Defending Against Low-rate TCP Attacks: Dynamic Detection and Protection*

Haibin Sun John C.S. Lui
Department of Computer Science & Engineering
The Chinese University of Hong Kong
Email: {hbsun,cslui}@cse.cuhk.edu.hk

David K.Y. Yau
Computer Science Department
Purdue University
Email: yau@cs.purdue.edu

Abstract

We consider a distributed approach to detect and to defend against the low-rate TCP attack [7]. The low-rate TCP attack is essentially a periodic short burst which exploits the homogeneity of the minimum retransmission timeout (RTO) of TCP flows and forces all affected TCP flows to back off and enter the retransmission timeout state. This sort of attack is difficult to identify due to a large family of attack patterns. We propose a distributed detection mechanism which uses the dynamic time warping method to robustly and accurately identify the existence of this sort of attack. Once the attack is detected, a fair resource allocation mechanism is used so that (1) the number of affected TCP flows is minimized, and (2) we provide sufficient resource protection for the affected TCP flows. We report experimental results to quantify the robustness and accuracy of the proposed detection mechanism and the efficiency of the defense method.

1. Introduction

By providing reliable delivery of user data, TCP simplifies application design and is being used in many network applications including file transfers, e-commerce, and web HTTP access. However, designing a reliable protocol for many heterogeneous users sharing an unreliable network is challenging and involves many subtle issues. Under severe network congestion, for example, TCP requires sources to reduce their congestion window to one packet and wait for a retransmission timeout (RTO) before attempting to resend. If there is further packet loss, the RTO is doubled after each subsequent loss. The RTO needs to ensure that TCP sources will give the network sufficient time to recover from a congestion event; Allman and Paxson [4] recommend a lower bound of one second for its value in order to achieve near-optimal network throughput.

However, while the TCP RTO can maximize network throughput when users are conforming and cooperative, it can also be exploited by a malicious user to effect a denial-of-service attack. In [7], the authors present a form of *low-rate* TCP attack, in which an attacker periodically sends attack traffic to overflow a router's queue and cause packet loss. As discussed, a well behaving TCP source will then back off to recover from the congestion and retransmit only after one RTO. If the attacker congests the router again at the times of retransmission, little or no real user traffic can get through. Hence, by synchronizing the attack period to the RTO duration, the attacker can essentially shut off most, if not all, legitimate TCP sources even though the long-term rate of attack traffic can be quite low. The low-rate attack raises serious concern because it can be significantly harder to detect than more traditional brute-force, flooding style attacks. Existing rate-limiting approaches [8, 16], for example, are designed to control aggressive attackers only.

We are interested in the detection of and defense against low-rate TCP attacks. Since TCP is widely implemented and deployed, a solution requiring changes to TCP and thus widespread modifications of users' software may not be practical. This motivates us to consider a solution approach that can be implemented in a resilient routing infrastructure and benefit a large community of standard TCP users.

For detection, since an attacker's primary objective is to ensure the periodic overflow of a router's buffer, a basic signature of attack traffic will be intermittent short bursts of high rate traffic in between periods of little or no activity (characterized by, say, a periodic square wave). In practice, however, attack traffic can deviate from the basic signature for various reasons: distortion caused by queueing in intermediate routers, aggregation with background traffic (e.g., UDP traffic), an attacker's own attempt to inject "noise" into its traffic to escape detection, etc. Moreover, in a distributed attack, the traffic from individual attack sources may not have the expected traffic characteristics, but the aggregation of such traffic does. Therefore, it is essential to develop de-

* J. C. S. Lui was supported in part by an RGC Grant. D. K. Y. Yau was supported in part by the National Science Foundation under grant numbers CCR-9875742 (CAREER) and CNS-0305496.

tection algorithms that are both robust to practical traffic distortions and efficient to carry out even at a busy router.

Once an attack has been detected, we seek to neutralize the effects of the attack traffic and minimize damage to legitimate users. The strategy is to rate limit and preferentially drop packets in an attack burst in order to reduce the loss of good user traffic. Note that the defense method has to provide near perfect isolation in the midst of a low-rate attack and, at the same time, have of low implementation cost.

The contributions of our work are:

- We provide a formal method of describing and generating a large family of low-rate attack traffic.
- We provide a distributed detection mechanism which uses the *dynamic time warping* (DTW) method to robustly and efficiently identify low-rate attacks.
- We provide a computationally efficient defense method to isolate legitimate traffic from malicious low-rate attack traffic.

The balance of the paper is organized as follows. In Section 2, we provide a formal model for describing and generating a large family of low-rate attack traffic. In Section 3, we present a distributed approach for detecting the existence of the attack traffic. We also show the robustness and accuracy of the proposed detection method. In Section 4, we present our defense mechanism. Experimental results are presented in Section 5 to illustrate the effectiveness of the defense mechanism. Related work is given in Section 6. Section 7 concludes.

2. Formal Description of Low-rate Attacks

Since the low-rate attack can appear in many different forms (as described below), we first provide a formal model of a low-rate TCP attack. Given the mathematical characterization, one can generate a family of low-rate attacks. We will then describe how one can extract *signatures* from the large family of attack flows.

2.1. Mathematical Model of Low-rate Attacks

A low-rate TCP attack is essentially a periodic burst which exploits the homogeneity of the minimum retransmission timeout (RTO) of TCP flows. Consider a router with capacity C (in bits/s). One form of attack is a periodic square wave as described in [7]. The period of the square wave is denoted by T, which is approximately one second so as to effectively force other TCP flows to enter the retransmission state. Within each period, the square wave has a magnitude of zero except for l units of time. During this time, the square wave has a magnitude of a normalized burst of R (note that in this work, the magnitude of the burst is *normalized* by the router's capacity C, therefore $R \in (0, 1]$). The average bandwidth of this periodic square wave is Rl/T. Again, the objective of the low-rate attack is that for a short duration l, the attack packets will fill up the buffer of a victim router so that packets of any TCP flows have to be discarded by the router. The packet loss will force most, if not all, TCP flows to enter the retransmission state. Also note that to be considered a low-rate TCP attack, the ratio of l/T has to be small. Otherwise, system administrators can easily detect an attack by its high traffic volume.

A general model of a low-rate TCP attack can be described by five parameters (T, l, R, S, N). The parameters T, l, and R have the same meaning as described above, S denotes the amount of time shift, starting from the initial measurement instant of the signal (i.e., $t = 0$) to the beginning of the attack pulse, and N denotes the amount of *background noise* (i.e., background traffic). The background noise is due to other UDP flows, which will not back off during congestion, or other TCP flows which are not in the retransmission state. Figure 1 illustrates an example of low-rate TCP attack traffic.

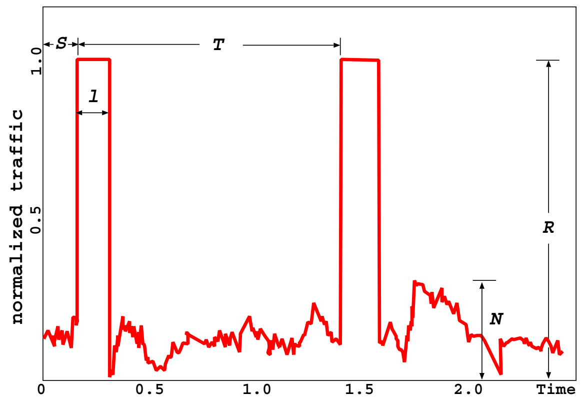


Figure 1. Low-rate TCP attack traffic with parameters (T, l, R, S, N).

Let us define the valid range of these five parameters.

- **Value of T:** As indicated in [7], the most effective value for the periodic low-rate attack is $T = 1$ second. In our study, we consider a larger range of T, which is $T \in [1.0, 1.5]$.
- **Value of l:** Assume that we have K TCP flows which are affected by the low-rate TCP attack. Let RTT_i represent the roundtrip time from the source i of a TCP flow to the victim router. To have an effective attack, the low-rate attack burst length should last long enough to keep the router's queue full for all RTT timescales.

Therefore, $l \geq \max_i\{RTT_i\}$, for $i = 1, 2, \ldots, K$. Since the aim of the low rate TCP attack is to avoid sending a high volume of traffic (and thus escape detection), the value of l cannot be too close to T. In our study, we have $\max_i\{RTT_i\} \leq l \leq \beta_1 T$, where $\beta_1 \leq 0.25$.

- **Value of R:** Since this is a normalized burst with respect to the router's capacity C, we have $R \in (0, 1]$.
- **Value of S:** The amount of time shift S, starting from the initial point of measurement (i.e., $t = 0$) to the beginning of the attack pulse, has a valid range of $0 \leq S \leq T - l$.
- **Value of N:** The amount of normalized background noise due to other UDP or TCP packets, it has a valid range of $0 \leq N \leq \beta_2 R$, where $\beta_2 \leq 0.5$.

Another important point to notice is that the low-rate TCP attack can be launched from either a single source, or multiple distributed sources. For the single source attack, it is easy to generate and it is effective when there is sufficient bandwidth between the attack source and the victim router. On the other hand, a distributed attack is more difficult to generate, since doing so requires time synchronization among the attack sources. In other words, the different attack sources, which have different propagation and transmission delays to the victim router, need to synchronize their attack traffic such that the aggregate traffic at the victim router forms a low-rate attack. There are at least two approaches for generating a distributed attack. In the first approach, each of the M attack sources generates a homogeneous and periodic attack waveform with a normalized burst size of $R \geq 1/M$. These flows will aggregate into a sufficient large burst at the victim router and force the affected TCP flows to back off. Another possible form of distributed attack, which has a lower synchronization requirement, requires each attack source to generate a large burst and for a longer period. For example, each of the M attack sources generates a homogeneous and periodic attack waveform with $T = M$ seconds and a normalized burst size of $R = 1$.

3. Distributed Detection

To defend against the family of low-rate TCP attacks, the first issue we have to address is how to perform effective *detection* in a computationally efficient manner. Unlike other intrusion detection or DDoS detection methods [8, 16], one cannot perform the detection at the victim site, say a web server denoted by $\mathcal{S}$. The reason is that the attack traffic will intrinsically "throttle" legitimate TCP flows destined for $\mathcal{S}$. Therefore, an attacker does not necessarily need to aim the attack at the victim site, but perhaps at a subset of upstream routers to $\mathcal{S}$ in order to throttle all the TCP flows passing through the routers.

Instead, we propose a distributed detection mechanism that is installed at a set of routers which are $k \geq 1$ hops away from the victim site. Each router needs to perform attack detection at the output port of packets being forwarded to the victim site $\mathcal{S}$. If a low-rate TCP attack is detected, the router needs to determine the input port(s) from which the attack traffic is being received. Detection will then be carried out at all these input ports of the affected router. If a low-rate attack is detected at an input port, say $\mathcal{P}$, then the affected router will push back the detection to all the upstream routers connected to the input $\mathcal{P}$. If the affected router cannot detect a low-rate attack at any of its input port, this means that the low-rate attack is being carried out in a distributed manner, and the defense mechanism to be discussed in Section 4 will be carried out. Note that the above distributed detection mechanism has several important features. They are:

- It pushes the detection of low-rate attacks as close as possible to the attack sources, and
- It is able to minimize damage to the legitimate TCP flows.

The overall detection mechanism can be described as follows:

Distributed Detection Mechanism
Let $\mathcal{R}$ be the deployment router. $\mathcal{P}_i$ is the set of input ports of $\mathcal{R}$, $\mathcal{P}_0$ is the output port $\mathcal{R}$ forwarding packets to the victim site $\mathcal{S}$.

```
1. $\mathcal{R}$ determines the existence of low-rate attack at $\mathcal{P}_0$;
2. If (low-rate attack exists) {
      determine the existence of low-rate attack at $\mathcal{P}_i$;
3.    If (attack exists for input port $\mathcal{P} \in \mathcal{P}_i$) {
4.       signals all upstream routers connected to
            $\mathcal{P}_i$ to perform distributed low-rate attack
               detection; }
5. execute the defense mechanism described in Sec. 4;
6  }
```

3.1. General Design of Detection

It is easy for an attacker to generate attack packets with spoofed header information (e.g., IP source address and type of transport protocol). There is no easy way to accurately differentiate low-rate TCP attack packets from legitimate packets. Instead, we must detect a low-rate TCP attack by matching current packet arrivals to given attack pattern signatures.

The detection mechanism is to be installed at a deployment router. It involves the following steps:

- *Statistical sampling of incoming traffic:* Traffic will be sampled and normalized using the transmission capacity of the network link.
- *Noise filtering:* Since other packets which arrive during the inactive period of a low-rate attack will also be included in the sampling process, one has to perform filtering before the feature extraction process.
- *Feature extraction:* We need to perform computationally efficient feature extraction that is resilient against time and space shifts.
- *Signature matching:* We need to compare extracted features of the incoming traffic with the signatures of low-rate TCP attack traffic.

3.2. Statistical Sampling of Incoming Traffic

Traffic needs to be periodically sampled at a constant rate. Each sample consists of an instantaneous throughput of the link interface. The rate of sampling should be frequent enough to record slight variations of the instantaneous throughput and, at the same time, should not put a heavy computational burden on the router. Note that statistical sampling can be easily achieved using standardized algorithms or off-the-shelf signal processing chips. The length of each sampling period, denoted by T_s, should also be properly chosen. In order to capture the periodicity of a low-rate TCP attack, the sampling period should be lower bounded by $T_S \geq 2T$. One should also put an upper bound on T_s. Note that a high value of T_s implies a higher storage cost, a higher computational cost for feature extraction at a later stage, and larger delay in detecting the attack. In our prototype, we have $T_S \leq 5$ seconds. Another technical issue we have to consider is *traffic normalization*. Since different link interfaces may have different line speeds, to facilitate feature extraction and matching at a later stage, the sampled traffic signal of a given link interface will be normalized by its line speed such that

$$\text{normalized throughput} = \frac{\text{sampled throughput}}{\text{maximum line speed}}.$$

3.3. Noise Filtering

The sampled input should be treated before the feature extraction process. Note that besides potential low-rate TCP attack packets, some other packets may also be included in the sampling process. These packets include:

- Packets that are forwarded to the same port but are not destined for the victim site $\mathcal{S}$.
- TCP packets – especially from flows with large RTTs – which can survive the low-rate TCP attack [7].
- UDP packets which will not back off in the face of low-rate attacks or network congestion in general.

These types of traffic have either higher frequencies or smaller magnitudes, as compared with the burst characteristics of a low-rate attack. To get a clean signal, a low-pass filter can be used to filter out the high frequencies and, at the same time, clamp all sampled signals to zero if they are less than or equal to a fraction β_2 of the peak value R. In our prototype, we set $\beta_2 = 0.5$.

3.4. Feature Extraction

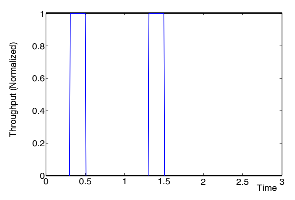
We use auto-correlation to extract the periodic signatures of an input signal. We use the auto-correlation measure not only because it is easy to calculate (e.g., for a sampled input of size n, the computational complexity is $\Theta(n^2)$), but one can also check the randomness or periodicity of a given signal in the presence of a time shift S.

Auto-correlation is calculated with unbiased internal normalization. The unbiased normalization is necessary if the input signal is a finite sequence. Consider an input signal with n values $(i_0, i_1, \cdots, i_{n-1})$ and all other $i_i = 0$. The unbiased normalized auto-correlation $A(k)$ can be calculated as follows:

$$A(k) = \frac{1}{n-k} \sum_{i=0}^{n-k+1} i_{i+k} i_i \qquad k = 0, ..., n-1. \qquad (1)$$

To illustrate, consider the following auto-correlation plots. Figure 2(a) shows the noise-filtered input signal with time shift $S = 0.3$ second and periodicity of $T = 1$ and $l = 0.2$ second, respectively. Note that this is the "classical" low-rate attack waveform. Figure 2(b) shows the corresponding auto-correlation plot. One important observation is that the *peak-to-peak* distance is 1, which captures the *period* of the input signal and that the auto-correlation plot is the *same*, independent of the time shift value S. Consider a more complicated attack waveform which is illustrated in Figure 3(a). In this attack, the time shift is $S = 0.5$, and the first period is $T = 1$ second. The first attack period has burst length $l_1 = 0.1$, while the second attack period has burst length $l_2 = 0.3$. The auto-correlation plot in Figure 3(b) reveals the existence of a period (i.e., the peak-to-peak distance in the auto-correlation plot) and that the bursts may have different durations.

Again, we extract the feature of auto-correlation plot from an input signal not only because it captures the periodicity property, but also because it eliminates the prob-


(a) Input sampled traffi c

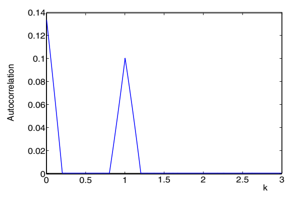
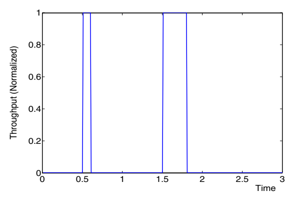

(b) Autocorrelation plot

Figure 2. Auto-correlation of input signal $T = 1, S = 0.2, l = 0.2, R = 1.0.$


(a) Input sampled traffi c

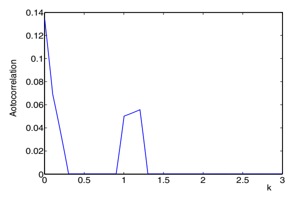

(b) Autocorrelation plot

Figure 3. Auto-correlation of input signal $S = 0.5, T = 1, l_1 = 0.1, l_2 = 0.3$**.**

lem of time shifting. The next question we need to address is how to compare the auto-correlation plot of an input signal with the auto-correlation plot (or signature) of a low-rate attack.

3.5. Pattern Matching via Dynamic Time Wrapping (DTW)

After the above steps, features are extracted from the sampled input. One now has to compare the *similarity* between the extracted features and the signatures of low-rate attack traffic, and decide whether there is an ongoing attack or not. An example signature of a low-rate attack is shown in Figure 2(b). If the auto-correlation plot of the sampled input is exactly the same as this signature, one can easily conclude the existence of a low-rate attack. However, not all auto-correlation plots of sampled inputs will match exactly the signature – for instance, consider the auto-correlation plot in Figure 3b). Therefore, one has to do more intricate processing so as to make an accurate decision.

The mechanism we adopt is called dynamic time warping (DTW) [6, 9]. It is a robust and computationally efficient method for comparing the similarity between a template signature and an input signal, even when the input signal is subjected to changes in time scale and magnitude.

The dynamic time warping algorithm can be described as follows. Suppose there are two time series, the template $\mathcal{S}$ and an input signal $\mathcal{I}$ of length n and m, respectively, where

$$\begin{aligned} \mathcal{S} &= s_1, s_2, s_3, ..., s_n, \quad \text{and} \\ \mathcal{I} &= i_1, i_2, i_3, ..., i_m. \end{aligned}$$

To compare the similarity of these two time series using DTW, we have to construct an n-by-m distance matrix $\mathcal{D}$ where $d(x, y)$ of $\mathcal{D}$ represents the Euclidean distance between the signature value s_x and the input signal value i_y, that is

$$d(x, y) = \| s_x - i_y \| \quad \text{for } 1 \le x \le n; 1 \le y \le m.$$

A warping path $\mathcal{W}$ is a contiguous set of matrix elements $\mathcal{D}$ that defines a mapping between the template $\mathcal{S}$ and input $\mathcal{I}$. The k^{th} element of $\mathcal{W}$ is defined as $w_k = d(i_k, j_k)$ where $\mathcal{W} = w_1, w_2, w_3, ..., w_k, ..., w_K$ and $\max(m, n) \le K \le m + n + 1$.

The construction of the warping path $\mathcal{W}$ is subject to the following constraints:

1. *Boundary constraint:* $w_1 = d(1, 1)$ and $w_K = d(n, m)$. This requires the warping path $\mathcal{W}$ to start and finish in diagonally opposite corner cells of the matrix $\mathcal{D}$.
2. *Continuity constraint:* Given $w_k = d(a, b)$ then $w_{k+1} = d(a', b')$, where $a' - a \le 1$ and $b' - b \le 1$. This restricts the allowable steps in the warping path to be adjacent cells.
3. *Monotonicity constraint:* Given $w_k = d(a, b)$ then $w_{k+1} = d(a', b')$ where $a' - a \ge 0$ and $b' - b \ge 0$. This restricts points in $\mathcal{W}$ to be monotonically spaced in time.

Note that there are many warping paths that satisfy the above constraints. However, we are interested in a path that minimizes the warping cost of $\mathcal{S}$ and $\mathcal{I}$. Formally:

$$DTW^*(\mathcal{S}, \mathcal{I}) = \min \left(\sqrt{\sum_{k=1}^{K} w_k} \right). \qquad (2)$$

In other words, the lower the value of $DTW^*(\mathcal{S}, \mathcal{I})$, the higher the similarity degree of the input string $\mathcal{I}$ as compared with the signature $\mathcal{S}$. The minimum cost warping path can be found using a *dynamic programming* approach. That is, we construct a matrix γ with dimension of n-by-m.

The entry $\gamma(x, y)$ in cell (x, y) defines the *cumulative distance* of the warping path $\mathcal{W}$ from position $(1, 1)$ to positive (x, y). The minimum of the cumulative distances of the adjacent elements $\gamma(x, y)$ is:

$$\begin{aligned}\gamma(x, y) \quad = \quad & d(x, y) + \min\{\gamma(x-1, y-1), \\ & \gamma(x-1, y), \gamma(x, y-1)\}\end{aligned}$$

where $1 \leq x \leq n; 1 \leq y \leq m$. At each step of calculating the value of $\gamma(x, y)$, if $\min\{\gamma(x-1, y-1), \gamma(x-1, y), \gamma(x, y-1)\} = \gamma(x-1, y)$ or $\gamma(x, y-1)$, it means that there is one point in the input signal $\mathcal{I}$ that has been matched twice to the template $\mathcal{S}$, or there is one point in $\mathcal{S}$ that has been matched twice to $\mathcal{I}$. Although this scenario is common in other applications like speech recognition and can be viewed as the homology of the input and the template, the matchings cannot be regarded as identical attack traffic patterns. As a result, we make a modification to the original DTW algorithm that adds some adaptive penalty v and h for this kind of vertical or horizontal "movement" in the warping path so as to evaluate the similarity while still distinguish the slight difference. However, the value of the penalty should not be too large. Otherwise, it will badly increase the DTW value of similar attacks, thus increasing the intersection of normal traffic and attack traffic, which in turn leads to high false positive and false negative rates. As a result, we choose the function of calculating the cumulative distances in our system to be:

$$\begin{aligned}\gamma(x, y) \quad = \quad & \| s_x - i_y \| + \min\{\gamma(x-1, y-1), \\ & \gamma(x-1, y)+v, \gamma(x, y-1)+h\}\,. \qquad (3)\end{aligned}$$

After creating the matrix γ, the value $\gamma(n, m)$ is the minimum cumulative distances of the DTW between the template $\mathcal{S}$ and the input $\mathcal{I}$ and it is the solution to Eq. (2). In general, a lower value of DTW implies that the input signal $\mathcal{I}$ is more similar to the signature $\mathcal{S}$.

Additionally, the process of generating the matrix γ by using *dynamic programming* to find the minimum DTW value is carried out as follows. The matrix is built column by column, from left to right and bottom up for each column. For an input size of m and template size of n, the computational complexity of the DTW is $\Theta(mn)$, which is acceptable in practice.

3.6. Robustness and Accuracy of DTW

In this section, we consider the robustness and accuracy of using DTW in detecting low rate TCP attacks. The experimental setup is as follows. For the low-rate attack signature, we consider $T = 1.2$ sec, $l = 0.2$ sec, $R = 1.0$, and $S = N = 0$. For the input traffic, we sample 100 times per second and the sampling duration is three seconds per detection. We set the noise filter threshold $\beta_2 = 0.5$ such that all background traffic that is less than or equal to 50% of the maximum link capacity C will be clamped to zero. Under the DTW, we set the penalty value $v = h = 0.01$. We consider three types of input traffic:

- *Square burst:* A periodic signal with a single burst of length l within a period of duration T.
- *Step-like burst:* A periodic signal with two adjacent bursts of length l within a period T. The first burst strength is R while the second burst strength is equal to the link capacity C.
- *General burst:* A periodic signal which is generated by a sine wave with period T with added random noise.

DTW values for low-rate attack: To generate the input traffic, the period T is uniformly distributed within $[1, 1.5]$. The burst length is uniformly distributed within $(0, 0.5]$, provided that $l/T \leq \beta_1 = 0.25$. The background noise is uniformly distributed in $[0, 0.5]$, the time shift S is uniformly distributed in $[0, T]$, and the magnitude of the burst is set to $R = 1$. We generate 1000 samples for each of the three types of input traffic discussed above. The results are illustrated in Table 1. From the results, we find that a large family of low rate attack has a DTW value that is less than or equal to 60 and that close to 90% has a value less than or equal to 10.

Values of DTW	Square Burst	General Burst	Step-like Burst
Max DTW	39.48	29.89	57.10
Min DTW	0.25	0.22	0.49
Mean DTW	5.73	5.11	7.97
Standard Deviation	6.93	4.61	11.39

Table 1. DTW values for attack traffic.

DTW values for normal traffic: The detection mechanism must distinguish normal traffic from the attack traffic so as to avoid false positives and false negatives. Therefore, it is desirable for the minimum DTW value of the normal traffic to be larger than the maximum DTW value of any attack traffic for clear distinction between the two types of traffic.

We carry out the following experiment with normal traffic. Based on our assumption before, if there is no low-rate attack, the TCP flows will not back off, and all the traffic including TCP and UDP packets will be processed by a router as usual. We assume that the normal traffic consists of a major constant throughput with some Gaussian noise. I.e., normal traffic $= C_1 +$ random$[0, N]$, where $C_1 \in [0.5, 1]$ and $N \in [0, 0.5]$. We vary the value of C_1 by a step size of

0.05 and for each value of C_1, we generate 100 different values of N. The results are shown in Table 2. As we can observe, the minimum DTW value for normal traffic is above 60 and we can use it as the threshold of detecting the existence of a low-rate TCP attack.

	Normal traffic
Max DTW	286.60
Min DTW	62.51
Mean DTW	205.24
Standard Deviation	66.63

Table 2. DTW values for normal traffic.

Probability density function of DTW for normal and attack traffic: To clearly illustrate the robustness of the proposed detection mechanism, we carry out an experiment to evaluate the DTW values of both attack and normal flows. For the attack flows, we consider three types of traffic as discussed above. The period T is randomly chosen from $[1, 1.5]$, the burst length l is randomly chosen from $(0, 0.5]$, and the burst magnitude is of full link bandwidth $R = 1$. We generate around around 400 attack flows. For the normal traffic, we generate around 2500 flows with C_1 uniformly distributed between $[0.5, 1]$ and the noise N uniformly chosen between $[0, 0.5]$. Figure 4 illustrates the *probability density function* of the DTW values for the attack and normal flows, respectively. From the figure, we observe that there is a clear determination point (i.e., $DTW = 60$) for differentiating the normal traffic from the attack traffic.

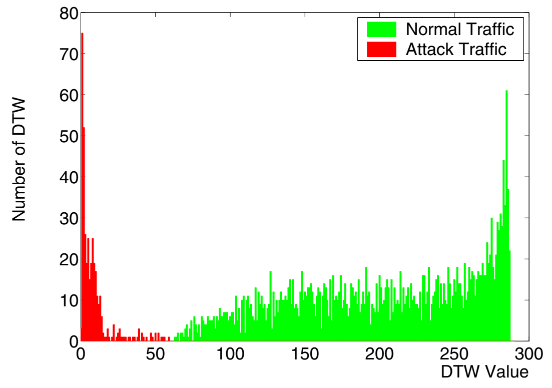


Figure 4. Probability density functions of DTW for attack and normal traffic.

4. Low-Rate Attack Defense Mechanism

As we discussed in the distributed detection mechanism in Section 3, a deployment router $\mathcal{R}$ first determines the existence of a low-rate attack at an output port $\mathcal{P}_o$ which $\mathcal{R}$ uses to forward packets to a victim site. When a low-rate attack is detected, $\mathcal{R}$ will then determine the input port(s) on which the attack packets arrive. If attack packets arrive on an input port, say $\mathcal{P}$, then $\mathcal{R}$ needs to signal all the upstream routers which are directly connected to $\mathcal{P}$ to execute the distributed detection algorithm, and then an accompanying defense mechanism to neutralize the effects of the attack. Our motivation is to move the attack detection and defense as close to the source as possible. This way, we minimize the number of TCP flows that will be affected by an attack.

When $\mathcal{R}$ discovers the existence of a low-rate attack at its output port, but *cannot* identify the attack at any of its input ports, the low-rate attack is being carried out in a distributed manner. For example, the attacker sends a short burst to each of the input ports such that the short bursts will aggregate to a low-rate attack burst an an output port of $\mathcal{R}$. In this case, $\mathcal{R}$ will need to properly allocate the output port bandwidth so as to minimize the damage to TCP flows going through the port.

In our work, we use the deficit round robin (DRR) algorithm to provide bandwidth allocation and protection between flows. The motivation for using DRR is its near perfect isolation of ill-behaved sources at a very low implementation cost. At the same time, packets from different classes can have different sizes, and fairness can still be achieved [11]. In general, the processing cost of DRR is $O(1)$ per packet. In our case, instead of classifying packets based on flow information, we classify packets by the input port of $\mathcal{R}$ on which they arrive. Let $\mathcal{P}_i$ denote the set of input ports of the router $\mathcal{R}$ and $|\mathcal{P}_i|$ denote the number of input ports. We have $|\mathcal{P}_i|$ classes. Packets arriving on input port i and being forwarded to output port $\mathcal{P}_0$ will be classified to class i, where $i = 1, \ldots, |\mathcal{P}_i|$.

DRR assigns a quantum of service to each class in each round and attempts to serve packets from each class on a per round basis. Each class has a deficit counter, denoted by deficit_counter[i] and to zero, for $i = 1, \ldots, |\mathcal{P}_i|$. At the beginning of a round, deficit counters of all the nonempty classes will be increased by the quantum value. A packet from class i will be served if the size of the packet is less than or equal to the value in deficit_counter[i]. When a packet is transmitted from class i, its deficit value will be adjusted by deficit_count[i] -= packet's size. If there is no packet in class i, then we reset the deficit counter as deficit_count[i] = 0. Note that the deficits of the previous rounds get carried over to the next round. A deficit is reset to zero only when there is no outstanding packet in the corresponding class.

5. Experiments

In this section, we report experiments using ns-2 to determine the effectiveness of the proposed defense mechanism. Because of space limit, we only present the results for TCP Tahoe. Please refer to our technical report [14] for results with TCP Reno and NewReno.

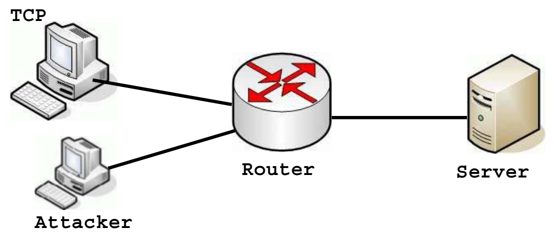


Figure 5. Single low-rate attack and single TCP flow.

Experiment 1 (Single TCP flow vs single source attack): The first experiment is shown in Figure 5. We consider a single low-rate TCP attack and a single TCP flow going through the same router. The latency of each link is 5 ms and the capacity of each link is 5 Mb/s. The low-rate attack is a square burst with $T = 1.0$ sec, burst length $l = 0.2$ sec, burst rate of 5 Mb/s, and $R = 1$. The low-rate attack uses UDP with packet size of 100 bytes. The packet size of the TCP flow is 500 bytes. Under DRR, we set the quantum size of each round to be 500 bytes and the buffer size to 5000 bytes. The result is illustrated in Table 3. Note that without the defense mechanism, the router simply uses conventional scheduling (i.e., drop tail and FCFS) to handle the packets. Also, we observe that the TCP flow can only utilize around 4% of the link's bandwidth. On the other hand, when one uses DRR, we observe an improvement in the TCP's throughput from 224.37 kb/s to 3.402 Mb/s, or an improvement from 4.49% to 68.04% of the link capacity. The result shows the effectiveness of the defense mechanism in protecting the TCP flows from the malicious attacking flows.

Experiment 2 (Multiple TCP flows vs single source attack): The second experiment is shown in Figure 6. We consider a single low-rate TCP attack and 8 TCP flows going through the same router. Parameters are the same as in Experiment 1 except that the link latency of the i^{th} TCP flow is equal to $5i$ ms, for $i \in \{1, \ldots, 8\}$. The buffer size of the DRR-enabled router is 12.5 kbytes. The results are illustrated in Table 4. Again, using conventional drop tail scheduling, the total TCP bandwidth is only around 11% of the link's bandwidth. When one uses DRR, we can improve the throughput of all the TCP flows from 583.96 kb/s to 4.173 Mb/s, or an improvement from 11.68% to 83.45% of the link capacity. This shows the effectiveness of the defense mechanism.

	TCP		**Attack flow**	
	throughput (Kbps)	% of capacity	throughput (Kbps)	% of capacity
Drop tail	224.37	4.49%	1016.5	20.33%
DRR	3402.10	68.04%	780.39	15.61%

Table 3. Result for low-rate attack on single TCP flow using Tahoe.

Experiment 3 (Multiple TCP flows vs synchronized distributed low-rate attack): The third experiment is shown in Figure 7. We consider a distributed low-rate TCP attack and 8 TCP flows going through the same router. Parameters are the same as in Experiment 2 except that we replace a single attacker by three distributed attackers. Each attacker sends a periodic attack burst every $T = 3.0$ sec. The i^{th} attacker sends a burst with $R = 1$ during the i^{th} sub-period so that the converged attack becomes a low-rate attack with period $T = 1.0$ sec. The results are illustrated in Table 5. One can observe that with DRR, we can improve the throughput of all the TCP flows from 469.67 kb/s to 4.296 Mb/s, or an improvement from 9.39% to 85.94% of the link capacity.

Experiment 4 (Network model of low-rate attack vs Multiple TCP flows): The fourth experiment is shown in Figure 8. The transmission bandwidth of all the links is 5 Mb/s and the propagation delay is 5 ms. The attacker is lo-

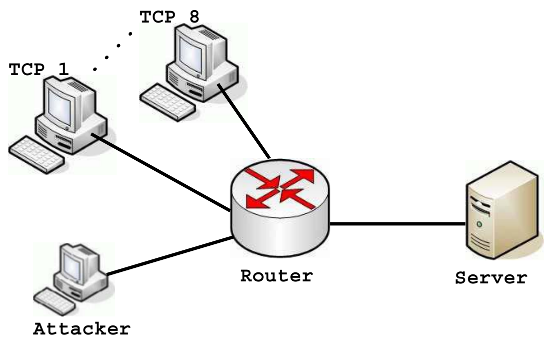


Figure 6. Single low-rate attack and Multiple TCP flows.

	Drop Tail		DRR	
	throughput (Kbps)	% of capacity	throughput (Kbps)	% of capacity
Attack flow	928.76	18.58%	343.09	6.86%
TCP 1	8.71	0.17%	965.91	19.32%
TCP 2	210.77	4.22%	645.79	12.92%
TCP 3	4.75	0.10%	629.15	12.58%
TCP 4	11.09	0.22%	618.05	12.36%
TCP 5	5.54	0.11%	468.3	9.37%
TCP 6	267.82	5.36%	356.57	7.13%
TCP 7	72.11	1.44%	293.97	5.88%
TCP 8	3.17	0.06%	194.93	3.90%
Total TCP	**583.96**	**11.68%**	**4172.67**	**83.45%**

Table 4. Results for single low-rate attack on multiple TCP flows using Tahoe.

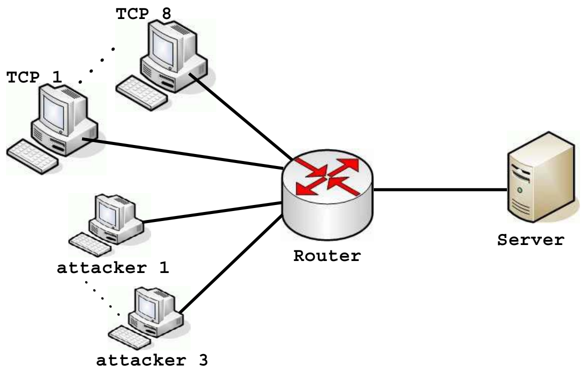


Figure 7. Distributed low-rate attack and Multiple TCP flows.

	Drop Tail		DRR	
	throughput (Kbps)	% of capacity	throughput (Kbps)	% of capacity
Attack 1	355.30	7.11%	120.98	2.42%
Attack 2	305.91	6.12%	116.34	2.33%
Attack 3	309.63	6.19%	111.17	2.22%
TCP 1	218.47	4.37%	818.58	16.37%
TCP 2	16.64	0.33%	748.80	14.98%
TCP 3	9.13	0.18%	748.80	14.98%
TCP 4	6.98	0.14%	489.54	9.79%
TCP 5	6.44	0.13%	436.93	8.74%
TCP 6	203.97	4.08%	432.64	8.65%
TCP 7	5.36	0.11%	314.55	6.29%
TCP 8	2.68	0.05%	307.03	6.14%
Total attack	**970.84**	**19.42%**	**348.49**	**6.97%**
Total TCP	**469.67**	**9.39%**	**4296.87**	**85.94%**

Table 5. Result for Synchronized Distribute Low-rate Attack to Multiple TCP Flows using Tahoe.

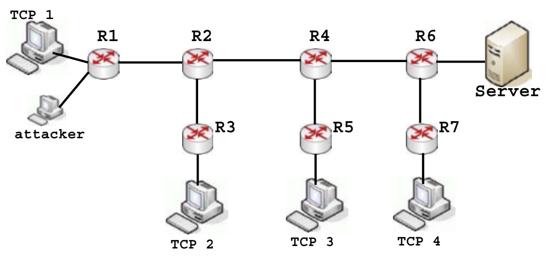


Figure 8. Network model of Low-rate attack and Multiple TCP flows .

cated at router R_1. It sends a periodic attack with $T = 1$ sec, $l = 0.2$ sec, and $R = 1$. There are four TCP flows, TCP 1 is attached to R_1, TCP 2 is attach to R_3, TCP 3 is attached to R_5, and TCP 4 is attached to R_7. All of them try to upload files to the server. Table 6 shows the throughput of attack and TCP flows when no defense mechanism is deployed (under the drop tail column), and the throughput of the various flows when DRR is employed at the different routers. The table shows that enabling DRR at the different routers will achieve different TCP throughputs. In particular, when DRR is enabled in R_6 only, the bandwidth of TCP 4 is approximately equal to the sum of bandwidth of all the upstream flows (i.e., TCP 1 to TCP 3, and the attack traffic). Under the proposed distributed defense mechanism, routers R_1, R_2, R_4, and R_6 will discover the presence of the low-rate attack and therefore enable DRR scheduling. One can observe that fairness is achieved in which all the TCP flows achieve approximately the same amount of bandwidth, and they are protected and isolated from the malicious attack traffic.

6. Related Work

Network denial of service is a well recognized problem of importance and urgency (e.g., [2, 3]). Various detection and defense approaches have targeted the control of high-

	Drop tail	**DRR on** R_6	**DRR on** R_6, R_4	**DRR on** R_6, R_4 R_2	**DRR on** R_6, R_4 R_2, R_1
	ρ(kbps)	ρ(kbps)	ρ(kbps)	ρ(kbps)	ρ(kbps)
Attack	640.00	561.00	453.00	419.00	404.00
TCP 1	386.00	358.00	311.00	314.00	778.00
TCP 2	264.00	329.00	282.00	874.00	763.00
TCP 3	324.00	251.00	1,245.00	924.00	788.00
TCP 4	425.00	1,719.00	1,154.00	966.00	765.00
Total TCP	1,399.00	2,657.00	2,992.00	3,078.00	3,094.00

Table 6. Througput of various TCP flows when different routers deploy the defense mechanism.

rate attacks [1, 8, 10, 13]. The low-rate TCP attack is first described by Kuzmanovic and Knightly [7], who characterize the attack and point out important challenges of detection and defense. Since low-rate attacks are most effective when the retransmission attempts by TCP sources are synchronized following a congestion, randomizing the TCP RTO is an intuitive solution approach and has been shown to be effective in [15]. However, randomizing the RTO requires widespread updates of existing end user software and may reduce the performance of TCP under non-attack conditions [4]. In comparison, we seek a solution at the router level. Other DDoS solutions at this level, but with a different focus than ours, include IP traceback [10], hash-based IP traceback for low volume traffic [12], pushback rate limit [8, 16], and the eXplicit Control Protocol (XCP) [5].

7. Conclusion

This paper presents an efficient mechanism to dynamically detect and defend against low-rate TCP attacks. We present a formal model to describe a large family of low-rate TCP attack patterns. We then propose a distributed detection mechanism which uses the dynamic time warping algorithm to compare the feature of the sampled input with the signature of the low-rate attack. We show that the detection mechanism is robust and accurate in identifying the existence of low-rate attacks. When a low-rate attack is present, we use a pushback mechanism to identify the attack as close to the attack source as possible. The reason of this pushback is to minimize the number of affected TCP flows. We show that one can use the deficit round-robin algorithm to protect the TCP flows and isolate them from the attack traffic. Extensive simulation experiments are reported to quantify the robustness and accuracy of the proposed detection and defense mechanisms.

References

[1] Blackhole route server and tracking traffi c on an IP network. http://www.secsup.org/Tracking.

[2] TCP SYN fboding and IP spoofi ng attacks. CERT Advisory CA-96.21. available at http://www.cert.org/.

[3] Smurf IP denial-of-service attacks. CERT Advisory CA-1998-01, January 1998. avaliable at www.cert.org/advisories/CA-98.01.html.

[4] M. Allman and V. Paxson. On estimating end-to-end network path properties. In *Proc. ACM SIGCOMM*, Vancouver, Canada, September 1999.

[5] D. Katabi, M. Handley, and C. Rohrs. Congestion control for high bandwidth-delay product networks. In *Proc. ACM SIGCOMM*, Pittsburgh, PA, August 2002.

[6] E. Keogh. Exact indexing of dynamic time warping. In *Proc. of the 28th VLDB Conference*, China, Aug. 2002.

[7] A. Kuzmanovic and E. Knightly. Low-rate TCP-targeted denial of service attacks. In *Proc. ACM SIGCOMM*, Karlsruhe, Germany, August 2003.

[8] R. Mahajan, S. Floyd, and D. Wetherall. Controlling high-bandwidth fbws at the congested router. In *Proc. ICNP*, Riverside, CA, November 2001.

[9] H. Sakoe and H. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Trans. Acoustics, Speech, and Signal Proc.*, 26(1), Feb. 1978.

[10] S. Savage, D. Wetherall, A. Karlin, and T. Anderson. Practical network support for IP traceback. In *Proc. ACM SIGCOMM*, Stockholm, Sweden, August 2000.

[11] M. Shreedhar and G. Varghese. Effi cient fair queueing using defi cit round robin. *IEEE/ACM Transactions on Networking.*, 4(3), June 1996.

[12] A. C. Snoeren, C. Partridge, L. A. Sanchez, C. E. Jones, F. Tchakountio, S. T. Kent, and W. T. Strayer. Hash-based ip traceback. In *ACM SIGCOMM*, San Diego, August 2001.

[13] D. Song and A. Perrig. Advanced and authenticated techniques for IP traceback. In *Proc. IEEE Infocom*, Anchorage, Alaska, 2001.

[14] H. Sun, J. C. Lui, and D. K. Yau. Defending against low-rate tcp attack: Dynamic detection and protection. In *CS&E Tech. Report, CUHK*, 2004.

[15] G. Yang, M. Gerla, and M. Y. Sanadidi. Defense against low-rate TCP-targeted denial-of-service attacks. In *Proc. ISCC*, Alexandria, Egypt, 2004.

[16] D. K. Yau, J. C. Lui, F. Liang, and Y. Yeung. Defending against distributed denial-of-service attacks with max-min fair server-centric router throttles. *Accepted for publication in IEEE/ACM Transactions on Networking, 13(1), February 2005.*