

EPM-RT-2006-07

ON THE EFFICIENCY OF TIMEOUT-BASED
DOS ATTACK PROTECTIONS

Boteanu D., Fernandez J.M., Mullins, J.
Département de génie informatique
École Polytechnique de Montréal

Octobre 2006

©2006

Boteanu D., Fernandez J.M., Mullins, J.
Tous droits réservés

Dépôt légal :

Bibliothèque nationale du Québec, 2006
Bibliothèque nationale du Canada, 2006

EPM-RT-2006-07

On the efficiency of timeout-based DoS attack protections

par : Boteanu, D., Fernandez, J.M., Mullins, J.

Département de génie informatique
École Polytechnique de Montréal

Toute reproduction de ce document à des fins d'étude personnelle ou de recherche est autorisée à la condition que la citation ci-dessus y soit mentionnée.

Tout autre usage doit faire l'objet d'une autorisation écrite des auteurs. Les demandes peuvent être adressées directement aux auteurs (consulter le bottin sur le site <http://www.polymtl.ca/>) ou par l'entremise de la Bibliothèque :

École Polytechnique de Montréal
Bibliothèque – Service de fourniture de documents
Case postale 6079, Succursale «Centre-Ville»
Montréal (Québec)
Canada H3C 3A7

Téléphone :

(514) 340-4846

Télécopie :

(514) 340-4026

Courrier électronique :

biblio.sfd@courriel.polymtl.ca

Ce rapport technique peut-être repéré par auteur et par titre dans le catalogue de la Bibliothèque : <http://www.polymtl.ca/biblio/catalogue/>

On the efficiency of timeout-based DoS attack protections

Daniel Boteanu, José M. Fernandez, John Mullins

Department of Computer Engineering

École Polytechnique de Montréal

C.P. 6079, succ. centre-ville

Montréal, Québec, Canada

H3C 3A7

{daniel.boteanu,jose.fernandez,john.mullins}@polymtl.ca

10 September 06

Abstract

In this paper, we study the performance of timeout-based protection against flood denial-of-service (DoS) attacks on connection-oriented protocols, where server resources are depleted by incompleted illegitimate requests generated by the attacker (e.g. SYN-flooding attacks). The use of connection timeouts by most of these protocols has the side-effect of providing some protection against DoS attacks, since illegitimate requests time out more often than legitimate ones. For that reason, simple threshold-based dynamic adjustments of timeout values have been proposed and implemented as a DDoS attack mitigation counter-measure: when the number of connections go above a certain threshold, decrease the timeout value. Here, we introduce a Markov chain model in order to analyse the tradeoff between attacker and defender resource consumptions for various connection management strategies, whilst maintaining a similar quality of service. We use this model and experimental simulations to compare the performance of the simpler threshold-based strategies with a slightly more sophisticated, yet simple and efficient, linear timeout adjustment strategy. Both modeling and simulations indicate that threshold-based protection performs better than the standard fixed-timeout strategy, and that linear timeout adjustment performs better than both. We further consider the performance of two alternative methods of implementing adjustable timeout strategies: the *deterministic* method, where connections are flushed when a pre-determined expiry time (based on the timeout value at arrival) is reached, and the *deferred* method, where elapsed time is continuously compared with the current timeout value. Simulations indicate that the latter has superior performance in all cases.

1 Introduction

A denial-of-service (DoS) network attack occurs when the victim receives a malicious stream of packets that prevent the legitimate communication from taking place. DoS flood attacks consist in sending the victim (typically a server) a higher volume of traffic than it can handle. This can be achieved either by saturating the server's network connection or by using weaknesses in the communication protocols that typically allow the attacker to generate high server resource usage for a limited attacker effort [1]. Distributed denial-of-service (DDoS) attacks are simply DoS attacks performed by multiple agents, most frequently simultaneously. While attacks meant to saturate the server's network connection are addressed by routing congestion and Quality of Service (QoS) mechanisms, flood attacks taking advantage of protocol weaknesses are much harder to cope with because of their diversity. Similarities between malicious and legitimate traffic make it more difficult to protect against these types of attacks. Typical attacks of this kind include the SYN-flood attack and SSL connection depletion attack.

Industry and academia have provided several defense techniques against connection depletion attacks. These techniques do not completely solve the issue of the connection depletion attacks and can often

be avoided by attackers. Some of these techniques prevent the attack itself only to open the door to other possible attacks. However, there are no suitable formal models for quantitatively analyzing the performance of these techniques against DoS attacks and their tradeoffs in terms of QoS. In this paper, we hope to partially address this issue not only by proposing new protection strategies, but more importantly, by providing strong quantitative evidence of increased performance.

Following this introduction, Section 2 provides an overview of previous related work on modeling DoS attacks and protection strategies. In Section 3, we propose a stochastic modeling tool for DoS attacks, based on Markov chains. Using this model, we analyse three difference timeout-based protection strategies in Section 4. We provide in Section 5 the experimental results obtained by simulating these strategies according to two different implementations. Finally, we conclude and directions for future work in Section 6.

2 Previous Related Work

In this section we will provide an overview of previous work related to DoS modeling and prevention strategies. We also discuss previous work that have proposed the use of Markov chains as a modelling tool in computer and network security.

Exhausting the connections of a server is always possible but this comes with a cost to the attacker. It is important to note that there is a direct relationship between the attacker resources and the attack impact on one hand and between the server resources and service level provided on the other hand. These tradeoffs were first formalized by Meadows [2, 3] in a formal framework for analyzing network denial of service attacks. The framework itself is not sufficient for measuring the degree to which a protocol is vulnerable to denial of service because of the difficulty of providing concrete cost values for elementary operations (e.g. blocking a message, computing a digital signature). Finally, it does not consider the “cost” to the defender of a reduced QoS in situations where the server is still functional during (or after) the DoS attack.

Another approach is to try to understand the tradeoff between the attacker resources and attack impact on various network designs. This was studied by the authors of [4] that suggest a formal model for DDoS attacks in centralized and distributed networks. Experimental results showed that distributed networks perform better faced to a DDoS attack than centralized networks. In our case, we are trying to show that the tradeoffs can be further improved by modifying the server configuration in the same way that they are improved by the using different network designs.

Various methods for protecting against DoS attacks have been suggested. We concentrate here on those directed against connection-oriented protocols, and in particular the SYN-flood attack.

The first class of protection mechanisms try to tip the tradeoff between attack and defender resource consumption by forcing the initiating party (i.e. the potential attacker) to spend extra effort before server resources are committed. In this class are *SYN-cookies* [5] which are particular initial TCP sequence numbers, generated by the server with the purpose of rendering the three-way handshake of TCP connections stateless. Although the rationale behind SYN-cookies is straightforward, the implementation has to deal with several constraints like fitting in the space defined for the TCP header and containing all the TCP options sent by the client in the SYN message. SYN-cookies are implemented in Linux systems and McAfee security appliances [6] but the current implementations deals differently with the TCP protocol constrains handling only some of them [7]. Also in that category are *client puzzles* [8], that consist in having the server generate a cryptographic puzzle that the client must answer correctly before it is given service. Suggestions have been made to implement the client puzzles in the IP protocol [9]. Limitations of this solution are due in part to the current limitations of the IP header but also to the fact that it could lead to adverse impact on legitimate clients, i.e. a reduction in QoS.

The second class tries to address the weaknesses of TCP/IP in that respect. For example, it has been

proposed [10] that the the first SYN message sent in the three-way handshake be intentionally dropped. Although attacks that do not expect such a protection mechanism might fail, once this is known, attacks can simulate the behavior of legitimate clients and bypass the protection mechanism. Furthermore, because the first SYN message is dropped, the latency for connection establishment is increased, again potentially reducing QoS. Since in SYN-flood attacks, IP spoofing can be used to render the detection of an attack very difficult, solutions have been developed that address this particular problem. For example, Network Ingress Filtering (RFC2267) [11] specifies that Internet Service Providers (ISP) routers should only accept traffic from IP addresses that are supposed to be behind those routers. Unfortunately, while there are vendor-provided solutions to implement it, ingress filters are not yet widely deployed. A possible amelioration of the RFC2267 would be to have a distributed framework of heterogeneous systems that would cooperate to defend against DDoS attacks [12, 13]. Since for political and social reasons the RFC2267 was not adopted world-wide, it is therefore not reasonable to believe that such a distributed framework could be set up over the Internet in the near future.

Solutions in the third category do not suffer from such problems, as they can be applied on individual servers or routers. For example, the SYN-flood protection solution that is implemented in the TCP/IP stack of the Microsoft Windows Server [14] operating systems consists in having administrator-configurable thresholds on the number of half-open connections and resent SYN ACK messages. When either of these thresholds is reached the system enters in a “protective” state where the time for which the resources remain allocated is decreased. Also, the amount of information stored during the three-way handshake of TCP connection establishment is reduced and some TCP options are disabled in the protective state. McAfee has proposed a similar but more general solution [6], where the administrator can configure thresholds on different parameters that are used to trigger attack detection. Nonetheless, the problem is that it is not clear at all what the optimal threshold values are as there is no quantitative model for establishing this. In order to try to fill this gap, we have used Markov Chains to build a model with which we can answer some of these questions.

3 DoS Modeling with Markov Chains

Markov chains are a stochastic modeling tool that describe the states and dynamics of a system at successive times. Markov chains are said to have the *memoryless property* if the probability of transition between any two states is independent of the previous states; this is also referred to as being *markovian*. This is in fact equivalent to the fact that state transition events are distributed in time according to a Poisson distribution. Nonetheless, Markov chains can still be used as a formal modeling tool for systems in which this is not the case, i.e. systems where state transitions probabilities will depend on past history. In instances where the key parameters such as rate of arrivals and departures are known, the model can be “solved”. First, this means that given state probabilities at given time, predictions can be made about state probabilities at a later time. We can also compute *steady-state probabilities*, which correspond to the likelihood of the various states at the equilibrium of the system.

Markov chains are suitable for modeling network performance and has been used in that purpose for many years. In particular, Markov Chains have also been used as modeling tool in network security. Baras [15] suggests detecting route falsification attacks in mobile ad-hoc networks (MANET) using a Hidden Markov Model (HMM). More recent studies [16] show that using edge sampling techniques along with HMM can be used to reconstruct a network attack path. HMMs can also be used in Intrusion Detection Systems [17, 18], the transitions between each state in the Markov model being generated by intrusion, detection and recovery events. In our case, we will use Markov chains to model the performance of servers under DoS attacks. In the rest of section we will describe our use of Markov chains to model the behaviour of servers under DDoS attacks in connection oriented protocols.

3.1 Description of the model

Typical Markov chains models used in network performance consists in having each state characterized by the number of connections in the system. A maximum number of connection c can be served in the same time. Connections arrive with the rate λ and are served with the rate μ . Figure 1 illustrates the state transition diagram for a typical Markov chain model.

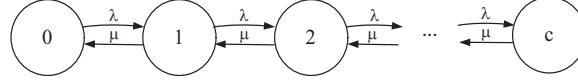


Figure 1: Typical network performance Markov chain model

In our case, each state in the chain is characterized by two values: N_l and N_m ; the number of connections used by legitimate users and malicious users, respectively. A maximum number of both legitimate and malicious connection c can be served in the same time. All connection requests that arrive when the server is in a saturated state ($N_l + N_m = c$) will be rejected. Transitions between states occur with different rates for the legitimate and malicious connection requests: λ_l and μ_l for the arrivals and servings of legitimate connections and λ_m and μ_m for the arrivals and servings of malicious connections. The state transition diagram is presented in Figure 2. The chain has a triangular form where states on the upper line represent that no malicious connections are present in the system and states on the diagonal represent that only malicious connections are present in the system.

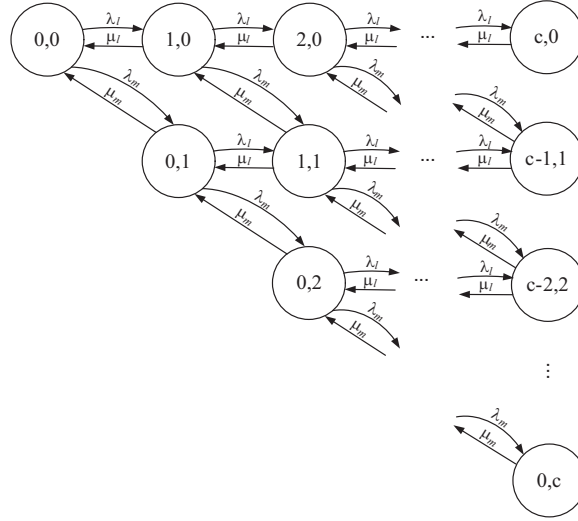


Figure 2: Triangular DoS Markov chain model

The following events generate transitions between states:

- *connection arrival*, meaning that the server received a connection request from a client. It occurs at a rate λ ;
- *connection completed*, meaning that the connection was either elevated or that the client was served with the required information and the connection was closed successfully. It occurs at a rate μ_l ;
- *connection rejected*, meaning that the server was not able to serve the connection because no more connections channels were available. It occurs at a rate ϕ_r ;
- *connection expired*, meaning that the server tried to serve the connection but the communication timed out and the connection was dropped; It occurs at a rate ϕ_e ;

- *connection failed*, meaning that the connection was either rejected or expired. It occurs at a rate $\phi = \phi_r + \phi_e$.

In the particular case of a SYN-flood attack, N_l and N_m will actually represent the number of connections that are half-open. The *connection arrival* and *connection completed* events represent a SYN message and the corresponding ACK message being received by the server. In the case of an SSL connection depletion attack, N_l and N_m represent the number of connections that have not yet established a secure channel and for which the negotiation phase is still in progress. The *connection arrival* events represent the Client hello message being received by the server and the *connection completed* events represent the corresponding Finished message being sent by the server.

It is even possible to consider a nested model, each level representing a layer in the protocol stack. In this case, the values of N_l , N_m will have different significations at each level. Also, a *connection completed* at level l would correspond to a *connection arrival* event at level $l + 1$.

How realistic is this model? It is known that user sessions initiations resemble phone calls [19] and thus have a Poisson arrival process with exponential inter-arrival times. We will make the supposition that all incoming connections follow this pattern. In most cases, serving rates depend only of network transit times but in most cases user interaction is also a factor. It has been shown that because of the network queuing algorithms, all the IP packet traffic tends toward a Poisson process as the load increases [20]. We will therefore make the supposition that connections complete events are generated at exponential intervals of time from the connection arrived events if the timeout has not elapsed and connection expire events are generated at timeout intervals of time from the connection arrived events otherwise.

The rate at which *connection completed* messages are generated by the legitimate clients is μ_l . Only messages that arrive to the server before the timeout elapses will generate *connection completed* events; i.e. in general we have that $\mu_l \geq \mu_c$. All the other will be ignored by the server and *connection expired* events are generated when the timeout elapses. Therefore, the Probability Distribution Function (PDF) of the legitimate connection service time ($G_l(t)$) will have the form of an exponential distributions for t smaller than the timeout (t_{out}) followed by a dirac function in t_{out} . The weight of the dirac function will be so that the Cumulative Distribution Function (CDF) of the service time is 1 in t_{out}^+ and represents the probability that an individual connection will expire:

$$G_l(t) = \begin{cases} \mu_c e^{-t\mu_c} & t < t_{\text{out}} \\ 0 & \text{otherwise} \end{cases} + \delta(t - t_{\text{out}})p_{\text{expire}} \quad (1)$$

where

$$p_{\text{expire}} = \int_{t_{\text{out}}}^{\infty} \mu_c e^{-t\mu_c} dt = e^{-t_{\text{out}}\mu_c} \quad (2)$$

The mean service time for legitimate connections is:

$$t_l \triangleq \int_0^{\infty} t G_l(t) dt = \frac{1 - e^{-t_{\text{out}}\mu_c}}{\mu_c} \quad (3)$$

The legitimate connections service rate is:

$$\mu_l \triangleq \frac{1}{t_l} = \frac{\mu_c}{1 - e^{-t_{\text{out}}\mu_c}} \quad (4)$$

In order to pass undetected by time analysis detection methods, an attacker would want to simulate as much as possible the legitimate traffic and would also generate connections using a Poisson process

with the rate λ_m . The strategy of the attacker is to exhaust the server resources using the smallest effort possible. This is achieved by generating the *connection arrival* events and then abandoning the communication without any notice to the server. Malicious connections will eventually all expire and generate *connection expired* events at t_{out} intervals of time from the *connection arrival* events. The malicious connections service rate is:

$$\mu_m \triangleq \frac{1}{t_{\text{out}}} \quad (5)$$

Memoryless Markov models are easily solvable, because their dynamics can be accurately represented as linear transformation of the state probability vectors. In our case, however, even though we made the assumptions that the arrival rates are markovian the model is only semi-markovian since the service rate that does not follow a Poisson (exponential) distribution. The model dynamics is thus non-linear, and hence it is not possible to use standard linear equations techniques to solve the model and calculate predictions. Although the triangular DoS Markov chain model that we presented describes the states in which the server will be during an attack, these states are not directly visible because individual connections can not be labeled as legitimate or malicious. For these two reasons stated, we will analyse the *visible Markov chain* that has $c+1$ states, each state being characterized by the number of connections (N) used by both legitimate and malicious users. The probability that the visible Markov chain is in a state N is the sum of all probabilities that the hidden Markov chain is in state (N_l, N_m) with $N_l + N_m = N$. This chain is similar to the typical Markov chain used in network performance represented in Figure 1

The visible connection arrival process is the sum of two Poisson processes with rates λ_l and λ_m and thus also a Poisson process with rate $\lambda = \lambda_l + \lambda_m$. In a Markov chain model the *load* is defined as the ratio between the arrival and service rates. In our case, we distinguish the load generated by the legitimate users ρ_l and the load generated by malicious users ρ_m .

$$\rho_l = \frac{\lambda_l}{\mu_l}; \quad \rho_m = \frac{\lambda_m}{\mu_m}; \quad (6)$$

The overall load cannot be computed directly because the service processes are not memoryless. Our goal is to compute the overall load by approximating the overall mean service time \tilde{t} . We consider \tilde{t} to be constant in time and equal to the average of the mean legitimate service time t_l and mean malicious service time t_m weighted by the legitimate load and the malicious load, respectively:

$$\tilde{t} = \frac{\rho_l}{\rho_l + \rho_m} t_l + \frac{\rho_m}{\rho_l + \rho_m} t_{\text{out}} \quad (7)$$

The approximative mean service rate in the visible chain is:

$$\tilde{\mu} \triangleq \frac{1}{\tilde{t}} = \frac{\mu_l \mu_m (\lambda_m \mu_l + \lambda_l \mu_m)}{\lambda_m \mu_l^2 + \lambda_l \mu_m^2} \quad (8)$$

We can now calculate an approximative overall load generated by both legitimate and malicious users as:

$$\tilde{\rho} \triangleq \frac{\lambda}{\tilde{\mu}} \quad (9)$$

With this approximation we can compute the steady-state probability that the system is in the state k using Erlang's loss formula:

$$p_k = \frac{\tilde{\rho}^k}{k!} / \sum_{i=0}^c \frac{\tilde{\rho}^i}{i!} \quad (10)$$

3.2 Approximate solutions to the model

Because the connections are served independently, the only significant performance measure is the probability that a legitimate connection will fail ϕ which is equal to the probability that the connection will be rejected ϕ_r plus the probability the connection will expire ϕ_e .

$$\phi = \phi_r + \phi_e \quad (11)$$

The *blocking probability* is by the definition the probability system is saturated, i.e. that the queue is full. A connection is rejected if the server is saturated when the *connection arrival* event is generated. The probability that a connection is rejected ϕ_r is thus equal to the probability that the server is in state c at that moment. If the system were at equilibrium, this will be exactly the steady-state probability p_c . If we assume that the system will never be far from equilibrium, we can approximate it as such, i.e. $\phi_r \approx p_c$.

A connection expires with the probability p_{expire} if the server was not saturated when the *connection arrival* event is generated. The connection expire probability can also be approximated with the steady-state probabilities as follows:

$$\phi_e \approx \sum_{k=0}^{c-1} p_k p_{\text{expire}} \quad (12)$$

In this model, the resources that the attacker spends to achieve a negative impact on the service level correspond to the malicious connections arrival rate λ_m . The resources that the server spends to assure the required service level is the capacity c . We are interested in how the tradeoff between the attacker and server resources varies for the same legitimate connection fail probability ϕ . Even though the fully expanded expression of Equation 11 is quite complex, what lies beneath it is a tradeoff between these quantities that is essentially linear for the same connection fail probability, as we have verified with several numerical simulations. Figure 3 illustrates the contour curves for the connection complete probability lines for different values of attack rates and server capacities; note that they are essentially straight.

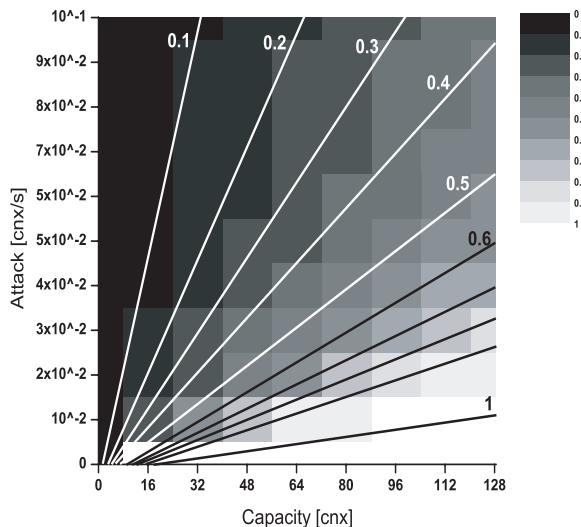


Figure 3: Steady-state legitimate connection complete probabilities for various capacities and attack rates, and fixed legitimate arrival rate $\lambda_l = 10$ cnx/s, service rate $\mu_l = 1$ cnx/s, and timeout $t_{\text{out}} = 75$ s.

Given a certain attack rate λ_m and server capacity c , the parameter that can be optimized is the timeout. As Figure 4 shows, the two components of the legitimate connections reject probability ϕ : ϕ_e and ϕ_r are in opposition. ϕ_e decreases exponentially with the timeout. When no attack is present ϕ_r is null for $\lambda_l < \mu_l c$; it has the limit $\lambda_l - \mu_l c$ for infinite timeout when $\lambda_l > \mu_l c$. When an attack is present, ϕ_r has

the limit λ_l when timeout is infinite. For a specific attack rate and capacity there is an optimal timeout value that can be calculated numerically.

In summary, inspired on typical Markov models used in network performance we introduced a richer model that captures both malicious and legitimate connections. We can use this later model to have a better description of the overall service rate and closed the loop by using this non-Poisson service rate in the typical network performance model. The resulting model allows us to quantitatively calculate the tradeoff between the attacker and server resources for the same service level. In those cases where the tradeoff is acceptable, the model will permit the calculation of the optimal timeout.

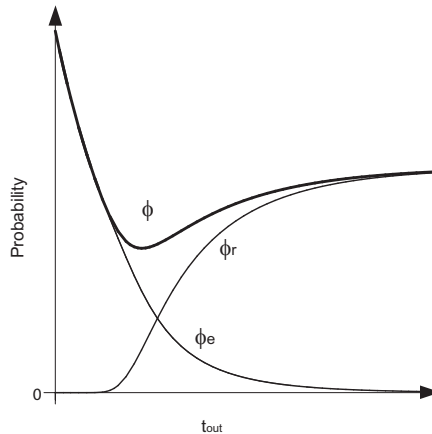


Figure 4: Steady-state reject, expire and fail probabilities.

4 Timeout-based Protection Strategies

We will now analyse two protection strategies that consist in adjusting the timeout. This, of course, in contrast with the standard strategy of having a fixed, non-adaptive connection timeout value. Ideally we would want to make this adjustment by looking at the bidimensional Markov chain and choosing a timeout according to the number of legitimate and malicious connections in the server. Unfortunately, this model is not visible because the server is unable to distinguish, unless some detection mechanism is used, if a connection request is legitimate or malicious. Therefore, the only information available to adjust the timeout is the total number of connections used. While the threshold prevention strategy is already implemented in Microsoft Windows Server 2003 and some security appliances, the second strategy, linear timeout prevention, is a concept that we introduce.

There are for each of these strategies, two alternative methods for deciding how to flush out timed out connections. The first is that which we call the *deterministic method*, and consists in tagging each connection with a pre-determined expiry time upon its arrival. The expiry time is simply the arrival time plus the timeout value at the moment of arrival. But to take into account the fact that the reality of the system might have changed drastically since the arrival of a connection, another approach would seem more suitable: to *defer* the assignation of an expiry time, such that if the timeout decreases after its arrival the connection is checked against the new timeout value. Thus at any given time, connections are flushed if the time elapsed since their arrival is bigger than the current timeout value. We refer to this as the *deferred method*. In the rest of this section, we instantiate the general Markov models of Section 3 and compute steady-state probabilities for the deterministic method only. We will nonetheless view simulation results for both in Section 5.

4.1 Threshold timeout prevention

The threshold prevention strategy consists in using a normal, long timeout t_0 at first. If the number of connections used in the server is greater than a certain threshold S , a shorter, attack timeout t_1 will be used. The timeout used will depend at all times of the state k in which the server is:

$$t_{\text{out}}^{(k)} = \begin{cases} t_0 & k < S \\ t_1 & \text{otherwise} \end{cases} \quad (13)$$

The probability that an individual connection will expire p_{expire} , the legitimate service rate μ_l and the approximative overall service rate $\tilde{\mu}$ described in equations (2), (4) and (8) all become state dependent:

$$p_{\text{expire}}^{(k)} = e^{-t_{\text{out}}^{(k)}\mu_c} \quad (14)$$

$$\mu_l^{(k)} = \frac{\mu_c}{1 - e^{-t_{\text{out}}^{(k)}\mu_c}} \quad (15)$$

$$\tilde{\mu}^{(k)} = \frac{\mu_l^{(k)}\mu_m(\lambda_m\mu_l^{(k)} + \lambda_l\mu_m)}{\lambda_m(\mu_l^{(k)})^2 + \lambda_l\mu_m^2} \quad (16)$$

We use the same principle as before to calculate the probability that the server is in a specific state k using Erlang's loss formula:

$$p_k = \frac{\frac{1}{k!} \prod_{j=0}^{k-1} \frac{\lambda_l + \lambda_m}{\tilde{\mu}^{(j)}}}{\sum_{i=0}^c \left(\frac{1}{i!} \prod_{j=0}^{i-1} \frac{\lambda_l + \lambda_m}{\tilde{\mu}^{(j)}} \right)} \quad (17)$$

Similar to the case where no protection strategy was used, the significant performance measure ϕ representing the legitimate *connection fail* event probability is calculated as:

$$\phi = \phi_r + \phi_e = p_c + \sum_{k=0}^{c-1} p_k p_{\text{expire}}^{(k)} \quad (18)$$

When using the threshold timeout protection strategy, the tradeoff between the attacker and server resources is also linear but more favorable for the server than with a fixed timeout. Figure 5 illustrates this tradeoff for numerical values of the rates (λ_l and μ_l), timeouts (t_0 and t_1) and threshold S similar to what we can find in Microsoft and McAfee products that use this strategy in a real-life scenario.

4.2 Linear timeout prevention

The linear timeout prevention strategy differs from the threshold prevention strategy in the way the timeout is decreased. Instead of suddenly decreasing the timeout when the server state reaches a certain threshold, this strategy gradually decreases the timeout as the number of connections in the server increases. When no connection is used (i.e. the server is in the state 0 an empty queue, long timeout t_0 is used; when all connections are used (i.e. the server is in the state c) a full queue, short timeout t_1 is used; a linear interpolation of the two values is used in all other server states. Thus, Equation 13 becomes:

$$t_{\text{out}}^{(k)} = t_0 + (t_1 - t_0) \frac{k}{c} \quad (19)$$

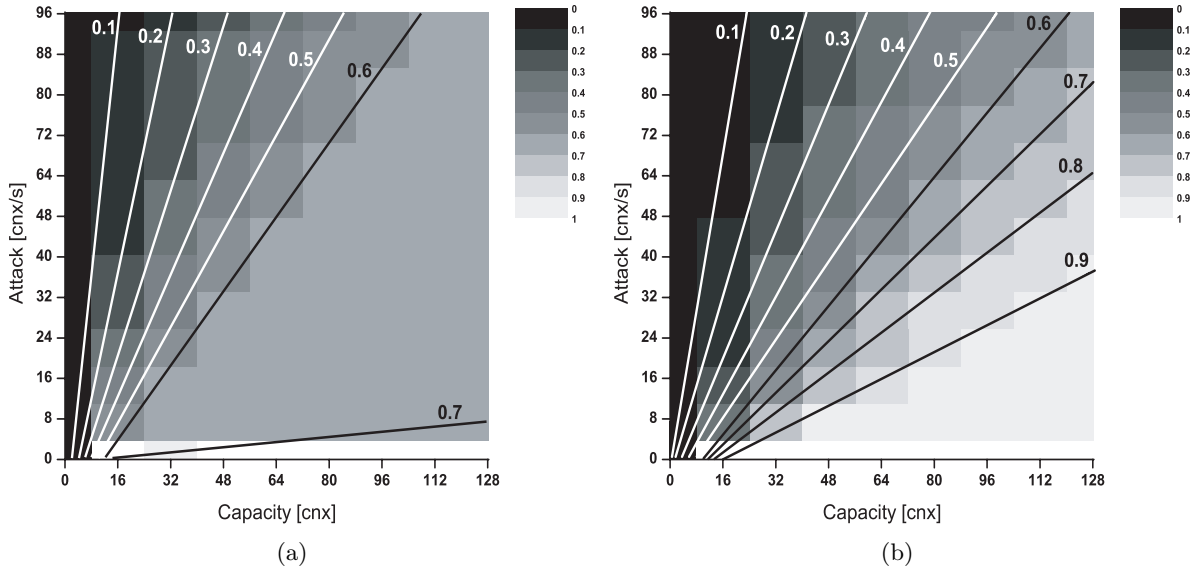


Figure 5: Steady-state legitimate connection complete probabilities for various queue capacities and attack rates, and fixed legitimate arrival rate $\lambda_l = 10$ cnx/s, service rate $\mu_l = 1$ cnx/s, and timeout values $t_0 = 75$ s, $t_1 = 1$ s for (a) a single threshold at $S = c/2$, and (b) linear adjustment.

The same Equations 14, 15 and 16 that describe the individual *connection expire* event probability $p_{\text{expire}}^{(k)}$, the legitimate service rate $\mu_l^{(k)}$ and the approximative overall service rate $\tilde{\mu}^{(k)}$ can be inserted in the Erlang loss formula described of Equation 17 to calculate the legitimate *connection fail* event probability:

$$\phi = \phi_r + \phi_e = p_c + \sum_{k=0}^{c-1} p_k p_{\text{expire}}^{(k)} \quad (20)$$

Once again, we are interested in the tradeoff between the attacker and server resources. Analysis of the two protection strategies show that for the same values of systems parameters and traffic (within the range explored), the linear timeout protection strategy could perform slightly better than the threshold timeout protection strategy. These results are illustrated in Figure 5(b). Finally, it is important to note that while the linear timeout prevention strategy is slightly more complex than the threshold timeout prevention strategy but computational overhead for a server implementing this strategy is negligible.

5 Experimental Results and Interpretation

We implemented these two protection strategies, in both their deterministic and deferred variants, and experimentally measured their performance using a traffic simulator¹. For the sake of comparison, we also implemented and measured the performance of the standard fixed-timeout strategy. The legitimate and malicious *connection requests* were generated using Poisson processes. The *connection complete* events for connections that did not expire were also generated using a Poisson process.

Five input data sets, each consisting in a series of scenarios, were run with both prevention strategies (threshold and linear timeout) and also with no prevention. We chose to use values representative for a SYN-flood attack, having each data set composed of scenarios in which the server's capacity c varied exponentially from 2 to 1024 cnx and the attack rate λ_m varied linearly from 0 to 768 cnx/s.

¹A collection of home made C++ programme were used to generate scenarios and data sets and to simulate and to simulate the behaviour of the server with the various strategies under the same scenarios.

The legitimate connections arrival rate λ_l used was 10 cnx/s and the connection response rate μ_c was 1 cnx/s. The timeout t_{out} was 75 s in the case of no prevention, which is a timeout value common in most operating systems TCP configuration. The same value was used for the threshold prevention strategy normal timeout t_0 and the linear prevention strategy empty queue timeout t_0 . For the threshold prevention strategy, we chose to use a threshold of $c/2$ and an attack timeout t_1 such that no more half of the legitimate connections in the queue would expire. This means $t_1 = 1/\mu_l = 1$ s. The same value was used for the linear prevention strategy for the full queue timeout t_1 . Note that these are the same values used in Figures 3 and 5 for the theoretical steady-state approximations, however the range of queue capacities and attack rates explored in simulations is much wider. The standard deviation was smaller than 10^{-2} for all scenarios and all three strategies tested (See Figure 7 in the appendix).

We can draw several conclusions from observation of this data, which are valid at least for the range of values observed. First, for all algorithms it seems that simulations confirm the exponential tradeoff between queue capacity and attack rate, that we observed using the steady-state approximations; this we can by observing that the contour curves are essentially linear when we plot capacity on a logarithmic scale, as we have done. Second, with both strategies it seems clear that in all cases the deferred method performs better than the deterministic method. Finally, while the linear method performs better in general, there is a slight inversion in the case of the deterministic method with high attack rates and small queues. In other words, it might seem that there might not a universally better strategy.

In order to better understand these possible compromises, it is useful to define the notion of *relative attack virulence*. In fact, our first observation seem to indicate that most important parameter in determining completion frequencies is the ratio between the rate of attack and the size of the queue, i.e. its relative virulence. Conversely, we can think of the inverse ratio as *relative attack speed*, which corresponds to the expected time (in seconds) it would take the attacker to fill the queue if he were able to completely flood the incoming network links, i.e. no legitimate requests in the queue. We thus conducted another set of simulations where we fixed the queue capacity to a more realistic 1024, and were able to explore wider range of attack rate and hence relative virulence. This is almost without loss of generality because of the first observation. As a result, Figure 6 gives a much clearer picture of the relative performance of the various methods we have discussed here.

Thus we can confirm the theoretical predictions we had made in Section 4. First, that both timeout adjustment strategies are much better than a fixed timeout. Second, that the linear timeout prevention strategy seems to perform slightly better than the threshold prevention strategy. In particular, we can observe that for very high or very low capacity/attack virulence levels the performance of the two protection strategy is similar. However, the differences in performance can be as high as 30%, when the attack speed is between 2 and 4 s. This is a relatively powerful attack, under which all strategies would notice a significant decrease in QoS (at least 25% legitimate connections lost), except the linear deferred strategy where QoS is very small (a few percent). The difference in performance could be explained by the fact that the threshold protection decreases the timeout too aggressively with the unnecessary effect of making many legitimate connections expire during such an attack.

6 Conclusions

Good quantitative engineering best practices, presently shine by their absence in many areas of network security. In this paper, we have made a modest (but honest) effort to partially address this issue by examining the performance of a certain type of protection strategies against flood DoS attacks against connection-oriented protocols: server- or appliance-based timeout adjustment strategies, an idea which is not new but on which little or no quantitative work had been done.

We have first constructed a Markov model describing the behavior of a server under DoS attack that tries to exhaust the available connections. This model has allowed us to gain intuition on the likely

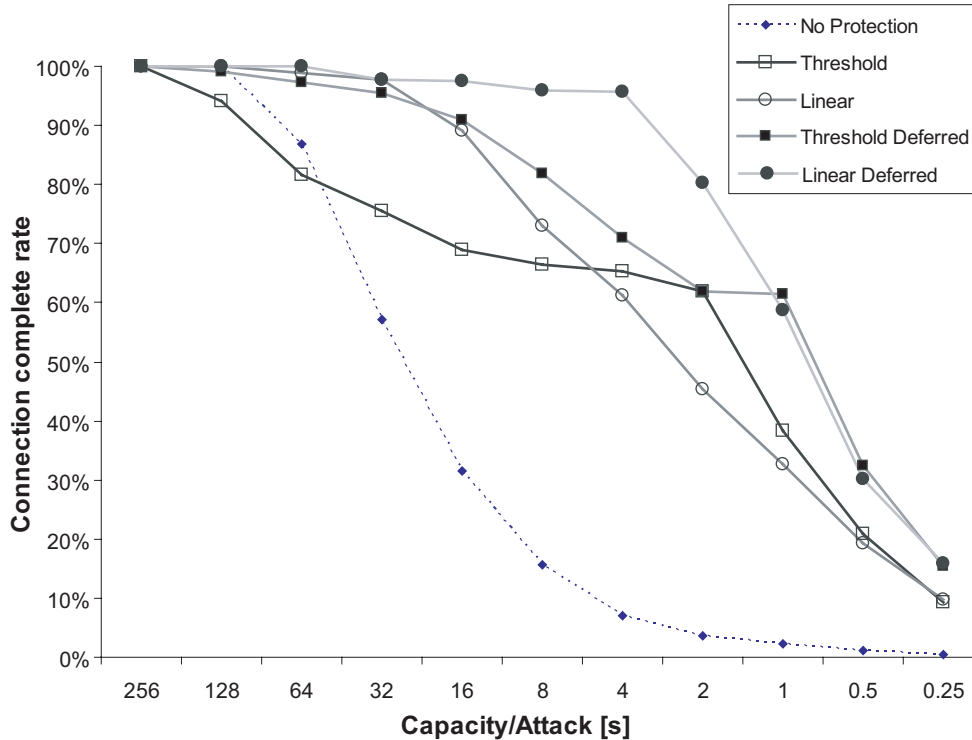


Figure 6: Legitimate connection complete rate for various strategies, with $c = 1024$, $\lambda_l=10$ cnx/s, $\mu_l=1$ cnx/s, $t_0=75$ s, $t_1=1$ s and various attack speeds.

tradeoffs between the various parameters that characterise a system under attack (traffic and service rates, queue size, etc.). It is however very suspect for two reasons. First, we have liberally replaced instantaneous probability vectors with steady-state approximations, thus assuming equilibrium, which is non-sensical in the case of virulent attacks. (For the intersecting values of capacity and attack rate between our numerical estimations and simulations, we did observe a discrepancy of up to 20% in some cases.). Second, we have not described in this paper the model for the deferred method of policing timeout connections out of the queue; that is the object of current research within our group.

Most objectably, however, we have assumed that the attack would send packets in a markovian fashion. While in the case of a large DDoS attack, the Internet might “poissonize” packet arrivals, this might not be true of a directed attack from selected high upstream bandwidth locations. Furthermore, it is not at all clear, even if we fix the average rate of arrivals, what kind of distribution the attacker should choose or avoid in order to optimise the effect on the target and/or possible detection by perimeter defenses. This question, we believe warrants further investigation. This criticism also applies to our simulation results, since we have generated our simulation traffic exactly according to a Poisson process. We hope to strengthen perform similar experiments on on large emulated test networks, whose traffic distribution should more closely resemble reality.

Nonetheless, the simulations results are quite conclusive and from interesting conclusions can be drawn that should of immediate application for those vendors and system administrators that are incorporating or using such strategies:

1. Adjusting timeout is almost always a good idea, except maybe for those applications requiring three 9’s QoS or better.
2. Fine-grained (e.g. linear) timeout adjustments seem to perform better than coarse single threshold adjustments, at least for attacks that are not extremely virulents (aka “Zen” DoS attacks).

3. In all cases, utilizing the deferred method of policing connections out of the queue shows superior performance.

It is important to note that the memory and CPU overhead of implementing these strategies is negligible, even for hardware-based implementations. As always, there are a few *caveat* regarding these principles. While we hinted at the possibility of optimising the timeout value in the fixed timeout case, we did not explore this possibility with the other strategies. In particular, we deliberately chose “industry standard” timeout values. The validity of our conclusions must be confirmed while varying the timeout values also. In particular, it might turn out that this might make, in the end, a bigger difference that changing timeout adjustment and queue policing strategy. This issue also, we hope to address in future research.

References

- [1] J. Mirkovic and P. Reiher. A taxonomy of DDoS attack and DDoS defense mechanisms. *SIGCOMM Comput. Commun. Rev.*, 34(2):39–53, 2004.
- [2] C. Meadows. A formal framework and evaluation method for network denial of service. In *PCSFW: Proceedings of The 12th Computer Security Foundations Workshop*. IEEE Computer Society Press, 1999.
- [3] C. Meadows. A cost-based framework for analysis of denial of service networks. *Journal of Computer Security*, 9(1/2):143–164, 2001.
- [4] Y. Chen, A. Bargteil, D. Bindel, R.H. Katz, and J. Kubiawicz. Quantifying network denial of service: A location service case study. In *Proc. International Conference on Information and Communication Security (ICICS)*, pages 340–351, 2001.
- [5] D. Bernstein. SYN cookies, 2003. <http://cr.yp.to/syncookies.html>.
- [6] McAfee. Deciphering detection techniques: Part III denial of service detection. http://www.mcafee.com/us/local_content/white_papers/wp_ddt_dos.pdf, 2005?? McAfee Network Security Technologies Group.
- [7] A. Zuquete. Improving the functionality of SYN cookies. In *Proc. IFIP TC6/TC11 6th Joint Working Conference on Communications and Multimedia Security*, pages 57–77, Deventer, The Netherlands, The Netherlands, 2002. Kluwer, B.V.
- [8] A. Juels and J. Brainard. Client puzzles: A cryptographic defense against connection depletion. In *Proc. Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, USA, 1999.
- [9] W. Feng, E. Kaiser, and A. Luu. Design and implementation of network puzzles. In *Proc. 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, volume 4, pages 2372–2382, New York, NY, USA, 2005. IEEE Computer Society Press.
- [10] B. Al-Duwairi and G. Manimaran. Intentional dropping: a novel scheme for SYN flooding mitigation. In *Proc. 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, volume 4, pages 2820–2824, 2005.
- [11] P. Ferguson and D. Senie. RFC2267, 1998. <http://www.ietf.org/rfc/rfc2267.txt>.
- [12] M. Robinson, J. Mirkovic, S. Michel, M. Schnaider, and P. Reiher. DefCOM: defensive cooperative overlay mesh. In *Proc. DARPA Information Survivability Conference and Exposition*, volume 2, pages 101–102, Washington, DC, USA, 2003.

- [13] J. Mirkovic, M. Robinson, P. Reiher, and G. Kuenning. Alliance formation for DDoS defense. In *Proc. New Security Paradigms Workshop (NSPW)*, page ??? ACM SIGSAC, August 2003.
- [14] Microsoft. Security considerations for network attacks. <http://www.microsoft.com/technet/security/topics/networksecurity/secdeny.msp>. Microsoft Technet.
- [15] J.S. Baras. Modeling and simulation of telecommunication networks for control and management. In *Proc. 35th Winter Simulation Conference: Driving Innovation*, New Orleans, LA, USA, 2003.
- [16] R. Varanasi, V.V. Phoha, and S. Joshi. IP-traceback based attacker tracking: A probabilistic technique for detecting internet attacks using the concept of hidden markov models. In *Proc. 5th IEEE Information Assurance Workshop, US Military Academy of West Point*, New York, NY, USA, 2004. IEEE Computer Society Press.
- [17] X.D. Hoang and J. Hu. An efficient hidden markov model training scheme for anomaly intrusion detection of server applications based on system calls. In *Proc. 12th IEEE International Conference on Networks (ICON)*, volume 2, pages 470–474, New York, NY, USA, 2004. IEEE Computer Society Press.
- [18] B.B. Madan, K. Goseva-Popstojanova, K. Vaidyanathan, and K.S. Trivedi. Modeling and quantification of security attributes of software systems. In *Proc. International Conference on Dependable Systems and Networks (DSN)*, pages 505–514, Washington, DC, USA, 2002. IEEE Computer Society.
- [19] C. Nuzman, I. Saniee, W. Sweldens, and A. Weiss. A compound model for TCP connection arrivals for LAN and WAN applications. *Comput. Networks*, 40(3):319–337, 2002.
- [20] J. Cao, W. Cleveland, D. Lin, and D. Sun. Internet traffic tends toward Poisson and independent as the load increases, 2002.

A Additional figures

Mean results of the legitimate connection completion rates when using the fixed-threshold strategy and when using threshold timeout and linear timeout protection strategies are presented in Figure 7. The standard deviation was smaller than 10^{-2} for all scenarios and all three strategies tested.

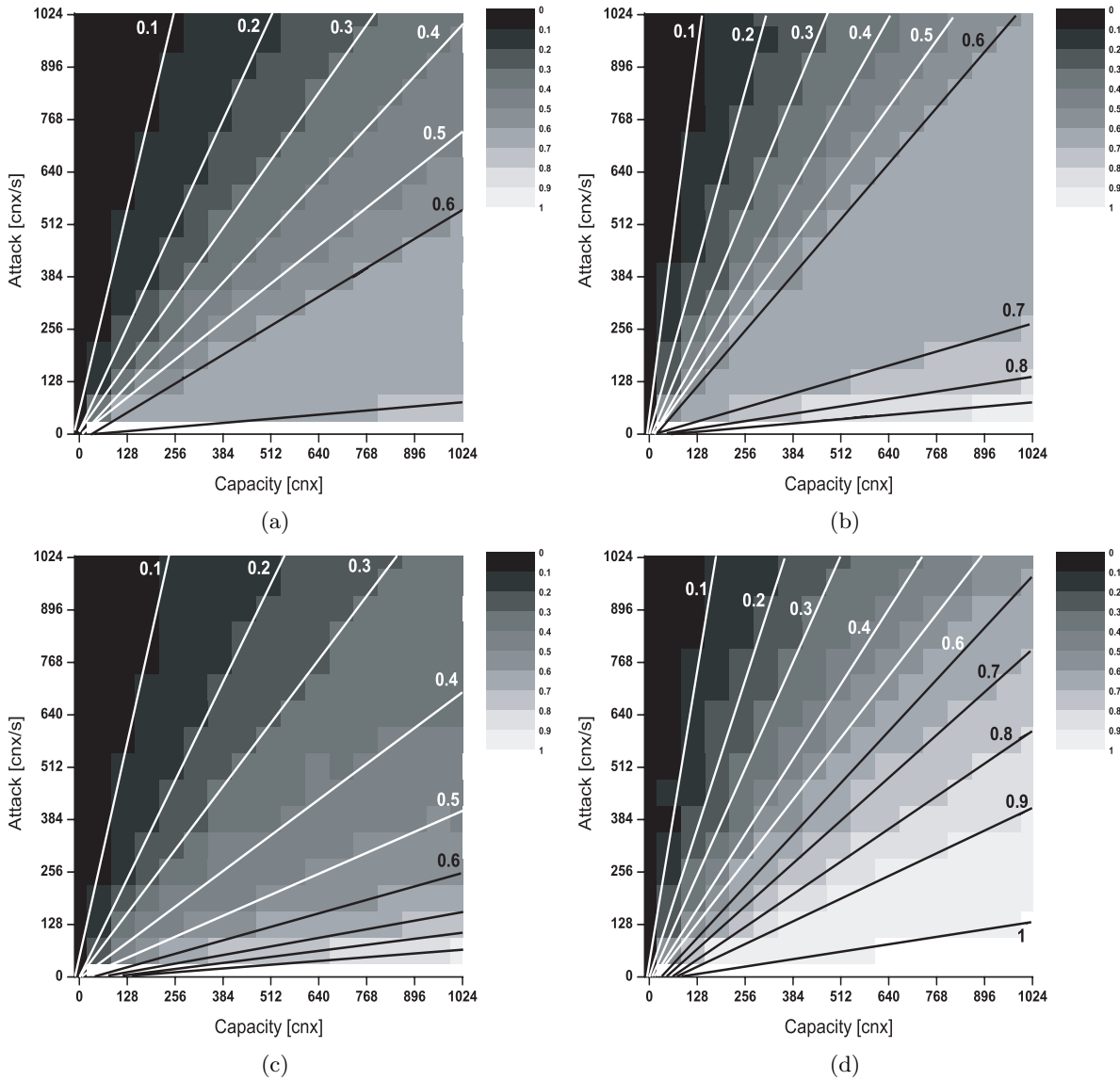


Figure 7: Simulation results showing legitimate connection complete frequencies for various queue capacities and attack rates, $\lambda_l=10$ cnx/s, $\mu_l=1$ cnx/s, $t_0=75$ s and $t_1 = 1$ s, for the single threshold, (a) and (b), and linear strategies, (c) and (d), using the deterministic and deferred methods, respectively.