# Performance Modeling and Analysis of Network Firewalls

Khaled Salah, *Member, IEEE,* Khalid Elbadawi, *Member, IEEE,* and Raouf Boutaba, *Fellow, IEEE*

***Abstract*—Network firewalls act as the first line of defense against unwanted and malicious traffic targeting Internet servers. Predicting the overall firewall performance is crucial to network security engineers and designers in assessing the effectiveness and resiliency of network firewalls against DDoS (Distributed Denial of Service) attacks as those commonly launched by today's Botnets. In this paper, we present an analytical queueing model based on the embedded Markov chain to study and analyze the performance of rule-based firewalls when subjected to normal traffic flows as well as DoS attack flows targeting different rule positions. We derive equations for key features and performance measures of engineering and design significance. These features and measures include throughput, packet loss, packet delay, and firewall's CPU utilization. In addition, we verify and validate our analytical model using simulation and real experimental measurements.**

***Index Terms*—Network firewalls, performance modeling, performance analysis, queueing systems.**

## I. Introduction

NETWORK firewalls act as the first line of defense in protecting network and server resources from unauthorized access and malicious attacks. Firewalls are typically deployed at the edge of the network or at the entry point of a private network. Incoming and outgoing Internet traffic is inspected by network firewalls. Based on a set of rules, firewalls can allow or block incoming or outgoing traffic. To accomplish this, network firewalls have a rule-based engine that interrogates incoming packets sequentially rule by rule until a match is found. In particular, commercial firewalls such as the popular Cisco PIX in addition to PC-based open-source network firewalls such as Linux Netfilter and FreeBSD ipfw have a huge rulebase or ACL (Access Control List) comprising a list of rules, where each rule represents a set of conditions [1]–[5]. If an incoming packet matches all conditions of a particular rule, then a certain action is taken, e.g., to pass or drop the packet. A packet can match the conditions of more than one rule. In such a case, the first rule will have priority and its action will be applied to the packet. Accordingly, the firewall checks the rules sequentially, one by one, until a rule is matched.

Manuscript received May 31, 2011; revised September 14, 2011. The associate editor coordinating the review of this paper and approving it for publication was E. Bertino.

K. Salah is with the Department of Computer Engineering, Khalifa University of Science, Technology and Research, PO Box 573, Sharjah, UAE (e-mail: khaled.salah@kustar.ac.ae).

K. Elbadawi is with the School of Computing, DePaul University, Chicago, IL, USA (e-mail: badawi@cdm.depaul.edu).

R. Boutaba is with the David R. Cheriton School of Computer Science, University of Waterloo, ON, N2L 3G1 Canada, and the Division of IT Convergence Engineering, POSTECH, Pohang, KB 790-784, Korea (e-mail: rboutaba@cs.uwaterloo.ca).

Digital Object Identifier 10.1109/TNSM.2011.122011.110151

Firewalls themselves can be subjected to malicious attacks from the Internet as they are typically deployed at the edge of the network. One of the most serious attacks is the Distributed Denial of Service (DDoS) attack. According to the 2010 Report conducted by Arbor Networks, there is a staggering and alarming 102 percent increase of DDoS attack bandwidth in 2010 when compared to 2009 [6]. The increase of this bandwidth has been attributed to the exponential growth of botnets from which such attacks originate.

In light of the above, it is becoming a design imperative to analyze the performance of network firewalls when subjected to DDoS attacks. If network firewalls are poorly designed to withstand DDoS attacks, the overall security of the protected network will be jeopardized. Specifically, there is an increasing demand for analytical models to aid firewall designers in predicting how effective and efficient is the network firewall under DDoS attacks. In addition, modeling and analyzing the performance of network firewalls can be extremely useful in gaining a deeper understanding of firewalls' behavior and characteristics. Firewall designers and system administrators can identify bottlenecks and key parameters that impact its performance, and then perform the necessary tuning for optimal performance. Analysis can provide quick answers to numerous design and operational questions. For instance, firewall designers can use analysis to carry out a first cut design to reduce the set of design alternatives and then use simulations and/or experiments to assess few good designs before building and deploying the system. In this paper, we present an analytical queueing model developed for the study and analysis of the performance of rule-based (also known as list-based) firewalls. Rule-based firewalls are the most widely deployed among other types of firewalls [7].

This paper builds on and significantly extends our preliminary work presented in [8]. The most notable extensions include a detailed analysis and mathematical derivations of a number of key performance measures namely throughput, delay, CPU utilization, and packet loss. In [8], a brief analysis and preliminary results were presented only for firewall's throughput and delay. In addition, this paper presents an easy to implement algorithm for the derivation of the state probabilities of the introduced Markov chain model. Furthermore, this paper presents and experimental verification and validation of the analytical model. Finally, this paper offers more insights into the understanding of firewalls' behavior and performance, particularly in terms of how the firewall's throughput and CPU utilization are affected under DoS attack flows of varying rates.

The rest of the paper is organized as follows. Section II discusses related work. Section III presents our analytical model

of a finite queueing system which represents and captures network firewall behavior and dynamics. Section IV is dedicated to the verification and validation of our analytical model and describes our experimental setup. Section V presents and compares analytical and experimental performance results of a network firewall under normal traffic flows and DoS attack flows targeting different positions in the rulebase. Finally, Section VI concludes the paper and describes future work.

## II. Related work

The literature comprises little or no work on modeling and performance analysis of network firewalls, particularly under DoS attacks. The majority of research work that exists in the literature is geared towards improving the overall firewall performance by proposing techniques to optimize and detect misconfiguration in firewall security policies as reported in [9]–[19]. In [20], two optimization approaches on using Ternary Content Addressable Memories (TCAM) chip have been presented. TCAM chip is a hardware chip dedicated for fast packet classification. Acharya, et al. in [7] developed a simulation framework to study and analyze firewall operations in order to improve its performance against dynamically changing network traffic characteristics. In [21] and [22], an experimental evaluation of firewall performance is presented using firewall analysis tools. Some work has also been done on the analysis of firewalls vulnerability to traffic-specific attacks, such as IP spoofing attacks [23]. In [24], performance metrics for vulnerabilities resulting from firewall operations are presented and analyzed. In [25], a traceroute technique was used to determine whether or not a particular packet can pass from an outside remote host to a destination host behind a firewall.

The analysis presented in this paper is based on a queueing model with multi-phase service. In the literature, multi-phase queueing systems are studied to some extent, particularly in the work presented in [26]–[29]. However, these studies only offer general guidelines on how to analyze such queueing systems, with no handling of the problem at hand, particularly with regards to solving state-transition probabilities and finding a closed-form solution for the general distribution of the service times. Such a solution is required to derive the key performance characteristics of the system. Other related work on two-phase queueing systems was presented by Krishna and Lee in [30] and then by Doshin in [31]. However this particular work considers a separate infinite queue for each service phase and with only two phases of service, which does not capture our particular system behavior where we have a finite buffer. In fact, all solutions for infinite queueing systems require that the arrival rate to be less than the service rate; otherwise, the system will be unstable. With today's Gigabit and 10 Gigabit Ethernet networks, such stability requirement is not practical as arrival rate of packets can far exceed the service rate of network servers.

Our analytical model can be used to analyze firewall performance when the firewall is subjected to normal traffic flows as well as DoS attack flows. The performance can be analyzed when launching DoS attack flows targeting top and bottom rules. Analyzing the performance of a firewall when targeting bottom rules is of a paramount importance

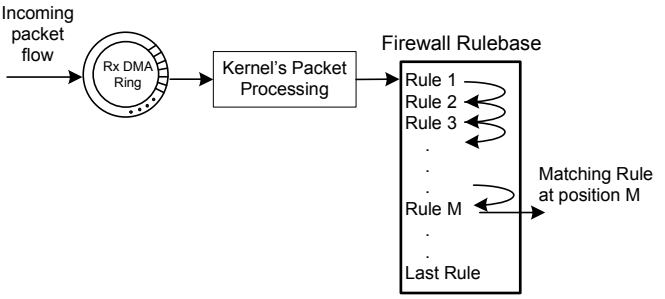


Fig. 1. Interrogation of firewall rulebase for incoming packets.

to network designers and security engineers to assess the resiliency of the firewall against worst-case DoS attacks. It was shown in [5] that bottom rules can be remotely discovered by an outside attacker. An attacker then can launch a complexity-algorithmic attack that primarily target bottom rules, and effectively degrading rapidly the performance of a firewall with a low-rate DoS attack flow. Complexity-algorithmic attacks, which have been first described in [32], are a class of low-rate DoS attacks that exploit algorithmic deficiencies in software design. The authors in [33]–[36] have shown how complexity algorithmic attacks can be mounted against network servers including Linux, Snort NDIS (Network Intrusion Detection System), and IPSs (Intrusion Prevention Systems).

## III. Analytical Model

In this section, we present a *finite* queueing model to represent the behavior and study the performance of a rule-based network firewall. Typically, and as shown in Figure 1, incoming packets carrying requests arrive at the firewall and get queued for processing in multiple stages. The first stage involves performing data-link and network layer functionalities, and subsequently the firewall rulebase search engine is activated to process incoming packets. Specifically, in Linux and FreeBSD [37]–[39], incoming packets are received by the Rx NIC (Receiving Network Interface Card) and copied using DMA (Direct Memory Access) into the Rx DMA Ring. The Rx DMA Ring is the receiving buffer and is located within the kernel memory. After successfully queueing the received packet into the Rx DMA Ring, an interrupt is generated to notify the device driver of the reception of a new packet. The device driver starts executing Data Link layer (known as Layer 2) functionalities and then invokes the kernel IP processing task. The kernel packet processing is responsible for performing IP Network layer (known as Layer 3) functionalities which include checking headers for errors, looking up routing tables, and forwarding the packet to the next destination or delivering it to user application, or in our case, to get processed or interrogated sequentially by the firewall rulebase search engine one rule at a time until a rule match occurs.

Figure 2 illustrates a finite queueing model to capture the behavior and dynamics of the system represented in Figure 1. In this model, incoming packets arrive to the firewall with an arrival rate $\lambda$. The queueing system has a buffer of $K$ packets with a queue size of $K-1$. A packet is first queued in the buffer and then served by the first stage consisting of kernel's

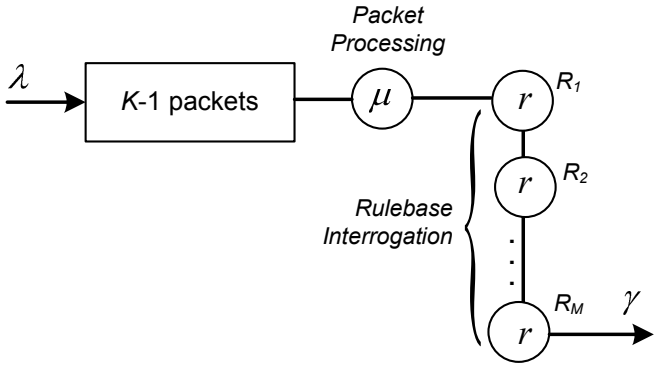


Fig. 2. Finite queueing system with multiple stages of service.

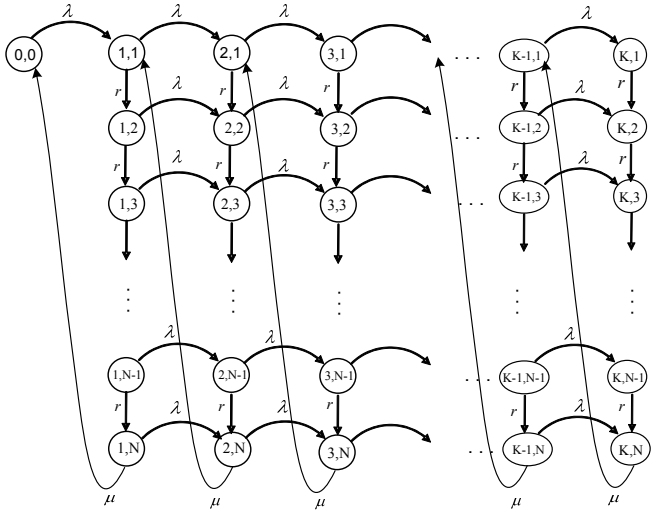


Fig. 3. State transition diagram for a rule-based network firewall with a finite buffer $K$.

packet processing with a mean service time $1/\mu$. Next, the packet is subjected to the firewall rulebase whereby each rule is interrogated sequentially until there is a matching rule at position $M$. The interrogation service time of each rule has a mean $1/r$. It is to be noted that incoming packets are served *sequentially* in $N$ stages, such that $N = 1 + M$, where $M$ is the position of the matching rule in the rulebase. A new packet enters $Stage\ 1$ of kernel's packet processing only after the previous packet has departed the queueing system, i.e. left $Stage\ N$, when rule number $M$ was triggered. The execution of all stages is mutually exclusive. That is, if one of the stages is running, no other stage will be running. This is the typical situation and the current Linux implementation in which the CPU executes one network task at a time [37]. We assume that incoming packets follow a Poisson arrival $\lambda$ and all of the service times are independent and exponentially distributed with means of $1/r$ and $1/\mu$, as shown in Figure 2. Packets are serviced according to FCFS (First Come First Served) discipline. In practice, $r > \mu$ , since the service time of packet processing involving device driver handling and network IP processing is on average much larger than that of processing individual rules.

In the following sections, we present two analytical models. The first model represents the behavior of a rule-based network firewall when all incoming packets are matched with a single rule at position $M$. The second model extends the first one to capture the behavior of a firewall when different rule positions are triggered.

### A. Model Analysis and Solution

Our finite queueing system with a multi-stage service can be represented and analyzed by an embedded Markov chain with a state space $S = \{(k,n), 0 \le k \le K,\ 0 \le n \le N\}$, where $k$ denotes the number of packets in the system and $n$ denotes the stage number that the CPU is performing. The queueing system has a queue size of $K-1$. When $n = N$, the CPU is performing packet processing, and when $n = 1 \cdots N-1$, the CPU is performing rule interrogations. A rule match and trigger will occur at stage number $N-1$ (or rule number $M$). In other words, state $(0,0)$ represents the special case when the system is empty. States $(k,n)$ represent the states where the CPU is busy handling rule $n$. States $(k,N)$ represent the states where the CPU is busy handling packet processing. The rate transition diagram is shown in Figure 3.

Let $p_{k,n}$ be the steady-state probabilities at state $(k,n)$. The steady-state balance equations are shown for each state as follows:

State $(0,0)$

$$0 = -\lambda p_{0,0} + \mu p_{1,N}$$

State $(1,N)$

$$0 = -(\lambda+\mu)p_{1,N} + r p_{1,N-1}$$

State $(1,n)$

$$0 = -(\lambda+r)p_{1,n} + r p_{1,n-1} \qquad (2 \le n \le N-1)$$

State $(1,1)$

$$0 = -(\lambda+r)p_{1,1} + \lambda p_{0,0} + \mu p_{2,N}$$

State $(k,N)$

$$0 = -(\lambda+\mu)p_{k,N} + \lambda p_{k-1,N} + r p_{k,N-1} \qquad (2 \le k \le K-1)$$

State $(k,n)$

$$0 = -(\lambda+r)p_{k,n} + \lambda p_{k-1,n} + r p_{k,n-1} \qquad (2 \le k \le K-1)\ ;\ (2 \le n \le N-1)$$

State $(k,1)$

$$0 = -(\lambda+r)p_{k,1} + \lambda p_{k-1,1} + \mu p_{k+1,N} \qquad (2 \le k \le K-1)$$

State $(K,N)$

$$0 = -\mu p_{K,N} + \lambda p_{K-1,N} + r p_{K,N-1}$$

State $(K,n)$

$$0 = -r p_{K,n} + \lambda p_{K-1,n} + r p_{K,n-1} \quad (2 \le n \le N-1)$$

State $(K,1)$

$$0 = -r p_{K,1} + \lambda p_{K-1,1}.$$

Therefore, the state probabilities of $p_{k,n}$ can be expressed recursively in terms of $p_{0,0}$ as follows:

From state $(0,0)$,

$$p_{1,N} = \left(\frac{\lambda}{\mu}\right) p_{0,0} \tag{1}$$

From state $(1,N)$,

$$p_{1,N-1} = \left(\frac{\lambda+\mu}{r}\right) p_{1,N} \tag{2}$$

From state $(1,n)$,

$$p_{1,n-1} = \left(\frac{\lambda+r}{r}\right) p_{1,n} \quad (2 \leq n \leq N-1) \tag{3}$$

From state $(1,1)$,

$$p_{2,N} = \left(\frac{\lambda+r}{\mu}\right) p_{1,1} - \left(\frac{\lambda}{\mu}\right) p_{0,0} \tag{4}$$

From state $(k,N)$,

$$p_{k,N-1} = \left(\frac{\lambda+\mu}{r}\right) p_{k,N} - \left(\frac{\lambda}{r}\right) p_{k-1,N} \quad (2 \leq k \leq K-1) \tag{5}$$

From state $(k,n)$,

$$p_{k,n-1} = \left(\frac{\lambda+r}{r}\right) p_{k,n} - \left(\frac{\lambda}{r}\right) p_{k-1,n} \quad (2 \leq k \leq K-1, 2 \leq n \leq N-1) \tag{6}$$

From state $(k,1)$,

$$p_{k+1,N} = \left(\frac{\lambda+r}{\mu}\right) p_{k,1} - \left(\frac{\lambda}{\mu}\right) p_{k-1,1} \quad (2 \leq k \leq K-1) \tag{7}$$

From state $(K,N)$,

$$p_{K,N-1} = \left(\frac{\mu}{r}\right) p_{K,N} - \left(\frac{\lambda}{r}\right) p_{K-1,N} \tag{8}$$

From state $(K,n)$,

$$p_{K,n-1} = p_{K,n} - \left(\frac{\lambda}{r}\right) p_{K-1,n} \quad (2 \leq n \leq N-1) \tag{9}$$

From state $(K,1)$,

$$p_{K,1} = \left(\frac{\lambda}{r}\right) p_{K-1,1} \tag{10}$$

It is to be noted that the state probability $p_{K,1}$ can be derived from either Equation (9) or (10), which are equivalent. The equivalency can be proved numerically.

Using the normalization condition, $p_{0,0}$ can be obtained as follows:

$$p_{0,0} + \sum_{k=1}^{K} \sum_{n=1}^{N} p_{k,n} = 1$$

Dividing both sides by $p_{0,0}$, we get

$$p_0 = p_{0,0} = \frac{1}{1 + \sum_{k=1}^{K} \sum_{n=1}^{N} \frac{p_{k,n}}{p_{0,0}}} \tag{11}$$

The above equation enables us to compute $p_{0,0}$ by first computing the terms $p_{k,n}/p_{0,0}$, which requires only $\lambda$, $\mu$ and $r$. Obtaining $p_{0,0}$ can then be used to find all other state probabilities $\{p_{k,n} : 1 \leq k \leq K, 1 \leq n \leq N\}$. Algorithm 1 shows how we can obtain recursively all state probabilities using Equations (1-10). The computation of Algorithm 1 is optimized by first computing loop invariants (as those expressions are involving $\lambda$, $r$ and $\mu$) as shown in Line 4. Then, the algorithm computes the terms $p_{k,n}/p_{0,0}$ recursively as shown in lines 5-21. In line 22, the algorithm uses Equation 11 to compute $p_0$. And in line 23, the algorithm updates the other state probabilities by multiplying the matrix $\mathcal{P}$ with the scalar value $p_0$.

**Algorithm 1** Determining all state probabilities including $p_0$

**Input:** The values of $\lambda, \mu, r, K, N$
**Output:** $p_0$ and Matrix $\mathcal{P}[1..K, 1..N]$

1: **for all** $i$ and $j$ such that $1 \leq i \leq K$ and $1 \leq j \leq N$ **do**
2: $\quad \mathcal{P}[i,j] \leftarrow 0$
3: **end for**
4: $C_1 \leftarrow \lambda/\mu$; $C_2 \leftarrow (\lambda+\mu)/r$; $C_3 \leftarrow (\lambda+r)/r$; $C_4 \leftarrow (\lambda+r)/\mu$; $C_5 \leftarrow \lambda/r$; $C_6 \leftarrow \mu/r$;
5: $\mathcal{P}[1,N] \leftarrow C_1$
6: $\mathcal{P}[1,N-1] \leftarrow C_2 \times \mathcal{P}[1,N]$
7: **for** $i = N-1$ **downto** 2 **do**
8: $\quad \mathcal{P}[1,i-1] \leftarrow C_3 \times \mathcal{P}[1,i]$
9: **end for**
10: $\mathcal{P}[2,N] \leftarrow C_4 \times \mathcal{P}[1,1] - C_1$
11: **for** $i = 2$ to $K-1$ **do**
12: $\quad \mathcal{P}[i,N-1] \leftarrow C_2 \times \mathcal{P}[i,N] - C_5 \times \mathcal{P}[i-1,N]$
13: $\quad$ **for** $j = N-1$ **downto** 2 **do**
14: $\quad\quad \mathcal{P}[i,j-1] \leftarrow C_3 \times \mathcal{P}[i,j] - C_5 \times \mathcal{P}[i-1,j]$
15: $\quad$ **end for**
16: $\quad \mathcal{P}[i+1,N] \leftarrow C_4 \times \mathcal{P}[i,1] - C_1 \times \mathcal{P}[i-1,1]$
17: **end for**
18: $\mathcal{P}[K,N-1] \leftarrow C_6 \times \mathcal{P}[K,N] - C_5 \times \mathcal{P}[K-1,N]$
19: **for** $i = N-1$ **downto** 2 **do**
20: $\quad \mathcal{P}[K,i-1] \leftarrow \mathcal{P}[K,i] - C_5 \times \mathcal{P}[K-1,i]$
21: **end for**
22: $p_0 \leftarrow 1/(1 + Sum(\mathcal{P}))$
23: $\mathcal{P} \leftarrow p_0 \times \mathcal{P}$
24: **return** $p_0$ and $\mathcal{P}$

Consequently, key features and performance measures can be derived as follows. First, the mean system throughput $\gamma$ is basically the departure rate, i.e. the rate at which packets finish successfully after being processed by *Stage N*, that is

$$\gamma = \mu \sum_{k=1}^{K} p_{k,N} \tag{12}$$

Equivalently, the mean system throughput $\gamma$ can be expressed as

$$\gamma = (1 - p_0)/\bar{X} \tag{13}$$

where $p_0$ is given in Equation (11), and $\bar{X}$ is the mean service time which is basically the sum of the mean service time of all stages, and can be expressed as

$$\bar{X} = \frac{1}{\mu} + \frac{N-1}{r} = \frac{(N-1)\mu + r}{\mu r} \tag{14}$$

The departure rate $\gamma$ can also be expressed as the effective arrival rate $\lambda'$ which is $\lambda(1 - P_{loss})$. Therefore,

$$\gamma = (1 - p_0)/\bar{X} = \lambda(1 - P_{loss}) \tag{15}$$

where $P_{loss}$ is the loss probability (or blocking) probability. $P_{loss}$ can be expressed from Equation (15) as

$$P_{loss} = 1 - \frac{1 - p_0}{\rho} = \frac{p_0 + \rho - 1}{\rho} \tag{16}$$

where $\rho = \lambda\bar{X}$ is defined as the traffic intensity or offered load. Alternatively, and equivalently, $P_{loss}$ can be expressed as the probability of being in states $(K, 1 \cdots N)$, that is

$$P_{loss} = \sum_{n=1}^{N} p_{K,n} \tag{17}$$

The mean number of packets $\bar{K}$ in the system can be expressed as

$$\bar{K} = \sum_{k=1}^{K} \sum_{n=1}^{N} k p_{k,n} \tag{18}$$

Using Little's result, the mean time spent in the system by a job succeeding in entering the queue can be expressed as

$$W = \frac{\bar{K}}{\gamma} = \frac{1}{\gamma} \sum_{k=1}^{K} \sum_{n=1}^{N} k p_{k,n} \tag{19}$$

This gives the mean time spent waiting in the queue as

$$W_q = W - \bar{X} = \frac{1}{\gamma} \sum_{k=1}^{K} \sum_{n=1}^{N} k p_{k,n} - \frac{(N-1)\mu + r}{\mu r} \tag{20}$$

A key feature of interest is the firewall's CPU utilization. It is also called the *carried load*. The CPU utilization can be expressed as follows

$$U_{util} = \gamma\bar{X}, \tag{21}$$

where $\gamma$ is expressed in Equation (12). The *carried load* is to be distinguished from the *offered load*. The *offered load* is expressed as $\lambda\bar{X}$.

### B. Multiple Flows

In a realistic situation, the firewall can be subjected to multiple flows, with each flow targeting the same or different rule. Such a situation is very common in today's DDoS attacks launched by botnets. In this section, we model and analyze the performance of a firewall when subjected to multiple flows. For simplicity, we define a flow in this paper in a loose term such that an incoming flow will always trigger one rule. If a flow triggers more than one rule, then it would be defined as multiple flows. Our analytical model described earlier for one flow can also be used to study the performance when also having multiple incoming flows with corresponding arrival rates $\{\lambda_i : 1 \leq i \leq S\}$, such that each individual flow $i$ triggers a particular rule in $\{R_j : 1 \leq j \leq L\}$ of the firewall rulebase (as illustrated in Figure 4), where $S$ denotes the total number of incoming flows and $L$ denotes the size of a firewall rulebase. It is to be noted that more than one flow can trigger the same rule $R_j$.

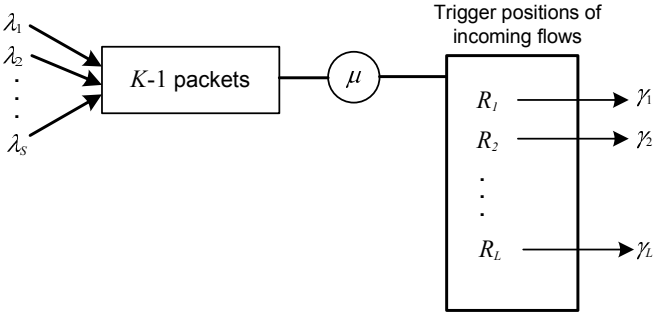


Fig. 4. Multiple incoming flows with each flow matching a different rule.

A solution in this case can be approximated by aggregating all flows into one aggregated flow and determining the average matching rule position ($\bar{M}$) for the aggregated flow. The aggregated flow rate $\hat{\lambda}$ can be expressed as

$$\hat{\lambda} = \sum_{i=1}^{S} \lambda_i \tag{22}$$

The average position $\bar{M}$ of all matching rules for the aggregated flow $\hat{\lambda}$ can then be expressed as

$$\bar{M} = \left\lceil \sum_{i=1}^{S} \left( \frac{\lambda_i}{\hat{\lambda}} \times M_i \right) \right\rceil, \tag{23}$$

where $M_i$ is the matching rule position of flow i. Note that $\bar{M}$ is the ceiling of the right-hand-side expression since $\bar{M}$ must be an integer.

We can estimate $p_0$ by applying Algorithm 1 where $\lambda$ is substituted by $\hat{\lambda}$ and $N$ is substituted by $\bar{M}+1$. The remaining input parameters $\mu$, $r$ and $K$ are the same. Equations (12-21) can then be used to compute the performance measures of the aggregated flow. For individual flows, performance measures can also be computed. For example, the individual throughput $\gamma_i$ can be expressed as

$$\gamma_i = \frac{\lambda_i}{\hat{\lambda}} \times \hat{\gamma} \tag{24}$$

where $\hat{\gamma}$ is the aggregated throughput given in Equation (12). Finding $\gamma_i$ can be used to compute other performance measures of flow $i$. For example, the CPU utilization per flow $U_{util,i}$ can be expressed using Equation (21) as

$$U_{util,i} = \gamma_i\bar{X}. \tag{25}$$

Finally, it is to be noted that the average packet delay $W_i$ is the same as average packet delay for aggregated flow $W$ , i.e. the average packet delay can be expressed using Equation (19) as

$$W = W_i = \frac{\bar{K}}{\hat{\gamma}} \tag{26}$$

The same is also true for the packet loss per flow $P_{loss,i}$ being equivalent to the packet loss of aggregated flow $P_{loss}$ given by Equation (16).

### C. Infinite Queue

A related issue of special interest is considering an infinite queue with multiple stages of service. In this case, the solution can be approximated using our analytical model with a large buffer size, e.g. $K = 10,000$. On the other hand, a more accurate solution can be found when modeling the system as an $M/G/1$ queueing system with a service time having a general distribution with a density function $b(x)$. The coefficient of variation $C_s = \sqrt{Var(x)}/\bar{X}$ can be determined, and the Pollaczek-Khinchin (P-K) formula can be used to find closed-form expressions for key features and performance measures [26]. Such closed-form expressions are presented and tabulated elegantly in [40]. The mean $\bar{X}$ is expressed in Equation 14 and the variance $Var(x)$ can be expressed as follows:

$$\bar{X} = \begin{cases} \dfrac{N-1}{r^2} + \dfrac{1}{\mu^2} & \text{if } \mu \neq r, \\ \dfrac{N}{\mu^2} & \text{if } \mu = r \end{cases}$$

### D. Limitations

Our analytical solution assumes that packet arrivals follow Poisson distribution with fixed packet size, and the services time follow Exponential distribution. For certain types of network traffic, assuming Poisson arrivals is adequate. In [41], it was concluded that modeling the voice traffic as Poisson with fixed-size packets gives adequate approximation, especially when the voice traffic is high. However, for general traffic such as Ethernet, network packets are not of fixed size, and their arrivals do not always follow a Poisson process but are rather bursty [42]–[44]. Also in reality, service times are not necessarily always exponential. An analytical solution becomes intractable and not a trivial task when considering variable-size packets and non-Poisson arrivals, and when also considering general service times. The impact of having bursty traffic and having general distribution for packet sizes and service times can be best modeled and studied using DES (Discrete Event Simulation) [40]. Despite of all of these limitations and adopted assumptions, the results obtained from our analysis were closely matching to results obtained from actual experimental measurements, as will be demonstrated in Section V.

### E. Summary of Key Performance Measures

As a quick reference, Table I lists the equation numbers for the key features and performance measures for firewall when a firewall is subjected to either a single flow or multiple flows. The performance measures include throughput, packet loss, packet delay and CPU utilization.

TABLE I
QUICK REFERENCE FOR KEY PERFORMANCE MEASURES

| Metric | Single Flow | Multiple Flows | |
|---|---|---|---|
| | | Aggregated Flows | Individual Flow |
| Throughput | (12) or (13) | (12) or (13) | (24) |
| Packet loss | (16) or (17) | (16) or (17) | (16) |
| Packet Delay | (19) | (19) | (26) |
| CPU Utilization | (21) | (21) | (25) |

## IV. VERIFICATION AND VALIDATION

To verify the correctness of our analytical models, we developed a discrete-event simulation taking into account the same assumptions as those in the analysis. The simulation followed closely the guidelines given in [45]. We used the PMMLCG as our random number generator [45]. The simulation was automated to produce independent replications with different initial seeds that were ten million apart. During the simulation run, we checked for overlapping in the random number streams and ascertained that such a condition did not exist. The simulation was terminated when achieving a precision of no more than 10% of the mean with a confidence of 95%. We employed and implemented dynamically the *replication/deletion* approach for means discussed in [45]. In such approach, only values beyond the warmup period from each simulation replication are used to estimate the mean. We computed the length of the initial transient period using the MCR (Marginal Confidence Rule) heuristic developed by White [46]. Each replication run lasts for five times of the length of the initial transient period. Simulation results for all performance metrics were very much in line with those of analysis, which imply that our analytical model is correct.

To validate our analytical model, we compare our analysis results to actual real experimental measurements reported in [5]. Figure 5 illustrates the experimental setup and testbed. In [5], reported measurements include firewall throughput, packet loss, CPU utilization, and packet delay. These measurements were taken when subjecting the firewall to two types of traffic: (1) normal traffic, and (2) DDoS traffic targeting different rules located at different positions in the firewall rulebase. The experiment comprised four modern Linux machines connected using Gigabit Ethernet links as shown in Figure 5. The four machines were all Intel Pentium 4 processors running at 3.2 GHz with 512 MB of RAM. The network cards were 3COM Broadcom NetXtreme Gigabit Ethernet with BCM5701 controller. All machines were running with Fedora Core 5 Linux 2.6.15 and with the default tg3 NIC device driver.

To generate normal traffic to pass through the firewall, we used the open-source D-ITG 2.4.4 generator [47]. To generate a single unidirectional flow, D-ITG has to be configured to send UDP traffic using the ITGSend agent to ITGRecv [47] where statistics are collected. NTP (Network Timing Protocol) protocol was used in order to synchronize timing between the ITGSend and ITGRecv machines. This was necessary to measure accurately the one-way packet delay. To generate DoS traffic, we used KUTE [48]. KUTE is an open-source kernel-level UDP traffic generator. For our measurements, the CPU utilization was measured by the sar Linux utility. However, the throughput, packet loss, and round trip time were measured by

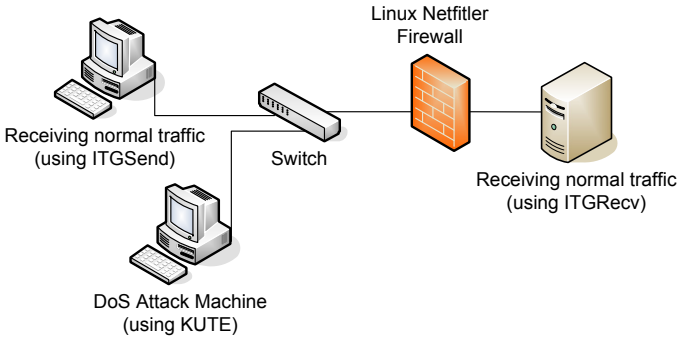


Fig. 5. Experimental setup.

D-ITG. We have used the smallest packet size of 64 bytes for generating flows by KUTE and D-ITG. Smallest packet size was used to generate the maximum traffic rate. More details on measurement and configuration setup can be found in [5].

We setup the firewall with Linux *Netfilter* and we created a ruleset using *iptables* commands that includes rules for D-ITG traffic as well as 10,000 other dummy rules. We configured the Linux *Netfilter* to accept and pass D-ITG traffic, but to drop probing packets. Rules related to D-ITG traffic or normal were positioned at the beginning of the firewall's ruleset. As discussed earlier, the firewall performance in terms of throughput, packet loss, and round trip time will be measured by the D-ITG being sent and received. The other 10,000 dummy rules were created without chains using a shell script of *iptables* command with each rule having its own same conditions of "*any*" except for the source MAC address. All of these dummy rules were encoded with UDP protocol type. We used a condition for source MAC addresses since they were experimentally found to be computationally more expensive. This is because the MAC address is one of the last conditions to be checked by *Netfilter* within a rule [3].

We measured the average processing time per rule (i.e. $1/r$) to be 0.05 $\mu$s. We measured this by instrumenting the Linux code with timestamps at the start and finish points of *Netfilter* processing in Linux kernel function `ip_local_delvier_finish` located in *ip_input.c* file as illustrated in [3]. For timestamps, we used the `rdtscl` macro which basically implements the assembly instruction of `rdtsc` (read time stamp counter) and returns the number of CPU cycles since system bootup. We took 1000 reads when subjecting the firewall to a low KUTE traffic rate $\lambda$ of 1000 pps (packets per second) for a duration of one full second. The difference between the start and finish timestamps were summed in memory in a single variable, and then the mean value was obtained by dividing this summed value by 1000. We found that the mean value for $1/r$ was approximately 0.5 ms for interrogating 10,000 rules, or 0.05 $\mu$s per rule. Similarly, we measured the average kernel's processing time of device driver and IP processing (i.e. $1/\mu$) by instrumenting the Linux code with timestamps from the point of packet reception in the device driver (at start of function `tg3_interrupt()` in *tg3.c* file) until the start point of delivery to *Netfilter* processing (in `ip_local_delvier_finish()` in *ip_input.c* file) We found the mean value for $1/\mu$ to be approximately 2.65 $\mu$s. Finally, the default maximum for both the *Tx and Rx DMA* Rings are set to 512 packets, according to the header definition in */net/drivers/tg3.h*

## V. Results and Discussion

In this section, we report experimental and analysis results of the firewall performance in terms of various key measures which include throughput, packet loss, firewall's CPU utilization, and packet delay. In particular, we report results of these key performance measures when sending a normal traffic and when subjecting the firewall to DoS traffic targeting different rules. In addition, we report analytical results and offer interpretation in order to gain a deeper insight in the firewall dynamics and behavior. For clarity reasons, results of simulation were not reported in the figures as they were closely matching those of analysis curves. For all of experimental results reported and shown in this section, we performed three experimental trials and final results are the average of these three trials. For each trial, we recorded the results after the generation of a flow with a specific rate for a sufficient duration of 30 seconds.

Figure 6 shows the performance impact on firewall performance when launching DoS attacks with different rates targeting different rule positions. The impact was measured by having ITGSend generates a normal constant UDP traffic flow at a rate of 10 Kpps. We setup the first rule in the ruleset of the Linux Netfilter to pass such traffic. We measured the performance degradation in terms of packet loss, throughput, CPU utilization, and one-way delay when sending ITG normal traffic and when subjecting the firewall to DoS attack flows of different rates and targeting different rules. We set the DoS attack flows to target firewall rules at position 1000, 5000, and 10000. In practice, DoS attack flow that targets rules positioned at 5,000 and 10,000 can represent complexity-algorithmic DoS attacks that target last-matching rules, whereas DoS attack flow that targets rules positioned at 1000 can represent traditional DoS attacks.

Figure 6 exhibits analysis curves and actual experimental measurements of the performance of the 10 Kpps normal ITG flow. The results from analysis and experimental measurements are closely matching when it comes to throughput, packet loss, and CPU utilization. However, the results obtained from experiments for one-way packet delay are shown to take the same shape of the curves but relatively higher than those of analysis, as exhibited in Figure 6(d). The reason for the extra delay obtained from experiments is due to the nature of the experiment setup whereby the delay is measured at the sender machine, and includes more than the delay encountered at the Linux Netfilter firewall machine. The experimental delay includes the delay from sending the packet by ITGSend at the sender machine, the delay at the firewall, the delay at the receiver machine of receiving packets by ITGRecv, in addition to transmission and queueing delays at the switch and links.

For CPU utilization, it is observed from Figure 6(c) that the analysis curves of the firewall CPU utilization are closely matching those of experimental measurements when DoS attack flows target rules positioned at 5,000 and 10,000. However, when targeting rules positioned at 1000, the CPU utilization analysis curve is observed to be slightly higher

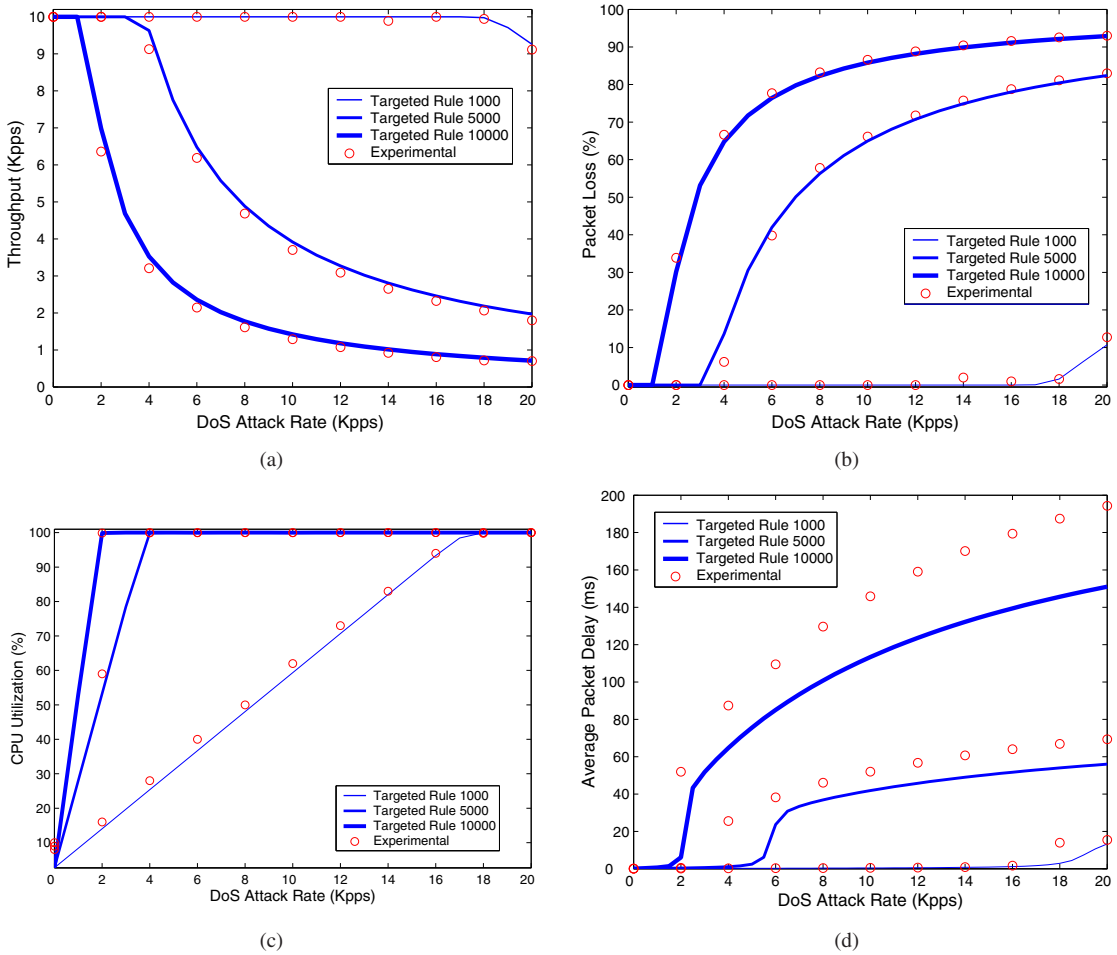


Fig. 6. The impact of DOS attack flows targeting different rules on firewall.

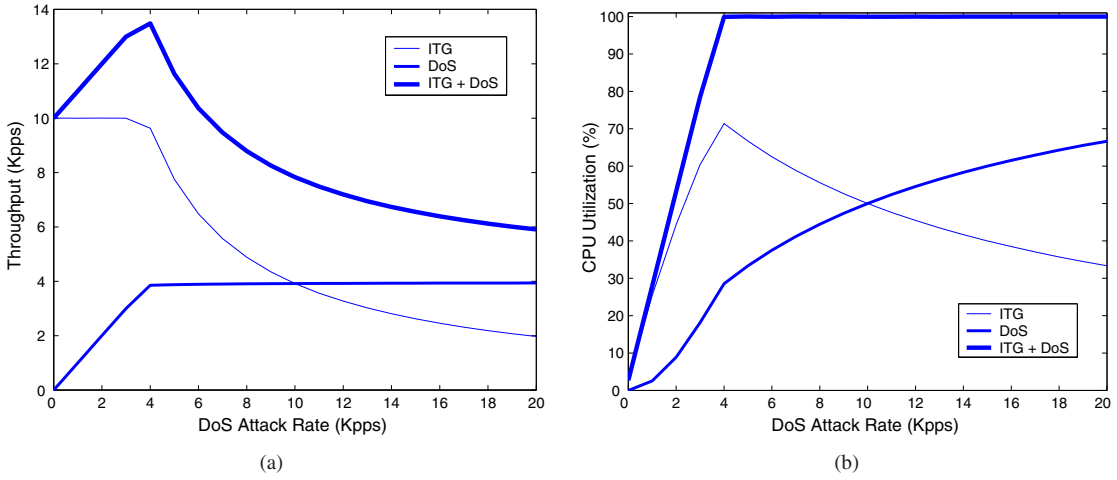


Fig. 7. Firewall's throughput and CPU utilization with respect to DOS attack rates targeting rule 5000.

than those of experimental results. The reason of this can be attributed to the fact that targeting rules at 1000 does not impose heavy processing requirement, leaving CPU processing power to be consumed by other lower-priority user processes or system tasks, thereby increasing slightly the reading of CPU utilization. However, when high-priority kernel processing consumes most of the CPU power (as is the case when targeting 5,000 and 10,000 rules), other low-priority tasks and processes do not run. In other words, at high rates, the CPU is primarily dedicated to the processing of firewall rules. This is supported by the fact when the DoS rate becomes higher (i.e. beyond 10 Kpps), the CPU utilization analytical curve of targeted rule 1000 starts matching closely those of experimental measurements.

Figure 6(a) exhibits the degree of throughput degradation of normal traffic launched by ITG when the firewall is subjected to DoS attacks launched by KUTE. Figures 6(b), 6(c) and 6(d) show the corresponding packet loss, CPU utilization, and one-way packet loss. It is clear from these figures that a slight degradation is exhibited when DoS attacks are targeting top rules, whereas significant degradation is exhibited when DoS attacks target bottom rules such as those positioned at 5,000 and 10,000. More specifically, when targeting bottom rules, severe and noticeable degradation can be observed with relatively low-rate DoS attacks of around 1 Kpps and 3 Kpps when targeting rules positioned at 10,000 and 5,000, respectively. However, when targeting the top rule positioned at 1000, degradation is exhibited only at high-rate DoS attacks of around 18 Kpps. Therefore, it can be concluded that targeting rules at the bottom of the ruleset can be severely detrimental to the performance of the firewall. The firewall performance is acceptable when DoS attacks (of up to a rate of 18 Kpps) targeting rules positioned around 1,000, but not rules greater than that.

To gain a deeper insight into understanding the firewall's behavior and performance, we plot the firewall's throughput and CPU utilization in relation to the received DoS attack flow targeting rule number 5,000. Figure 7 illustrates curves obtained from analysis for the throughput and CPU utilization with two flows: (1) the normal ITG flow with a constant rate 10 Kpps, and (2) the DoS attack flow with a rate ranging from 0 to 20 Kpps. The figure shows the throughput and CPU utilization for individual flows of ITG and DoS as well as the aggregated flow of the sum of ITG and DoS. We measured the throughput of these two flows at the firewall, and measurements were closely matching, as was the case in Figure 6(a). The CPU utilization for individual flows could not be measured because sar Linux utility is set up to measure only the overall CPU utilization of the firewall.

As shown from Figure 7, the firewall reaches a saturation point at approximately 4 Kpps. At this particular point, it is observed from Figure 7(a) that the throughput of DoS attack rate of KUTE start to flatten off, and also the normal ITG traffic starts to slowly degrade. In addition, at this saturation point, the corresponding CPU utilization of the aggregated flow reaches 100%, as shown in Figure 7(b).

The reason for this saturation point when targeting rule number 5,000 is that the traffic intensity or offered load $\rho = \lambda \bar{X}$ reaches 1, that is the capacity of the system reaches its maximum processing rate of $1/\bar{X}$, which is equivalent to *approximately* $1/(2.65\mu s + 5000 \times 0.05\mu s)$ or 3.958 Kpps. Similarly, and as shown in Figure 6, the saturation points when targeting rule numbers 1000 and 10,000 are observed when approaching their maximum processing rates. The slow degradation of the normal and constant rate of ITG traffic beyond the saturation point of 4 Kpps, as observed in Figure 7(a), is attributed to the fact that the packet loss probability starts to increase as the incoming DoS attack rate increases, thereby decreasing the throughput gradually of the constant rate of ITG traffic. This is also confirmed by Figure 7(b) in which the corresponding CPU utilization given to processing ITG traffic also decreases gradually. On the other hand, it is shown that the CPU utilization resulting from the DoS attack flow continues to gradually increase beyond the saturation point. This gradual increase can be attributed to the continued increase of incoming rate of DoS flow. It is to be noted that the throughput of DoS flow flattens off and is sustained approximately at the saturation point, as the system can not process more than its maximum capacity of 3.958 Kpps, despite the continued gradual increase of incoming DoS attack traffic rate.

## VI. Conclusion

We have presented and validated an analytical model to study and analyze the performance of rule-based network firewalls. From the model, we have derived key features and performance measures of engineering and design significance. These key features and measures include throughput, packet loss, packet delay, and CPU utilization. The model can be used to measure the performance when the firewall is subjected to normal traffic flows as well as DoS attack flows targeting different rule positions. It was demonstrated that targeting rules at the bottom of a relatively large ruleset can be severely detrimental to the performance of the firewall. As a good design practice and vital countermeasure against DoS attacks that target bottom rules, it is recommended to minimize the size of the firewall ruleset or to rearrange dynamically rules so that bottom rules can be served at the top of the ruleset, thereby making it harder to launch such complexity-algorithmic attacks that target bottom-rules. As a future work, we plan to model and analyze the performance of different solutions to mitigate DoS attacks targeting bottom rules. Specifically, we plan to study and analyze the performance of firewalls when implementing the mitigation solution of real-time dynamic re-ordering of the ruleset in which frequently triggered rules are placed on the top of the ruleset.

## References


[1] "Cisco PIX firewall release notes," 2004. Available: http://www.cisco.com/en/US/docs/security/pix/pix62/release/notes/pixrn624.html

[2] "Linux Netfilter." Available: http://www.netfilter.org

[3] A. J. Melara, "Performance analysis of the Linux firewall in a host," Master's thesis, California Polytechnic State University, June 2002.

[4] "FreeBSD ipfw." Available: http://www.freebsd.org/doc/en/books/handbook/firewalls-ipfw.html

[5] K. Salah, K. Sattar, M. Sqalli, and E. Alshaer, "A potential low-rate dos attack against network firewalls," *Int'l J. Security and Commun. Networks*, vol. 4, no. 2, pp. 109–238, Feb. 2011.

[6] Arbor Networks Inc., "Worldwide infrastructure security report, volume vi," 2010. Available: http://www.arbornetworks.com/report

[7] S. Acharya, J. Wang, Z. Ge, T. Znati, and A. Greeberg, "Simulation study of firewalls to aid improved performance," in *Proc. 2006 Simulation Symposium*.

[8] K. Salah, "Queueing analysis of network firewalls," in *Proc. 2010 IEEE Globecom*, pp. 1–5.

[9] A. El-Atawy, T. Samak, E. Al-Shaer, and H. Li, "Using online traffic statistical matching for optimizing packet filtering performance," in *Proc. 2007 IEEE INFOCOM*, pp. 866–874.

[10] H. Hamed, A. El-Atawy, and E. Al-Shaer, "Adaptive statistical optimization techniques for firewall packet filtering," in *Proc. 2006 IEEE INFOCOM*.

[11] E. Al-Shaer, H. Hamed, R. Boutaba, and M. Hasan, "Conflict classification and analysis of distributed firewall policies," *IEEE J. Sel. Areas Commun.*, vol. 23, no. 10, pp. 2069–2084, Oct. 2005.

[12] E. Al-Shaer and H. Hamed, "Modeling and management of firewall policies," *IEEE Trans. Network Service Management*, vol. 1, no. 1, pp. 2–10, 2004.

[13] A. Mayer, A. Wool, and E. Ziskind, "Fang: a firewall analysis engine," in *Proc. 2000 IEEE Symposium on Security and Privacy*.

[14] L. Yuan, J. Mai, Z. Su, H. Chen, C. Chuah, and P. Mohapatra, "Fireman: a toolkit for firewall modeling and analysis," in *Proc. 2006 IEEE Symposium on Security and Privacy*.
[15] E. W. Fulp, "Optimization of network firewalls policies using directed acyclic graphs," in *Proc. 2005 IEEE Internet Management Conference*.
[16] J. Qian, S. Hinrichs, and K. Nahrstedt, "ACLA: a framework for access control list (ACL) analysis and optimization," in *Commun. and Multimedia Security*, 2001.
[17] G. Misherghi, L. Yuan, Z. Su, C.-N. Chuah, and H. Chen, "A general framework for benchmarking firewall optimization techniques," *IEEE Trans. Network Service Management*, vol. 5, no. 4, pp. 227–238, 2008.
[18] A. X. Liu and M. G. Gouda, "Diverse firewall design," *IEEE Trans. Parallel Distrib. Syst.*, vol. 19, no. 9, pp. 1237–1251, Sep. 2008.
[19] M. G. Gouda and A. X. Liu, "Structured firewall design," *Computer Networks: The Int'l J. Computer and Telecommun. Networking*, vol. 51, pp. 1106–1120, Mar. 2007.
[20] C. R. Meiners, A. X. Liu, and E. Torng, "Topological transformation approaches to optimizing tcam-based packet classification systems," in *Proc. 2009 International Joint Conference on Measurement and Modeling of Computer Systems*, pp. 73–84.
[21] B. Hickman, D. Newman, S. Tadjudin, and T. Martin, "Benchmarking methodology for firewall performance," RFC3511, Apr. 2003.
[22] M. Lyu and L. Lau, "Firewall security: policies, testing and performance evaluation," in *Proc. 2000 IEEE International Computer Software and Applications Conference*, pp. 116–121.
[23] V. Santiraveewan and Y. Permpoontanalarp, "A graph-based methodology for analyzing IP spoofing attack," in *Proc. 2004 IEEE International Conference on Advanced Information Networking and Applications*, pp. 227–231.
[24] S. Kamara, S. Fahmy, E. Schultz, F. Kerschbaum, and M. Frantzen, "Analysis of vulnerabilities in Internet firewalls," *Int'l J. Computers and Security*, vol. 22, no. 3, pp. 214–232, 2003.
[25] D. Goldsmith and M. Schiffman, "Firewalking: a traceroute-like analysis of IP packet responses to determine gateway access control lists," Oct. 1998. Available: http://www.packetfactory.net/firewalk/firewalk-final.html
[26] D. Gross and C. Harris, *Fundamentals of Queueing Theory*. Wiley, 1998.
[27] H. Takagi, *Queueing Analysis, Vol. 1: Finite Systems*. North-Holland, 1993.
[28] L. Kleinrock, *Queueing Systems, Vol. 1: Theory*. John Wiley & Sons, 1975.
[29] M. F. Neuts, *Matrix-Geometric Solutions in Stochastic Models: An Algorithmic Approach*. Dover Publications Inc., 1981.
[30] C. Krishna and Y. Lee, "A study of two-stage service," *Operation Research Lett.*, vol. 9, pp. 91–97, 1990.
[31] B. Dosh, "Analysis of a two phase queueing system with general service times," vol. 10, pp. 265–272, 1991.
[32] S. A. Crosby and D. S. Wallach, "Denial of service via algorithmic complexity attacks," in *Proc. 2003 USENIX Security Symposium*, pp. 29–44.
[33] X. Cai, Y. Gui, and R. Johnson, "Exploiting unix file system races via algorithmic complexity attacks," in *Proc. 2009 IEEE Symposium on Security and Privacy*.
[34] R. Smith, C. Estan, and S. Jha, "Backtracking algorithmic complexity attacks against a NIDS," in *Proc. 2006 Annual Computer Security Applications Conference*.
[35] J. Botwicz, P. Guciak, and P. Sapiecha, "Building dependable intrusion prevention systems," in *Proc. 2006 International Conference on Dependability of Computer Systems*.
[36] M. Guirguis, A. Bestavros, and I. Matta, "Exploiting the transients of adaptation for RoQ attacks on Internet resources," in *Proc. 2004 IEEE International Conference on Network Protocols*, pp. 184–195.
[37] D. Bovet and M. Cesati, *Understanding the Linux Kernel*, 3rd edition. O'Reily, 2005.
[38] M. McKusick, K. Bostic, M. Karels, and J. Quarterman, *The Design and Implementation of the 4.4BSD Unix Operating System*. Addison Wesley, 1996.
[39] K. Salah and A. Qahtan, "Implementation and experimental performance evaluation of a hybrid interrupt-handling scheme," *Int'l J. Computer Commun.*, vol. 32, no. 1, pp. 179–188, 2009.
[40] R. Jain, *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. John Wiley & Sons, 1991.
[41] M. Karam and F. Tobagi, "Analysis of delay and delay jitter of voice traffic in the Internet," *Computer Networks Mag.*, vol. 40, no. 6, pp. 711–726, Dec. 2002.
[42] W. Leland, M. Taqqu, W. Willinger, and D. Wilson, "On the self-similar nature of Ethernet traffic," *IEEE/ACM Trans. Networking*, vol. 2, no. 1, pp. 1–15, Feb. 1994.
[43] V. Paxson and S. Floyd, "Wide-area traffic: the failure of Poisson modeling," *IEEE/ACM Trans. Networking*, vol. 3, no. 3, pp. 226–244, June 1995.
[44] W. Willinger, M. Taqqu, R. Sherman, and D. Wilson, "Self-similarity through high-variability: statistical analysis of Ethernet LAN traffic at the source level," in *Proc. 1995 ACM SIGCOMM*, pp. 100–113.
[45] A. Law and W. Kelton, *Simulation Modeling and Analysis*, 2nd edition. McGraw-Hill, 1991.
[46] J. White, "An effective truncation heuristic for bias reduction in simulation output," *Simulation J.*, vol. 69, no. 6, pp. 323–334, Dec. 1997.
[47] "Distributed Internet traffic generator," 2008. Available: http://www.grid.unina.it/software/ITG
[48] S. Zander, D. Kennedy, and G. Armitage, "KUTE: a high performance kernel-based UDP traffic engine," CAIA, Tech. Rep., 2005. Available: http://caia.swin.edu.au/reports/050118A/CAIA-TR-050118A.pdf

**Khaled Salah** is associate professor in the Department of Computer Engineering, Khalifa University of Science, Technology and Research, UAE. He received the B.S. degree in Computer Engineering with a minor in Computer Science from Iowa State University, USA, in 1990, the M.S. degree in Computer Systems Engineering from Illinois Institute of Technology, USA, in 1994, and the Ph.D. degree in Computer Science from the same institution in 2000. His primary research interests are in the areas of computer and network security, network design, queueing systems, and performance evaluation and modeling. Dr. Salah published several research articles on network firewalls, Snort NIDS, deployment of triple-play network services, and Linux performance. Dr. Salah is an Editorial Board member of several prestigious international journals including *IET Communications*, Elsevier JNCA, Wiley IJNM, Wiley SCN, and J.UCS. Dr. Salah was the recipient of KFUPM University Excellence in Research Award of 2008/09 and the recipient of KFUPM Best Research Project Award of 2009/10.

**Khalid Elbadawi** is currently a Ph.D. candidate at the School of Computing, College of Computing and Digital Media, DePaul University, USA. He received his BS degree in Mathematics and Computer Science from University of Khartoum, Sudan, in 1994. From 1994-2000, he worked with NARIS Inc. in the software development of network and communication tools. In 2001, he joined King Fahd University of Petroleum and Minerals and obtained his MS degree in 2003. Khalid worked at KFUPM as a lecturer for two years before being admitted to the Ph.D. program at DePaul University, Chicago. His research interests are in performance analysis, operating systems, and network security and management.

**Raouf Boutaba** received the M.Sc. and Ph.D. degrees in computer science from the University Pierre & Marie Curie, Paris, in 1990 and 1994, respectively. He is currently a professor of computer science at the University of Waterloo and a distinguished visiting professor at the division of IT convergence engineering at POSTECH. His research interests include network, resource and service management in wired and wireless networks. He is the founding editor in chief of the IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT (2007-2010) and on the editorial boards of other journals. He has received several best paper awards and other recognitions such as the Premier's Research Excellence Award, the IEEE Hal Sobol Award in 2007, the Fred W. Ellersick Prize in 2008, and the Joe LociCero and the Dan Stokesbury awards in 2009. He is a fellow of the IEEE.