167 ms to complete the handshaking process needed to establish a security association between two hosts [164]. The handshaking process is mainly based on two phases. The first phase is to establish the secure data channel that is needed to exchange the security parameters between the hosts. Such phase was based on the security main mode that implements the Diffie-Hellman key exchange method. The second phase performs the process of exchanging the association parameters through the established secured channel. The calculations ignore the process of deriving the RSA security keys by using preexisting shared keys between the two hosts, which is similar to our development.

In our development, we have implemented a feedback-IPsec protocol with pre-existing shared keys on 175 MHz processor. The developed protocol makes a transition from the static IPsec protocol, which applies static security service levels on the real-time data packets into an adaptive real-time security service protocol that adaptively upgrades the security levels of the real-time data packets. The security
enhancement process is based on the implementation of a feedback mechanism from the edge router about the status of the network, where the edge router performs a resource estimation mechanism at the end stations (LAN stations), and decides the best security level to be adopted, such that no congestion occurs in the network.

The data generator receives the asynchronous feedback control message, and adopts the new security level on the real-time data packets. Due to the security association phase, an additional overhead $D_{sec}$ shall be added to the system with each change in the security level, such that:

$$D_{sec} = 167 ms \times \beta$$

(5.1)

Where ($\beta$) is the security processing speed rate factor and is given by:

$$\beta = P_f / (206 MHz)$$

(5.2)

5.3 Overloading IEEE 802.1Q Frame Format for Single and Weighted Multi-Layer Security Design Models

According to the security association protocol, the data generator and the destination negotiate on the security algorithm that will be applied on the real-time data packets for each security service (confidentiality, integration, and authentication); they also exchange the data keys that will be used by the destination host in the decryption process. For our adaptive real-time security-aware scheduling, a question that appears is: how does the destination host examine the type of security algorithm that was applied by the source host on the real-time data packet without performing a security association between the two parties (source and destination
agents)? The answer for such question was based on which security design model we are using; single-layer or weighted multi-layer design model.

### 5.3.1 IEEE 802.1Q for Single-layer Security Model

According to the single-layer security design model, we have solved such problem by overloading the priority code point (PCP) fields of the IEEE 802.1Q tagged frame format [64]. The IEEE 802.1Q modifies the Ethernet frame format by adding a field with a size of 4 bytes between the source MAC address field and the type/length field as shown in Fig. 5.1. The IEEE protocol was implemented to define the functionality of virtual LANs (VLANs), where the one physical Ethernet network is divided into a group of logically shared networks for security aspects.

![Figure 5.1: IEEE 802.1Q Tag Insertion.](image)

The IEEE 802.1Q tag consists of two main fields. The first field is the tag protocol identifier (TPID) with a size of 2 bytes and a value of 0x8100; such field is used to identify the beginning of the IEEE 802.1Q tagged frame. The second field is the tag control information (TCI) with a size of 2 bytes; such field is divided into
three sub-fields: (1) the priority code point (PCP) with a size of 3 bits, which is used to prioritize the different types of data flows (text, video, audio, etc.) as shown in table 5.1; (2) the canonical format indicator (CFI) with a size of 1 bit, which is used to define the format of layer-2 MAC address of the OSI model; when the value of this field is 0, the address is in a canonical format. It’s always set to 0 for Ethernet frames; and (3) the VLAN identifier (VID) with a size of 12 bits, which specifies the VLAN for which the real-time data belongs. The IEEE 802.1Q registered fields are shown in Fig. 5.2.

### Table 5.1: IEEE 802.1Q PCP Fields.

<table>
<thead>
<tr>
<th>PCP Value</th>
<th>Traffic Type</th>
<th>Traffic Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Background</td>
<td>0 (Lowest)</td>
</tr>
<tr>
<td>1</td>
<td>Best Effort</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>Excellent Effort</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>Critical Applications</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>Video, &lt; 100 ms Latency</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>Voice, &lt; 10 ms Latency</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>Internetwork Control</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>Network Control</td>
<td>7 (Highest)</td>
</tr>
</tbody>
</table>

Since the implementation of the PCP field is left to the user, we have overloaded this field to represent the different priorities of security algorithms that are used for enhancing each individual security service (confidentiality, integrity, and authentication) as shown in table 5.2.
Upon receiving the real-time data packet, the destination agent checks the PCP field and determines the security algorithm that was implemented on the packet. Consequently, by using a hash table for the pre-defined keys, the destination will have the ability of processing the secured real-time data packet.

Table 5.2: Overloading IEEE 802.1Q PCP Fields.

<table>
<thead>
<tr>
<th>PCP Priority</th>
<th>Confidentiality</th>
<th>Integrity</th>
<th>Authentication</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (Lowest)</td>
<td>SEAL</td>
<td>MD4</td>
<td>HMAC-MD5</td>
</tr>
<tr>
<td>1</td>
<td>RC4</td>
<td>MD5</td>
<td>HMAC-SHA-1</td>
</tr>
<tr>
<td>2</td>
<td>Blowfish</td>
<td>RIPEMD</td>
<td>CBC-MAC-AES</td>
</tr>
<tr>
<td>3</td>
<td>Knufu/Khafre</td>
<td>RIPEMD-128</td>
<td>Not Used</td>
</tr>
<tr>
<td>4</td>
<td>RC5</td>
<td>SHA-1</td>
<td>Not Used</td>
</tr>
<tr>
<td>5</td>
<td>Rijndael</td>
<td>RIPEMD-160</td>
<td>Not Used</td>
</tr>
<tr>
<td>6</td>
<td>DES</td>
<td>Tiger</td>
<td>Not Used</td>
</tr>
<tr>
<td>7 (Highest)</td>
<td>IDEA</td>
<td>Not Used</td>
<td>Not Used</td>
</tr>
</tbody>
</table>

5.3.2 IEEE 802.1Q for Weighted Multi-layer Security Model

Since the weighted multi-layer security design model provides the real-time data packet with different security services, the Ethernet frame format should be designed to carry a specific code for each one of such security services (confidentiality, integrity, and authentication).
According to single-layer design model, the security association phase was neglected by overloading the priority code point of the Ethernet frame format to represent the associated codes for a single security service algorithm. Such implementation doesn’t fit our weighted multi-layer model, where the priority code point field has a size of 3 bits, and thus it has the ability to code only 8 security algorithms. From the other hand, the weighted multi-layer model provides the real-time data packet with the three security services (confidentiality, integrity, and authentication), which means that the Ethernet frame should have a specific code for each algorithm of the previous security services: 8 algorithms for confidentiality, 7 algorithms for integrity, and 3 algorithms for authentication. Accordingly, two implementation methods were proposed: 1) repeated single-layer method; and 2) overloading PCP and VID fields of the IEEE 802.1Q tag frame format.

According to the repeated single-layer method, the source agent sends three repeated versions for the same IP packet. The payload at each version will be secured with one of the three security services’ algorithms (confidentiality, integrity, and authentication). Such method follows the single-layer design model to provide the associated codes for each security service algorithms, where the priority code point field will be overloaded for such codes. Although such method neglected the security association phase between the two parties (source and destination), it adds more repeated traffic on the network, and thus additional overhead that may lead to congest the network. Besides neglecting the security association phase, the repeated single-layer method provides a method for data error recovery that is if the destination
checks the cyclic redundancy check (CRC) field of the IP packet for errors and finds an error, it doesn’t need to request the source to resend the packet, since it has another two versions of the same packet.

The previous implementation method doesn’t seem efficient for general-purpose networks, where the three security services are needed to be applied to the packet at the same time. In order to solve the limitations of using the repeated single-layer method, another method based on overloading both the PCP and the VID fields of the IEEE 802.1Q tag frame format was proposed.

The Ethernet frame format should have a security field with a size of 8 bits to code the three security algorithms: 3 bits for the eight confidentiality algorithms, 3 bits for the seven integrity algorithms, and 2 bits for the three authentication algorithms. Accordingly, we overload the 3-bit PCP field and the 5 higher bits of the VID field of the IEEE 802.1Q frame. The overloaded bits in the frame are shown as red b’s in Fig. 5.3.

<table>
<thead>
<tr>
<th>16 bits</th>
<th>3 Bits</th>
<th>1 Bits</th>
<th>12 bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>TPID</td>
<td>PCP</td>
<td>CFI</td>
<td>VID</td>
</tr>
<tr>
<td>0x8100</td>
<td>b_{11}</td>
<td>b_{10}</td>
<td>b_{0}</td>
</tr>
<tr>
<td></td>
<td>b_{11}</td>
<td>b_{10}</td>
<td>b_{0}</td>
</tr>
<tr>
<td></td>
<td>b_{11}</td>
<td>b_{10}</td>
<td>b_{0}</td>
</tr>
<tr>
<td></td>
<td>b_{11}</td>
<td>b_{10}</td>
<td>b_{0}</td>
</tr>
<tr>
<td></td>
<td>b_{11}</td>
<td>b_{10}</td>
<td>b_{0}</td>
</tr>
</tbody>
</table>

**Figure 5.3: IEEE 802.1Q Overloaded Bits for Weighted Multi-layer Model**

The PCP field is overloaded to represent the associated code for each confidentiality security algorithm. The VID bits \{b_{11}, b_{10}, b_{0}\} are overloaded to
represent the associated code for each integrity security algorithm. The VID bits \( \{b_8 b_7\} \) are overloaded to represent the associated code for each authentication security algorithm. Upon receiving the real-time data packet, the destination agent checks the overloaded fields of the IEEE 802.1Q frame and determines the three security algorithms that are adopted. By using a hash table for the pre-defined keys, it can process the packet and extract the original information. The associated codes for the confidentiality, integrity, and authentication security services are shown in table 5.3, table 5.4, and table 5.5 respectively.

Table 5.3: Overloaded PCP Bits for Confidentiality.

<table>
<thead>
<tr>
<th>PCP bits: ( b_2 b_1 b_0 )</th>
<th>Algorithm</th>
<th>Sec. Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>SEAL</td>
<td>1</td>
</tr>
<tr>
<td>001</td>
<td>RC4</td>
<td>2</td>
</tr>
<tr>
<td>010</td>
<td>Blowfish</td>
<td>3</td>
</tr>
<tr>
<td>011</td>
<td>Knufu/Khafre</td>
<td>4</td>
</tr>
<tr>
<td>100</td>
<td>RC5</td>
<td>5</td>
</tr>
<tr>
<td>101</td>
<td>Rijndael</td>
<td>6</td>
</tr>
<tr>
<td>110</td>
<td>DES</td>
<td>7</td>
</tr>
<tr>
<td>111</td>
<td>IDEA</td>
<td>8</td>
</tr>
</tbody>
</table>

Table 5.4: Overloaded VID Bits for Integrity.

<table>
<thead>
<tr>
<th>VID bits: ( b_{11} b_{10} b_9 )</th>
<th>Algorithm</th>
<th>Sec. Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>MD4</td>
<td>1</td>
</tr>
<tr>
<td>001</td>
<td>MD5</td>
<td>2</td>
</tr>
<tr>
<td>010</td>
<td>RIPEMD</td>
<td>3</td>
</tr>
<tr>
<td>011</td>
<td>RIPEMD-128</td>
<td>4</td>
</tr>
<tr>
<td>100</td>
<td>SHA-1</td>
<td>5</td>
</tr>
<tr>
<td>101</td>
<td>RIPEMD-160</td>
<td>6</td>
</tr>
<tr>
<td>110</td>
<td>Tiger</td>
<td>7</td>
</tr>
<tr>
<td>111</td>
<td>Reserved</td>
<td>-</td>
</tr>
</tbody>
</table>
By overloading 5 bits of the VID field for security purpose, we have reduced the number of VLANs that can be configured in a single network. The number is reduced from $2^{12} = 4096$ to $2^7 = 128$. Such shrink deteriorates the network performance in terms of user capacity. However, the user capacity trades off with the bandwidth usage per user. This fits well with the needs of the high-throughput real-time secure data network. The network congestion is avoided by eliminating the overhead of the security associations and reducing the number of VLANs.

<table>
<thead>
<tr>
<th>VID bits: $b_8 b_7$</th>
<th>Algorithm</th>
<th>Sec. Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>HMAC-MD5</td>
<td>1</td>
</tr>
<tr>
<td>01</td>
<td>HMAC-SHA-1</td>
<td>2</td>
</tr>
<tr>
<td>10</td>
<td>CBC-MAC-AES</td>
<td>3</td>
</tr>
<tr>
<td>11</td>
<td>Reserved</td>
<td>-</td>
</tr>
</tbody>
</table>
CHAPTER 6
SYSTEM SIMULATION AND NUMERICAL RESULTS

6.1 Introduction

In this chapter, we will evaluate the efficiency of our proposed adaptive security-aware scheduler using the .Net simulator platform. According to the simulation process, we will show that our proposed system has the following capabilities: (1) providing guaranteed QoS requirements for both video and audio data flows; (2) preserving the overall performance of the network; (3) providing a best utilization for the buffering system in the network; (4) enhancing different security requirements for the real-time data packets; (5) keeping a balance between providing the required security requirement for the data packets and protecting the network from being congested by heavy traffic load; (6) minimizing the total average delays for serving the real-time data packets at both queue and destination agents; (7) minimizing the buffer consumption at the destination agent, which leads to provide a better security enhancement for the real-time data packets; and (8) minimizing the overhead of using the security association phase by the IPsec protocol.

In order to examine the performance of our proposed system, different simulations were carried out. According to the NPM that we are measuring, the performance of our proposed system was evaluated from different real-time agent perspectives.
6.2 Initializing Simulation Parameters

In order to evaluate the performance of our proposed system and achieve our research goals, we have carried out different research’s experiments. The collected data from each experiment was saved in a certain data base to be analyzed using the offline monitoring technique. According to the analysis phase, different NPMs will be measured to evaluate and examine the efficiency of our proposed system.

The first step of running our simulated experiments is to initialize the environment’s parameters, which are required to accomplish the tasks of our proposed real-time multi-agent system. In this work we simulate a packet switched network with $N$ pairs of distinct source and destination, such that each source ($S_i$) is connected to its corresponding destination ($D_i$) through a secure data channel. The number of source/destination pairs ($N$) starts from two pairs and increases by a step of two pairs until it reaches a number of 32 source/destination pairs in the last simulated iteration, such that $N= \{2, 4, 6, \ldots, 30, 32\}$ pairs.

According to our simulation, we assume that our system serves two types of real-time data packets: video and audio. Among the $N$ data traffic streams, we assume that there are $N/2$ real-time video streams and $N/2$ real-time audio ones, such that the single real-time data generator generates only one type of data traffics (video or audio). In our simulation we set the sending rate for each real-time source ($\lambda_f$) to be equals to 1250 real-time packets per second; accordingly, the exponential function that was used to generate the real-time packet’s inter-arrival time will have a mean of $(1/1250 =0.0008 \text{ s})$. The size of the real-time video or audio packet was equals to the
maximum frame size, which has a size of 1500 bytes. According to the implementation of the IEEE 802.1Q, this size will be modified by adding extra 4 bytes for the implementation of the single-layer and weighted multi-layer security design models.

The average aggregate bandwidth used in our research was evaluated using equations 3.6 and 3.7, where the ratio of the overall bandwidth consumption between video and audio streams is 5Mbps : 3Mbps; hence, real-time video streams consume about 62.5 % of the overall bandwidth, while audio streams consumed about 37.5 % of it. We have used an average aggregate bandwidth equals to 4 Mbps for both video and audio flows; accordingly, the mean of the service-time distribution depends on such value as shown in equation 3.1. The required deadline miss ratio of each audio stream is a random variable uniformly distributed on [160 ms, 300 ms] and the one of each video stream is a random variable uniformly distributed on [40 ms, 150 ms]. Accordingly, the minimum deadline miss ratio is $\Phi_{\text{min}} = 40$ ms. The coordinator needs the value of $\Phi_{\text{min}}$ to calculate the parameter $C_f$ for the shadow function.

For both single and weighted multi-layer security design models, we assume that real-time data sources start sending packets with the lowest security level for each security service (confidentiality, integration, and authentication). Such process continues until each source receives a notification from the edge router (coordinator agent) with the appropriate security level to be adopted for each security service. Our research simulations were carried out for different destination’s initial available buffers (maximum buffer size), where the initialization process was based on the
sending rate of the real-time data flow; accordingly, we chose the values of \( \frac{\lambda_f}{25} \), \( \frac{\lambda_f}{8} \), and \( \lambda_f \) (the unbounded case), multiplied by unit time.

Another simulation parameter was the destination’s agent processing speed \( (P_f) \). Our simulations were based on 175 MHz processing speed machines, and thus the processing rate factor \( (\beta) \) will be equals to 1. Such parameter will be passed from destination agent to coordinator agent at the network initiation process, or any time a new end user is installed to the network’s topology. The coordinator periodically performs the security enhancement with a time period \( T = 30 \text{ ms} \). For the weighted multi-layer model, the weights of the three security services are predefined in two security threshold vectors: \( \psi_1 = (0.5; 0.3; 0.2) \) and \( \psi_2 = (0.33; 0.33; 0.33) \) for all real-time audio and video streams.

6.3 The Effect of Destination’s Buffering System on the Overall Network Performance

In this section, we demonstrate the performance of our proposed algorithm at the destination agent through carrying out different simulated experiments. The simulations were carried out for both single-layer and weighted multi-layer security design models. According to this simulation, two main performance metrics were studied: the enhancement of the packet’s security service levels and the traffic’s average total delays at the destination buffer. Fig. 6.1, Fig. 6.2, and Fig. 6.3 show the effect of initial buffer length at the destination agent on the security enhancement process for the confidentiality, integrity, and authentication single-layer security design models respectively.
Figure 6.1: Destination Buffer Effect on Confidentiality Security Level Enhancement.

Figure 6.2: Destination Buffer Effect on Integrity Security Level Enhancement.
Figure 6.3: Destination Buffer Effect on Authentication Security Level Enhancement

As we can see from the previous figures, the average security level for the packets will be enhanced for larger number of $N$ source/destination pairs, since the packets’ arrival rates at each destination decrease for higher number of end nodes in the iteration as shown in Fig. 6.4, and thus the consumption of the destination’s buffer decreases, which totally leads to more flexibility in applying higher security level on the real-time packets. We can also notice that the security service level with larger number of available buffers will be higher, since destinations have more flexibility in decrypting high security level packets without much caring to the number of buffers needed to accommodate the new arrived data packets.
For the weighted multi-layer design model, we have run our simulation using two different weighted threshold vectors: $\psi_1 = (0.5, 0.3, 0.2)$ and $\psi_2 = (0.33, 0.33, 0.33)$. Fig. 6.5 and Fig. 6.6 prove the same results achieved for the previous three single-layer design models.

Figure 6.4: Arrival-rate of Packets at the Destination.
Figure 6.5: Buffer Effect on Security Enhancement (Weighted Multi-layer $\psi_1$).

Figure 6.6: Buffer Effect on Security Enhancement (Weighted Multi-layer $\psi_2$)
However, the previous flexibility advantage in decrypting high security level packets at large initial available buffer length trades off with the quality-of-service (QoS) in terms of average total packet delays at the destination agent, where higher security level achieved by applying more complex security service algorithm needs more processing time from the destination for the packet’s decryption process. Fig. 6.7, Fig. 6.8, and Fig. 6.9 show such trade off for the confidentiality, integrity, and authentication single-layer security design models respectively, while Fig. 6.10 and Fig. 6.11 show it for the weighted multi-layer design model with the two threshold vectors.

![Graph](chart.png)

**Figure 6.7: Buffer Effect on Average Packets Delays (Single-confidentiality).**
Figure 6.8: Buffer Effect on Average Packets Delays (Single-integrity).

Figure 6.9: Buffer Effect on Average Packets Delays (Single-authentication).
Figure 6.10: Buffer Effect on Average Packets Delays (Weighted Multi-layer $\psi_1$).

Figure 6.11: Buffer Effect on Average Packets Delays (Weighted Multi-layer $\psi_2$).
6.4 Adaptive Security-aware Scheduler Vs Static IPsec Protocol

Our adaptive security-aware scheduler provides both security and QoS requirements for real-time data flows; it also provides the best utilization for the buffering system at both destination and queue agents. The level of the buffer-utilization controls the network congestion mechanism, and thus determines the overall performance of the network. In this section, we demonstrate the efficiency of our proposed security-aware system in preserving the overall performance of the network.

In doing that, we compare the performance of our proposed scheme with the IPsec protocol at a static security level for the two cases of initial buffer length; $\lambda_f/25$ and $\lambda_f/8$. The initialized static security level for the IPsec protocol is the steady state security level of the proposed adaptive algorithm as shown in Figures: 6.1, 6.2 and 6.3 for the three single-layer security design models: confidentiality, integrity, and authentication respectively, while the steady state levels for the weighted multi-layer model are shown in Fig. 6.5 and Fig. 6.6.

The utilization of the destination’s buffering system had been reflected through measuring the average consumption of the destination’s buffer in both adaptive scheme and IPsec protocol. The results for applying such simulations on the three single-layer security design models: confidentiality, integrity, and authentication are shown in Figures: 6.12, 6.13, and 6.14 respectively, while Fig. 6.15 and Fig. 6.16 simulates the weighted multi-layer design model using both threshold vectors ($\psi_1$ and $\psi_2$).
Figure 6.12: Average Buffer Consumption at Destination for Confidentiality.

Figure 6.13: Average Buffer Consumption at Destination for Integrity.
Figure 6.14: Average Buffer Consumption at Destination for Authentication.

Figure 6.15: Average Buffer Consumption (Weighted Multi-layer $\psi_I$).
According to our simulations, both adaptive scheme and IPsec protocol apply the same level of security requirements on the real-time data flows. Whereas, our proposed adaptive protocol is more effective in protecting the destination’s buffer from being congested by heavy traffic load, and thus increases the chances of meeting the QoS requirements for the real-time data streams. The previous results show the enhancement of the network’s utilization that leads to preserve the overall performance of the network.

Figure 6.16: Average Buffer Consumption (Weighted Multi-layer $\psi_2$).

In the case of a fully consumed destination’s buffer, our system was designed to allow new arrived data packets to wait until the destination processes queued data packets, and releases its memory resources. In our simulations, we compare the
efficiency of our proposed scheme over the static IPsec protocol by measuring the average number of pending packets at a fully consumed destination’s buffer. The comparison process was carried out at the steady state security level for the cases of initial buffer length: $\lambda_f/25$ and $\lambda_f/8$. Tables: 6.1, 6.2, 6.3, and 6.4 show simulation results for the single-layer confidentiality, single-layer integrity, single-layer authentication, and weighted multi-layer security design models respectively.

Table 6.1: Average Pending Packets at Destination for Confidentiality.

<table>
<thead>
<tr>
<th>Pending Packets</th>
<th>Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Single-Confidentiality) (((\lambda f / 25) Buffers)</td>
<td>1</td>
</tr>
<tr>
<td>(IPsec-Confidentiality) (Level 5) (((\lambda f / 25) Buffers)</td>
<td>4</td>
</tr>
<tr>
<td>(Single-Confidentiality) (((\lambda f / 8) Buffers)</td>
<td>2</td>
</tr>
<tr>
<td>(IPsec-Confidentiality) (Level 6) (((\lambda f / 8) Buffers)</td>
<td>6</td>
</tr>
</tbody>
</table>

Table 6.2: Average Pending Packets at Destination for Integrity.

<table>
<thead>
<tr>
<th>Pending Packets</th>
<th>Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Single-Integrity) (((\lambda f / 25) Buffers)</td>
<td>2</td>
</tr>
<tr>
<td>(IPsec-Integrity) (Level 3) (((\lambda f / 25) Buffers)</td>
<td>5</td>
</tr>
<tr>
<td>(Single-Integrity) (((\lambda f / 8) Buffers)</td>
<td>3</td>
</tr>
<tr>
<td>(IPsec-Integrity) (Level 5) (((\lambda f / 8) Buffers)</td>
<td>9</td>
</tr>
</tbody>
</table>

As we can see from the results, our adaptive scheme reduces the average number of pending packets at the fully consumed destination’s buffer without affecting the provided security level. Such reduction decreases the network congestion chances and increases the chances of meeting the QoS requirements for
real-time data classes, which totally leads to preserve the overall performance of the network.

<table>
<thead>
<tr>
<th>Pending Packets</th>
<th>Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Single-Authentication) ((λf / 25) Buffers)</td>
<td>5</td>
</tr>
<tr>
<td>(IPsec-Authentication) (Level 2) ((λf / 25) Buffers)</td>
<td>7</td>
</tr>
<tr>
<td>(Single-Authentication) ((λf / 8) Buffers)</td>
<td>10</td>
</tr>
<tr>
<td>(IPsec-Authentication) (Level 3) ((λf / 8) Buffers)</td>
<td>12</td>
</tr>
</tbody>
</table>

Table 6.4: Average Pending Packets at Destination (Weighted Multi-layer).

<table>
<thead>
<tr>
<th>Pending Packets</th>
<th>Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Weighted Multi-layer) ((λf / 25) Buffers)</td>
<td>7</td>
</tr>
<tr>
<td>(Weighted Multi-layer) (IPsec) {Levels (C:5, I:3, A:1) or (C:4, I:3, A:2)} ((λf / 25) Buffers)</td>
<td>12</td>
</tr>
<tr>
<td>(Weighted Multi-layer) ((λf / 8) Buffers)</td>
<td>8</td>
</tr>
<tr>
<td>(Weighted Multi-layer) (IPsec) {Levels (C:6, I:4, A:2) or (C:5, I:4, A:3)} ((λf / 8) Buffers)</td>
<td>13</td>
</tr>
</tbody>
</table>

6.5 Network Performance from Edge Router Agents’ Perspectives

Different real-time scheduling algorithms could be used to provide the required service for different classes of real-time data flows in a packet switched network. In this section, we demonstrate the efficiency of using the differentiated earliest deadline first (Diff-EDF) scheduling algorithm at the scheduler agent over the
two well-known scheduling algorithms: earliest deadline first (EDF) and the first come first serve (FCFS) algorithm.

According to the implementation of our agent-based system, changing the scheduling algorithm will only affect the functionality of the real-time queue agent, where queuing and fetching transactions are algorithm dependent. According to the FCFS, real-time data packets will be queued and fetched based on their arrival time, where the packet that arrives first to the queue will be retrieved first. The EDF algorithm is similar to the Diff-EDF algorithm in that both of them are priority based scheduling algorithms. The only difference between such algorithms is in the priority key, where the EDF algorithm considers the relative deadline as the queuing key rather than the effective deadline.

In order to examine the efficiency of our scheduling algorithm, we have studied the effect of the scheduling algorithm on the performance of two real-time edge router’s agents: the server and the queue agents. The efficiency was measured by considering two main QoS metrics: the miss rate at the server agent and the average total packet delays at the queue agent. Simulation results for the two metrics are shown in Fig. 6.17 and Fig. 6.18 respectively.
Figure 6.17: Miss Ratio Metric at Server Agent.

Figure 6.18: Average Total Packet Delays Metric at Queue Agent.
Since the previous metrics were measured at the edge router agent, and because we are using the same traffic characteristics for both single-layer and weighted multi-layer security design models, the obtained simulation results are design-model independent. As we can see from Fig. 6.17, applying Diff-EDF algorithm minimizes the number of dropped packets at the server agent, and thus increases the chances to meet the QoS requirements of the real-time data flows.

Fig. 6.18 shows the efficiency of using the Diff-EDF algorithm over the EDF and FCFS algorithms in protecting the edge router’s buffer from being congested by heavy traffic load; such protection was achieved by minimizing the total average delays for the queued packets that are waiting to be served by the server agent. According to the previous simulation results, the real-time Diff-EDF scheduling algorithm has been chosen for such time-critical video/audio packet switched network.

6.6 Adaptive Security-aware Scheduler Vs Feedback IPsec

According to the implemented feedback-IPsec protocol, a feedback from the edge network to the IPsec protocol was implemented, where a notification with the new security levels to be adopted will be sent. In this section, we demonstrate the efficiency of our proposed adaptive security-aware scheduling algorithm over the implemented feedback IPsec protocol. In order to evaluate such efficiency through simulation, we measure the number of security level changes of the feedback-IPsec protocol that are required to reach the steady state security level.
The simulations were carried out at different values of both initial available destination’s buffer and the negotiated time interval \((T)\). Fig. 6.19, Fig. 6.20, Fig. 6.21, Fig. 6.22, and Fig. 6.23 show such feedback-IPsec security level changes for single-layer confidentiality, single-layer integrity, single-layer authentication, and the two weighted multi-layer \((\psi_1, \psi_2)\) security design models respectively.

![Figure 6.19: Feedback IPsec Security Level Changes for Confidentiality.](image)
Figure 6.20: Feedback IPsec Security Level Changes for Integrity.

Figure 6.21: Feedback IPsec Security Level Changes for Authentication.
Figure 6.22: Feedback IPsec Security Level Changes (Weighted Multi-layer $\psi_1$).

Figure 6.23: Feedback IPsec Security Level Changes (Weighted Multi-layer $\psi_2$).
According to the simulation results and for each security level change, the feedback-IPsec protocol repeats the pre-security association phase (SA) between the data packet generator and the end user, which means adding additional overhead to the system ($D_{sec}$) as shown in equations: 5.1 and 5.2. The figures show that the number of security switches will be less for both higher number of initial available buffers and initial negotiated time interval ($T$), where the system reaches its steady state average security level faster than it for lower values of initial available buffers and time interval ($T$).

Our proposed scheme eliminates the repeated security association phase performed by the feedback-IPsec, and thus less overhead is added to the system. Such elimination increases the chances to meet the QoS requirements for different classes of data flows in terms of both miss rate and average total delays; it also preserves the overall performance of the network by protecting it from being congested by heavy traffic load.
CHAPTER 7
CONCLUSIONS AND FUTURE WORK

7.1 Conclusions

In the area of real-time networks and data communication, a huge amount of research has been performed to provide different levels of service guarantees to the real-time network applications. In this research, we propose an adaptive security-aware scheduling with congestion control mechanism for packet switching networks using real-time agent-based systems. The proposed system combines the functionality of real-time scheduling with the security service enhancement, where the real-time scheduling unit uses the differentiated-earliest-deadline-first (Diff-EDF) scheduler, while the security service enhancement scheme adopts a congestion control mechanism based on a resource estimation methodology.

The security service enhancement unit was designed based on two models: single-layer and weighted multi-layer design models. For single-layer, the design provides an enhancement for a single security service: confidentiality, integrity, or authentication, while the weighted multi-layer design provides an enhancement for multiple security services with different weights. The proposed system provides the required QoS guarantees for different classes of real-time data flows (video, audio), while adaptively enhances the packet’s security service levels according to a feedback from the congestion control model, which efficiently utilizes the buffering system at the edge network, and thus protects the network from being congested by heavy traffic load.
Our agent-based system eliminates the overhead of the security association phase performed by the internet protocol security (IPsec). Such elimination had been achieved by overloading the priority code point (PCP) fields of the IEEE 802.1Q tagged frame format for the single-layer scheme, while repeated single-layer and overloading both the PCP & the virtual-LAN identifier (VID) fields of the IEEE 802.1Q tagged frame format fields were the adopted methodologies by the weighted multi-layer security design model.

Simulation results prove that using the differentiated-earliest-deadline-first (Diff-EDF) scheduler minimizes the flows miss rates and the flows average total delays compared to the earliest-deadline-first (EDF) and the first-come-first-served (FCFS) schedulers. From the other hand, simulation results show that our adaptive security enhancement scheme minimizes the buffer consumption, the average total packet delays, and the pending packets at the end users compared to the IPsec protocol. Our system was also compared to an implemented feedback-IPsec, where our adaptive system eliminated the repeated security associations performed by the feed-back-IPsec; hence, less overhead and increases the chances to meet the flows QoS requirements. Moreover, the implemented feedback monitoring mechanism makes our system capable of treating any dynamics in the network’s topology, which increases the reliability of our adaptive system.
7.2 Research Contributions

Our proposed multi-agent system provides the required QoS requirements for a variety of real-time data classes; it also provides an adaptive enhancement for the security levels of the real-time data classes in a packet switched network, such that the overall performance of the network is preserved. While carrying out our proposed system, the following contributions were achieved:

1) Our proposed system implements an object-oriented agent-based architecture that combines the functionality of real-time scheduling with the security service enhancement for packet switched networks, where the real-time scheduling unit uses the differentiated-earliest-deadline-first (Diff-EDF) scheduler, while the security service enhancement scheme adopts a congestion control mechanism based on resource estimation methodology.

2) The security service enhancement unit was designed based on two models: single-layer and weighted multi-layer design models. For single-layer, the design provides an enhancement for a single security service: confidentiality, integrity, or authentication, while the weighted multi-layer design provides an enhancement for multiple security services with different weights.

3) The proposed system provides the required QoS guarantees for different classes of real-time data flows (video, audio), while adaptively enhances the packet’s security service levels according to a feedback from the control congestion model, which efficiently utilizes the buffering system at the edge network, and thus protects the network from being congested by heavy traffic load.
4) Our agent-based system eliminates the overhead of the security association phase performed by the IPsec protocol. Such elimination had been achieved by overloading the priority code point (PCP) fields of the IEEE 802.1Q tagged frame format for the single-layer scheme, while repeated single-layer and overloading both PCP and VID fields of the IEEE 802.1Q tagged frame format fields were the adopted methodologies by the weighted multi-layer security design model.

7.3 Future Work

According to the proposed work, there are variety of characteristics and certain concerns that could be solid bases for significant and relevant future work; such concerns include:

1- According to our proposed system, the packet’s security enhancement process is performed by the source agent according to a notification from the coordinator agent. Such approach could be modified by performing the packet’s security enhancement process at the edge router, where most of the hacking processes occur at that level of the network. Such modification reduces the control messages between coordinator and source agents; it also provides direct live responses to any dynamics in the network’s status.

2- A linear network coding unit could be implemented as a part of the security enhancement unit at the edge router. Such design provides the real-time data flows with an additional layer of security, which doesn’t affect the packet’s QoS requirements, where the linear network coding
process is a machine bit-wise operation; accordingly, this operation can be used to attain the maximum possible throughput in the network. The system design requires providing the server agent with a specific buffer for each end station, where the network coding process depends on combining several real-time packets together before transmission. One of the network coding packet’s combination processes is the logical bit-wise xor operation. According to the xor operation, the size of the generated coded packet will be equal to the maximum size of the combined packets. From the other hand, the destination agent performs a linear decoding process to extract the original transmitted real-time data packets.

3- The agent based system could be modified to serve both real-time (video, audio) and non real-time (text) data flows; such modification will be in terms of applying a hierarchal scheduling algorithm at the scheduler agent, where two scheduling algorithms are used; one for each class of data flows (real-time and non real-time). This design could also modify the structure of the queuing system through implementing an individual queue for each data class. The coordinator agent will be implemented to make the required transition between queues, such that QoS requirements will be achieved for all data classes.

4- Our edge-router agents could be redesigned to be installed on a wireless router to provide the required security requirements in a wireless mesh network (WMN). According to the designed agent-based system, the
scheduler agent should be capable of identifying the real-time data sources that have the ability to transmit; it also identifies their corresponding transmission power levels and rates. The security enhancement unit should keep into consideration the requested QoS requirements by the source agent, where the mobility of end stations affects such QoS requirements in terms of total average packets’ delays; accordingly, such mobility plays a key role in the packet’s security enhancement process.

5- A software agent-based system for multiprocessor edge router could be designed to provide the required QoS and security requirements for real-time data packets in a parallel forwarding system. In such design, the coordinator agent provides an adaptive load balancing technique that achieves the optimal resource utilization, maximizes the flow of data in the network (throughput), minimizes the packet’s waiting time at the queue agent (minimizes response time), and protects the network from being congested by heavy traffic load.
BIBLIOGRAPHY


