Real-Time Scheduling With Security Awareness for Packet Switched Networks

Maen Saleh # and Liang Dong *

#Electrical & Computer Engineering, Western Michigan University, Kalamazoo MI 49008, USA
*Electrical & Computer Engineering, Baylor University, Waco TX 76798, USA

***Abstract*— Conventional real-time scheduling algorithms provide timing constraints to network applications without enhancing or optimizing the security service according to the network dynamics. In this paper, we propose a GAIA multi-agent system that combines the functionality of real-time scheduling with the confidentiality security service enhancement for packet switched networks. The real-time scheduling uses the differentiated earliest deadline first scheduler. The security service enhancement scheme adopts a resource estimation methodology. The proposed system provides the required quality-of-service for the real-time data traffic while adaptively enhancing the packet's confidentiality security service level. The buffering systems at the edge router and the destination nodes are optimally used to protect the network from being congested by heavy traffic load.**

I. Introduction

The requirement of the quality-of-service (QoS) for real-time network applications keeps increasing. QoS metrics are in various forms such as bandwidth, latency, miss ratio, mean time between failures, mean time to restore service, and any combination of these metrics. The QoS can be guaranteed by implementing appropriate real-time scheduling algorithms. In order to provide security services to their real-time data streams, most sources use existing security protocols such as the secure socket layer (SSL) protocol, the transport layer security (TLS) protocol, and the Internet protocol security (IPSec). These protocols make the packets robust against security threats especially in the local-area network, where hacking activities occur at the network edge [1], [2]. With the current security protocols, any dynamic change in the network can not affect the predetermined security level. Therefore, the network performance metrics are not taken into account and the QoS may not be guaranteed for different types of real-time data streams [3], [4]. For real-time applications, a balance is needed between security requirement and overall network performance.

In this paper, we propose a GAIA multi-agent system [5] that provides security-aware scheduling for real-time video/audio packet switched networks. A heterogeneous environment such as the real-time network can be effectively designed using a multi-agent system, where multiple entities interact with a well-define protocol and the required timing constraints are enforced on the actions performed by the communicating entities [6]. The proposed scheme uses a real-time monitoring technique and adopts a buffer estimation method for the end nodes. It provides a real-time network congestion control while adjusting the data flows to strong confidentiality security levels. The overall network performance is preserved and the QoS can be guaranteed for real-time applications.

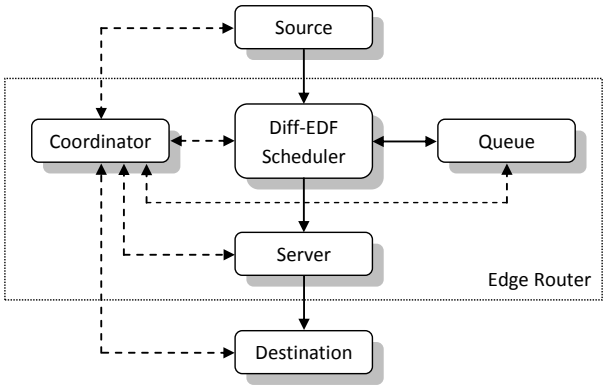

Fig. 1. GAIA multi-agent system model. Solid line: data transfer; Dashed line: interaction.

II. Design Model

The network system is designed using the real-time GAIA multi-agent system model. According to the agent-oriented GAIA methodology, the system is designed based on three main phases: decomposition, modeling, and communication protocol. The system is decomposed into six interactive entities: source, destination, differentiated earliest deadline first (Diff-EDF) scheduler, buffer queue, server, and coordinator. The last four entities are implemented at the edge router. We model each entity as an active agent by specifying its main functionalities, behaviors, and interaction schemes with others. The communication protocol describes the actions to be taken by an agent, upon receiving a trigger from another communicating agent. These actions include updating internal data, changing current status, and performing a specific task. Fig. 1 shows the GAIA multi-agent system model.

Source Agent: The source agent is a real-time data packet generator. It generates one of the two types of

real-time data packets: video or audio. Each packet has a fixed size $P_s = 1.46$ KB (1500 bytes), which is the maximum size of the Ethernet packet frame. The source agent sends real-time traffic f with a rate of λ_f. An exponential distribution with mean $1/\lambda_f$ is used to for the packet inter-arrival time. Another exponential distribution with mean $1/\mu_f$ is used for the packet service time, where μ_f is the service rate $\mu_f = 8B_w/P_s$, and B_w is the average aggregate bandwidth needed for both types of real-time traffic (video and audio). A uniform distribution is used to generate the relative deadline D_f associated with real-time traffic f. A QoS requirement is specified for traffic f in terms of deadline miss rate Φ_f.

In order to combat unauthorized access of the data packets (snooping threat) in the LAN environment, the source agent uses cryptographic security algorithms. A confidentiality security service level, ranging from 0 to 7, is applied to each real-time packet which indicates one of the eight cryptographic algorithms used by the source agent. Index 0 indicates the weakest cryptographic algorithm and index 7 indicates the strongest cryptographic algorithm. Table I shows these security algorithms, which are based on a study performed on 175 MHz processor machine [7]. In the table, μ_j is the data rate in KB/ms that can be enhanced using the jth cryptographic security algorithm, and S_j is a number between 0.08 and 1, which indicates the efficiency of the security algorithm with respect to the strongest algorithm $S_j = 13.5/\mu_j$. The source agent interacts with the coordinator agent by sending requests to serve its real-time traffic with QoS requirements.

Coordinator Agent: The coordinator agent is a software agent. It communicates with other agents to regulate their functionalities. The coordinator does not have a global view of the entire system. However, because it locates at the edge router (the default gateway of the LAN), the coordinator is capable of interacting with the source with known IP address and the destination with known MAC address. The coordinator interacts with the scheduler, queue, and server to deliver the packets. It also monitors the system behavior and decides when and how to inform other agents to change their behaviors.

Diff-EDF Scheduler Agent: The Diff-EDF scheduler agent enforces the timing constraints on the packets to provide QoS requested by the source. The Diff-EDF is one of the real-time priority scheduling algorithms that are based on the EDF scheduling algorithm. Instead of using the relative deadline as the priority key, the Diff-DEF uses the effective deadline D_{ef} as the priority key. The Diff-EDF scheduler applies a shadow function to the packet relative deadline by adding a parameter C_f and generates the effective deadline. In doing so, higher priorities are assigned to video flows over audio flows, and among the video flows, higher priorities are assigned to the stream with smaller deadline miss rate Φ_f.

Index (j)	Algorithm	S_j	μ_j (KB/ms)
0	SEAL	0.08	168.75
1	RC4	0.14	96.43
2	Blowfish	0.36	37.5
3	Knufu/Khafre	0.40	33.75
4	RC5	0.46	29.35
5	Rijndael	0.64	21.09
6	DES	0.90	15
7	IDEA	1.00	13.5

TABLE I
CRYPTOGRAPHIC ALGORITHMS

Server Agent: The server agent is responsible of serving the real-time data packets that are chosen 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. The server keeps track of a QoS parameter, the miss ratio, and reports it to the coordinator. The coordinator then adjusts system parameters accordingly in order to meet the QoS requirements.

Buffer Queue Agent: The buffer queue agent has two processes: the queuing (storing) process and the dequeuing (fetching) process. In the queuing process, the queue agent places the arriving packets in its buffer according to their effective deadlines. This process is in response to a request from the scheduler. In the dequeuing 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. There is feedback from the queue agent to the coordinator, through which the queue notifies the coordinator of its buffer usage. If the buffer usage exceeds a limit, the queue agent sends a feedback to the coordinator. In response, the coordinator adjusts system parameters to avoid dropping the real-time packets.

Destination Agent: The destination agent performs a first-come first-served (FCFS) scheduling algorithm on the receiving packets from the server. It sends two parameters to the coordinator. The first is its processing rate P_f of traffic flow f which is sent to the coordinator at the initiation phase. The second is the size B_f of its available buffers for accommodating packets of traffic flow f from the server. At every time period T that is specified by the coordinator, the destination agent sends B to be used in the process that determines the packet's confidentiality security service level.

III. SECURE DIFF-EDF MULTI-AGENT SYSTEM

Due to its adaptive feature, the proposed real-time scheduling with security awareness cannot be modeled based on the conventional static queuing theory. Upon receiving a request from the source, the coordinator models

the scheduling process as a general Brownian motion with negative motion drift parameter ($-\theta$) [8], such that

$$\theta = \frac{2(1-I)}{\sum_{f=1}^{N} \lambda_f (I_f^2 \sigma_{1f}^2 + \sigma_{2f}^2)} \quad (1)$$

where σ_{1f} is the standard deviation of the inter-arrival time for flow f, σ_{2f} is the standard deviation of the service time for flow f. The intensity of traffic flow f is $I_f = \lambda_f/\mu_f$, the total intensity of all flows is $I = \sum_{f=1}^{N} I_f$, and N is the number of flows. With the smallest deadline miss rate of all flows being Φ_{min}, the coordinator obtains the parameter C_f as

$$C_f = \theta^{-1} \log(\Phi_f/\Phi_{\text{min}}) \quad (2)$$

where Φ_f is the required deadline miss rate of traffic f. The coordinator evaluates the feasibility of serving such request by estimating the flow deadline miss rate

$$\hat{\Phi}_f = \exp(-\theta(D_{\text{avg}} - C_f)) \quad (3)$$

where D_{avg} is the average effective deadline for all data flows

$$D_{\text{avg}} = \sum_{f=1}^{N} I_f(\Phi_f + C_f). \quad (4)$$

If the estimated deadline miss rate meets the QoS requirement, i.e. $\hat{\Phi}_f < \Phi_f$, the coordinator performs the following actions: (1) Sending an acceptance message to the source; (2) Passing the parameter C_f to the Diff-EDF scheduler; and (3) Passing the required deadline miss rate Φ_f to the server. The source agent responds to the acceptance message by sending its real-time data packets to the scheduler. The scheduler performs a shadow function to obtain the effective deadline of the arrived packet

$$D_{ef} = D_f + C_f. \quad (5)$$

The scheduler then forwards the packet to the queue agent, which queues the packet based on its effective deadline.

The functionality of the server agent is to complete the process of serving the real-time packets. Once the sever agent completes serving a current real-time packet, it interacts with the coordinator by sending an idle status message. The coordinator responds by interacting with the queue agent to perform a dequeing process. The queue agent fetches a packet that is closest to expire (with smallest D_{ef}) and forwards it to the scheduler. The scheduler passes the packet to the server. Once it receives the packet, the server agent performs the following: (1) Changing its current status to busy; (2) Serving the unexpired packet (deadline is not exceeded) or dropping the expired packet; (3) Forwarding the packet to its destination according to its MAC address; and (4) Keeping track of two counters: n_f the number of packets served for the destination of traffic flow f and t_f the sum of time differences of served packets for the destination of traffic f, i.e. $t_f = \sum_{i=1}^{n_f}(t_i - t_{i-1})$.

Over every time period T, the coordinator interacts with the server and the destination requesting for the server's counter information (n_f, t_f) and the destination's resource information (B_f, P_f). Upon receiving such information, the coordinator finds the mean inter-arrival time $(1/\zeta_f)$ for the packets of traffic flow f delivered to their destination

$$1/\zeta_f = t_f/(n_f - 1). \quad (6)$$

In its data base, the coordinator stores the cryptographic information (Table I). Based on this, the coordinator determines the length of buffer L_{fj} that is needed to enhance n_f real-time packets with the jth cryptographic algorithm

$$L_{fj} = \mathcal{P}_f \zeta_f \quad (7)$$

where $\mathcal{P}_f$ is the total processing time of the packets of traffic flow f. It takes into account two delays: the delay of solving the problem of two or more equally prioritized packets ($D_{f,\text{equal\_priority}}$) and the delay of the preemption process when an arriving packet is closer to expiration than the remaining time of the currently processing packet ($D_{f,\text{preemption}}$). We have

$$\mathcal{P}_f = D_{f,\text{equal\_priority}} + D_{f,\text{preemption}} + \tau_{fj} \quad (8)$$

where τ_{fj} is the time required to decrypt a packet of 1500 bytes (1.46 KB) using the jth security algorithm $\tau_{fj} = 1.46 \text{ KB}/(\mu_j \beta)$, and β is the processing rate factor $\beta = P_f/175$ MHz.

The coordinator compares the length of available buffers at the destination with the required length of buffers to enhance n_f packets using different cryptographic algorithms. It adopts the strongest security algorithm with no congestion in 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 packets to security level z is L_{fz}, the coordinator enhances/reduces security to level z, or stays at the same security level z ($z = 0, 1, \ldots, 7$) such that

$$L_{fz} \leq B_f < L_{f(z+1)}. \quad (9)$$

Suppose that $L_{f8} = \infty$. The overall network performance is preserved as the buffers are well utilized. Once the decision on security level is made, the coordinator notifies the source. No notification will be sent if the decision is to stay at the same security level. Upon receiving a notification, the source agent applies corresponding new security enhancement algorithm to the packets to be sent.

There is also a feedback mechanism from the server agent to the coordinator agent. When the server notices a miss rate near the required miss rate limit Φ_f, it interacts with the coordinator. The coordinator notifies the source to adjust its parameters such as increasing the deadline miss rate limit or decreasing the sending rate. Accordingly, the coordinator adjusts the system parameters and notifies the scheduler and the server with new C_f and Φ_f, respectively.

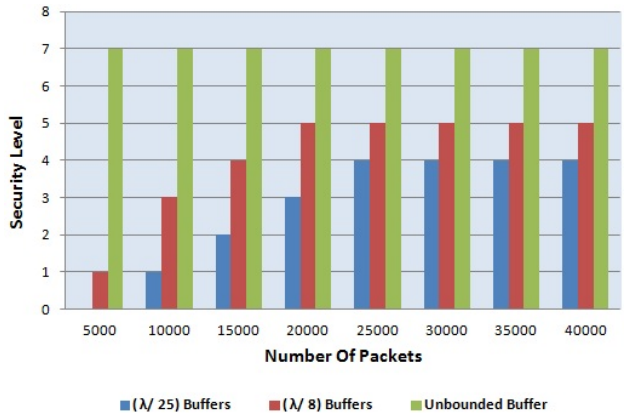

Fig. 2. Buffer effect on security level enhancement.

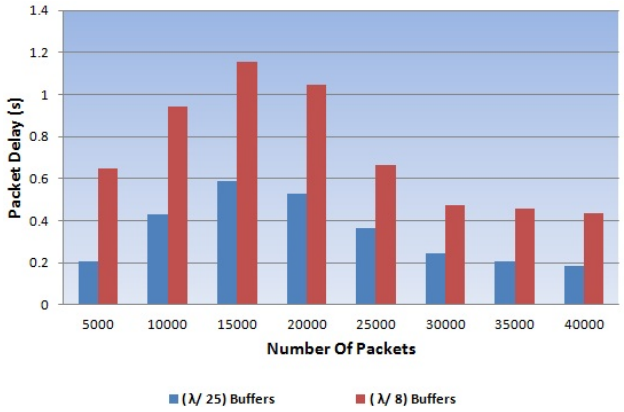

Fig. 3. Buffer effect on average packet delays.

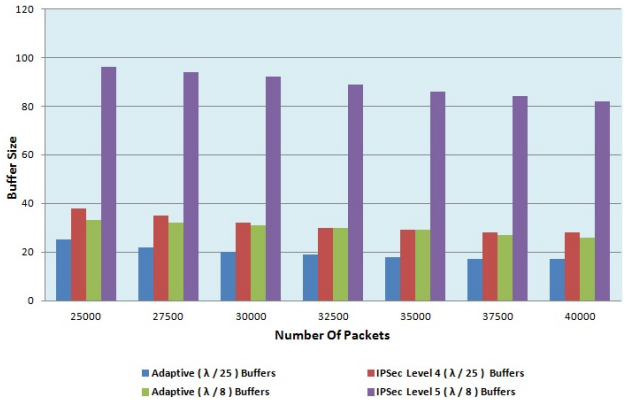

Fig. 4. Average buffer consumption at destination.

IV. Numerical Results

We simulate a network with N pairs of distinct source and destination, $N = 2, 4, \ldots, 32$. Among the N streams of traffic, there are $N/2$ real-time video streams and $N/2$ real-time audio streams. Each source has a sending rate of $\lambda_f = 1250$ packets per second and starts sending packets with the lowest security level. The edge router will notify each source to update to an appropriate security level. The length of initially available buffers at the destination is either $\lambda_f/25$, $\lambda_f/8$, or λ_f (the unbounded case), multiplied by unit time. Each destination agent has a processing rate factor $\beta = 1$, i.e. a processing speed of 175 MHz. Fig. 2 and Fig. 3 show the effects of initial buffer length at the destination agent on the confidentiality security enhancement process and the average packet delay, respectively. The confidentiality security level with larger number of available buffers is higher, since the destinations are more flexible in decrypting the packets. However, this trades off with the QoS in terms of packet delay.

The utilization of buffers at the destination agent determines the network congestion. The performance of the proposed scheme is compared with the IPSec protocol with a static security level for the cases of initial buffer length $\lambda_f/25$ and $\lambda_f/8$. The static security level is the steady state security level of the proposed adaptive algorithm (as shown in Fig. 2). Fig. 4 shows the buffer consumption at the destination. The proposed adaptive protocol is more effective in protecting the destination from being congested hence enhancing the network performance.

V. Conclusion

A multi-agent system is designed for real-time packet switched networks that aims at providing best security services without congesting the network. Effective buffer utilization is key to the enhancement of the overall network performance. An adaptive security-aware scheduling algorithm uses the buffering systems at both the edge router and the destination nodes to regulate the real-time traffic in the network. Simulation results show that the proposed scheme leads to strong security levels while preserving the network QoS.

References

[1] S. H. Son, S. Member, R. Mukkamala, and R. David, "Integrating security and real-time requirements using covert channel capacity," *IEEE Trans. Knowledge Data Eng.*, vol. 12, no. 6, pp. 865–879, Nov./Dec. 2000.

[2] E. Cody, R. Sharmana, R. H. Rao, and S. Upadhyaya, "Security in grid computing: A review and synthesis," *Decision Support Systems*, vol. 44, no. 4, pp. 749–764, Mar. 2008.

[3] T. Xie and X. Qin, "Security-aware resource allocation for real-time parallel jobs on homogeneous and heterogeneous clusters," *IEEE Trans. Parallel Distrib. Syst.*, vol. 19, no. 5, pp. 682–697, May 2008.

[4] X. Zhu and P. Lu, "A two-phase scheduling strategy for real-time applications with security requirements on heterogeneous clusters," *Comp. Elec. Eng.*, vol. 35, no. 6, pp. 980–993, Nov. 2009.

[5] F. Zambonelli, N. R. Jennings, and M. Wooldridge, "Developing multiagent systems: The Gaia methodology," *ACM Trans. Software Eng. Methodology*, vol. 12, no. 3, pp. 317–370, July 2003.

[6] W. Shen, "Distributed manufacturing scheduling using intelligent agents," *IEEE Intell. Syst.*, vol. 17, no. 1, pp. 88–94, Jan./Feb. 2002.

[7] T. Xie and X. Qin, "Scheduling security-critical real-time applications on clusters," *IEEE Trans. Comput.*, vol. 55, no. 7, pp. 864–879, July 2006.

[8] H. Zhu, J. P. Lehoczky, J. P. Hansen, and R. Rajkumar, "Diff-EDF: A simple mechanism for differentiated edf service," in *Proc. IEEE Real-Time Embedded Tech. App. Symp.*, 2005, pp. 268–277.