Distributed Mechanism in Detecting and Defending Against the Low-rate TCP Attack[*]

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— **In this paper, we consider a distributed mechanism to detect and to defend against the low-rate TCP attack. The low-rate TCP attack is a recently discovered attack. In essence, it is a periodic short burst that exploits the homogeneity of the minimum retransmission timeout (RTO) of TCP flows and forces all affected TCP flows to backoff and enter the retransmission timeout state. When these affected TCP flows timeout and retransmit their packets, the low-rate attack will again send a short burst to force these affected TCP flows to enter RTO again. Therefore these affected TCP flows may be entitled to zero or very low transmission bandwidth. This sort of attack is difficult to identify due to a large family of attack patterns. We propose a distributed detection mechanism to identify the low-rate attack. In particular, we use the "*dynamic time warping*" approach to robustly and accurately identify the existence of the low-rate attack. Once the attack is detected, we use a fair resource allocation mechanism to schedule all packets so that (1) the number of affected TCP flow is minimized and, (2) provide sufficient resource protection to those affected TCP flows. Experiments are carried out to quantify the robustness and accuracy of the proposed distributed detection mechanism. In particular, one can achieve a very low false positive/negative when compare to legitimate Internet traffic. Our experiments also illustrate the the efficiency of the defense mechanism across different attack patterns and network topologies.**

I. Introduction

The TCP protocol provides the reliable data delivery and simplifies application design and is being used in many network applications including file transfers, e-commerce, and web HTTP access. In general, designing a reliable protocol for many heterogeneous users sharing an unreliable network is challenging since it involves many subtle issues. For example, under severe network congestion, TCP requires sources to reduce their congestion window to one packet and wait for a retransmission timeout (RTO) before attempting to resend any packet. If there is further packet loss, the RTO is doubled after each subsequent loss. The purpose of using the RTO is to ensure that TCP sources will give the network sufficient time to recover from a network congestion event. In [1], authors recommend a lower bound of one second for its value in order to achieve near-optimal network throughput.

Although the use of RTO in the TCP protocol stack can reduce and relieve the event of network congestion, this feature can also be exploited by a malicious user to create a denial-of-service attack. Recently, authors in [2] 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. Due to the packet loss event, a legitimate (or well behaved) 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 the TCP's retransmission, then little or no real data packet can get through the router. 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 average bandwidth of the attack traffic can be quite low. The form of low-rate attack raises serious concern because it is significantly more difficult to detect than more traditional brute force, flooding based DDoS attacks. Existing rate-limiting approaches [3], [4], for example, are designed to control aggressive or flooding-based attackers only.

In this paper, we propose a distributed mechanism to detect against such low-rate TCP attacks. Because TCP is widely implemented and deployed, a proposal which requires changes to existing TCP protocol stack will incur a widespread modification of users' software and therefore this type of proposal 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 legitimate TCP users.

For detecting the low-rate attack, because an attacker's primary objective is to ensure the periodic overflow of a router's buffer, a basic signature of an attack traffic will then be intermittent short bursts of high rate traffic in between periods of little or no activity (characterized by, say, a periodic square

[*]The preliminary version of this paper appeared in the International Conference of Network Protocols (ICNP) 2004, Berlin, Germany.

wave). In practice, however, attack traffic can deviate from this basic attack 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 attack traffic does. Therefore, it is essential to develop detection algorithms that are both robust to traffic distortions and at the same time, computationally efficient to execute.

Once a low-rate 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 a near perfect isolation in the midst of low-rate attack and at the same time, has to have the property of low implementation cost.

The contribution of our work is:

- Provide a formal model to describe and to generate a large family of low-rate attack traffic.
- Provide a distributed detection mechanism which uses the *"dynamic time warping"* (DTW) approach to robustly and efficiently identify low-rate attack. We will show that the proposed detection mechanism has a very low false positive/negative, when comparing a low-rate attack with a legitimate traffic.
- Provide a computationally efficient defense method to isolate legitimate traffic from the ill-behaved low-rate attack.

The outline of this paper is as follows. In Section II, we provide a formal mathematical model to describe and generate a large family of low-rate attack. In Section III, we present the distributed mechanism of detecting the existence of the low-rate TCP attack. We also show the robustness and accuracy of the propose detection method when comparing an attack traffic with legitimate Internet traffic. In Section IV, we present the defense mechanism and its properties. Experiments are presented in Section V to illustrate the effectiveness of the defense scheme. Related work is given in Section VI and Section VII concludes.

II. Formal Description of Low-rate TCP Attacks

Because the low-rate attack can appear in many different forms (as describe below), let us first provide a formal model in describing a low-rate TCP attack. Given this mathematical description, one can generate a large family of low-rate attacks. We then proceed to describe how one can extract "*signatures*" from this large family of low-rate TCP attack flows.

A. Mathematical Model of Low-rate TCP 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 (with unit of bps), one form of attack is a periodic square wave as reported in [2]. The period of this square wave is denoted by T, which is approximately one second so as to effectively forcing other TCP flows to enter the retransmission period. 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 $R \in (0, 1]$. Although it is possible that in reality R may exceed 1, in our model we mainly focus on the range $(0, 1]$, since when $R > 1$, it will clearly cause packet loss and can be treated as the same class with $R = 1$. The average normalized 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 victimized router so that packets of any TCP flows have to be discarded by the router and forcing most, if not all TCP flows to enter the retransmission state. Also note that to be considered as a low-rate TCP attack, the ratio of l/T has to be small or else system administrators can easily detect a high volume attack.

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 denote the amount of time-shift, starting from the initial measurement instant of the signal (e.g., $t = 0$) to the beginning of the attack pulse, while N denote the amount of *background noise* or *traffic*. The background noise is due to other UDP flows, which will not backoff in the midst of congestion, or other TCP flows which are not in the retransmission period. Figure 1 illustrates an example of low-rate TCP attack traffic.

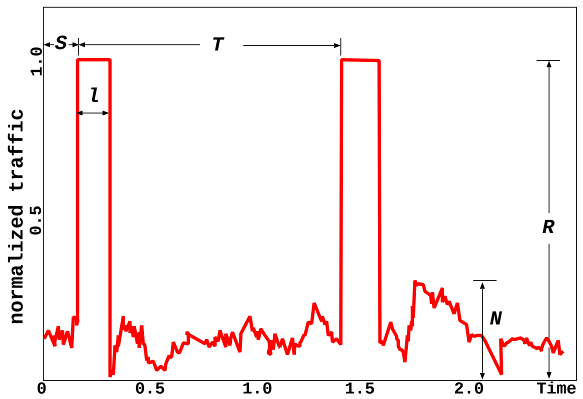


Fig. 1. Low-rate TCP attack traffic with parameters (T, l, R, S, N).

Let us define the valid range of these five parameters.

- **Values for T:** As indicated in [2], 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 $T \in [1.0, 1.5]$.

- **Values for l:** Assume that we have K TCP flows which are affected by the low-rate TCP attack. Let RTT_i represents the round-trip time from the source i of the TCP flow to the victimized 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 so as to avoid easy detection, the value of l cannot be very close to T. In our study, we have $\max_i\{RTT_i\} \leq l \leq \beta_1 T$ where $\beta_1 \leq 0.3$.
- **Values for 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 (e.g., $t = 0$) 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 $\beta_2 \leq 0.5$. Note that background noise is a general assumption, which exists most of the time in realtime signal processing. Note that adding a background noise to the model makes the detection a more challenging task. Yet, in realistic situation, noise is always present in the sampled traffic.

B. Other forms of Low-rate TCP Attacks

Based on the mathematics model above, more general attack wave form can be generated. For example, the attack traffic can be of the form of sine wave and an attacker can also generate different burst patterns within each sub-period. Figure 2 illustrates an instance of the general attack traffic which has three attack sub-periods, each sub-period has a different attack characteristic. The first sub-period has $T_1 = 0.8$ sec and $l_1 = 0.1$ sec, the second sub-period has $T_2 = 1.2$ sec and $l_2 = 0.3$ sec while the last sub-period has $T_3 = 1.0$ sec and $l = 0.2$ sec. The generality of attack waveforms makes it difficult and challenging to characterize, detect and defend the low-rate TCP attack.

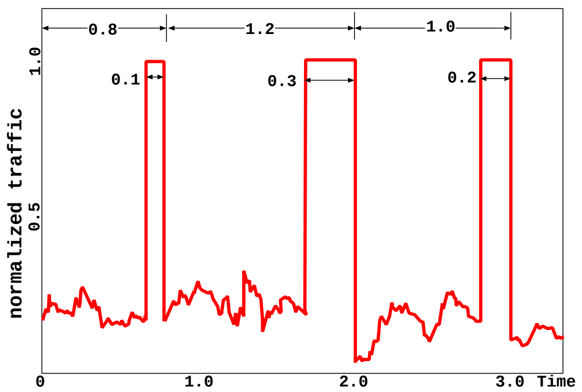


Fig. 2. General attack traffic with a varying pattern within each sub-period.

Another important point that is worth mentioning is that the low-rate TCP attack can be launched by either a single source, or by multiple distributed sources. For the single source attack, it is easy to generate and it is effective when there is sufficient bandwidth along the path between the attack source and the victimized router. For the distributed low-rate TCP attack, it is also possible to synchronize attacks over independent sources on the Internet, since jitter on the Internet is usually small, and it is on the order of 1RTT <100ms. However, compared with single source attack, issues concerning different propagation and transmission delays to the victimized router still need to be addressed. Thus, more effort is needed to generate a distributed attack. There are at least two approaches to generate a distributed attack. For 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 converge into a sufficient large burst at the victimized router and force all affected TCP flows to backoff. Another possible form of distributed attack, which has a lower synchronization requirement, is that each attack source generates a large burst but 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 $R = 1$. This kind of attack is illustrated in Figure 3 for three distributed attackers. The i^{th} attack source sends the attack burst at the i^{th} attacking sub-period and keeps silent for the remaining sub-periods. The converged attack traffic is illustrated in Figure 3(d).

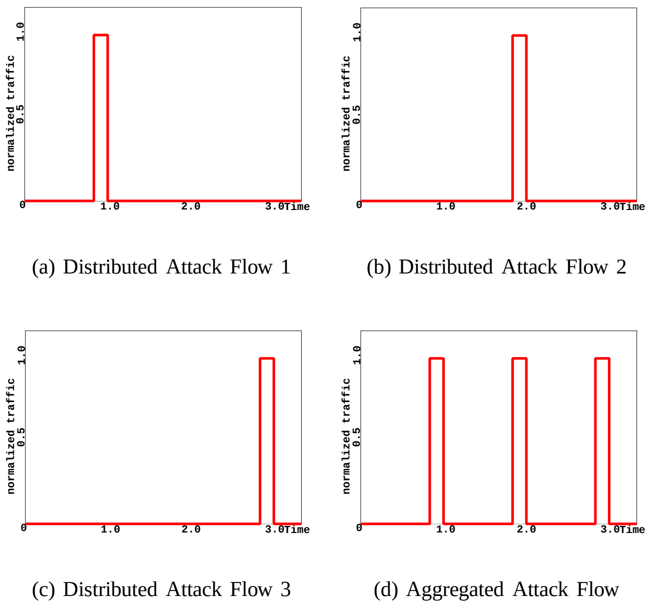


Fig. 3. Distributed low-rate attack with period $T = 3$ and $M = 3$ attack sources.

III. Distributed Detection Mechanism

Before we discuss how to defend against this family of low-rate TCP attacks, the first issue we need to address is how to perform an effective *detection* and that the detection method has to be computational efficient. Unlike other intrusion detection or DDoS detection methods [3] [4], one cannot simply install the detection mechanism at the victim site, say $\mathcal{S}$ (i.e., a web server). The reasons are as follows: First, to install the detection mechanism at the victim site, status of thousands of incoming TCP flows need to be monitored which maybe a burden to the server. More importantly, the low-rate TCP attack has the intrinsic characteristic to throttle legitimate TCP traffics at the victim site $\mathcal{S}$. Therefore, an attacker does not necessarily need to aim the attack at the victim site, but rather at a "subset of upstream routers" of $\mathcal{S}$ so as to throttle all TCP flows passing through these routers. Thus installing a detection mechanism at the victim will be ineffective since it provides no information for the victim site to determine where the attack is occurring, or the attack traffic is originated from which part of the network. As a result, any detection method installed at the victim site may not be very effective because the victim site only may not even detect the existence of attack. Instead, the victim site may think that only few users are interested to access information from the site $\mathcal{S}$ if is under the low-rate attack.

In this work, 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 the low-rate TCP attack detection on the *output port* that is forwarding packets to the victim site $\mathcal{S}$. If a low-rate TCP attack is detected, then the router needs to determine which input port(s) the low-rate attack is coming from. Detection will then be carried out on all these input ports of the affected router. If a low-rate attack is detected on an input port, say $\mathcal{P}$, then the affected router will push back the detection to all upstream routers that are connected to the input $\mathcal{P}$. If the affected router cannot detect any low-rate attack on any of its input port, this implies that the low-rate attack is carried out in a distributed manner, then the defense mechanism (which we will discuss in Section IV) will be carried out. Note that there are several important features of using the above distributed detection mechanism. They are:

- Detection is carried out from the output port to the input ports.
- Pushing the detection of low-rate attacks as close as possible to the attack sources so as to minimize the damage to other legitimate TCP flows when adopting the defense mechanism (Section IV).

The overall detection mechanism is as follows:

Distributed Detection Mechanism
Let $\mathcal{R}$ be the enabled router. $\mathcal{P}_i$ is the set of input port of $\mathcal{R}$, $\mathcal{P}_o$ is the output port $\mathcal{R}$ uses to forward packet to the victim site $\mathcal{S}$.

```
1. $\mathcal{R}$ determines the existence of low-rate attack on $\mathcal{P}_o$;
2. **If** (low-rate attacks exist ) {
       determine the existence of low-rate attack on $\mathcal{P}_i$;
3.     **If** (attack exits 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.     }
6. execute the defense mechanism describes in Sec. IV;
7 }
```

Let us describe in detail about the low-rate attack detection algorithm.

A. General Design of Low-rate Attack Detection

Because attack packets can be easily generated, all information in the packets' header can be spoofed, e.g., IP source addresses and types of transport protocol used, and there is no easy way to accurately differentiate low-rate TCP attacking packets from legitimate packets. The proper approach for the low-rate TCP attack detection is to compare the incoming traffic with attack pattern signatures.

The detection mechanism will be installed at enabled routers and the detection mechanism involves the following steps.

- **Statistical sampling of incoming traffic:** traffic will be sampled and normalized based on the transmission capacity of the link/port.
- **Noise filtering:** since other packets which arrive during the non-active period of the low-rate attack will also be included in the sampling process, therefore, one needs to perform filtering before the feature extraction process.
- **Feature extraction:** perform a computationally efficient feature extraction that is immune to time and space shift of the input signals.
- **Signatures comparison:** compare the extracted features of the incoming traffic with the signature of the low-rate TCP attack.

In the following, let us describe in detail the individual step involves in the distributed detection mechanism.

B. Statistical Sampling of Incoming Traffic

The router needs to periodically sample the incoming traffic at a constant rate. Note that each sample consists of a record of throughput of the link interface. The record of throughput is the measured throughput between two sample points. The rate of sampling should be frequent enough to record slight variation of the throughput, and at the same time, it should not put a heavy computational burden on the router. In our experiments, we set the rate of sampling to be 100 samples per second which means we will estimate the throughput every 0.01 second. Note that statistical sampling can be easily achieved using standardized algorithms. Additionally, we use T_s to denote the length of each sampling period, which should be properly chosen. In order to capture the periodicity property of the low-rate TCP attack, the sampling period should be lower bounded by $T_S \geq 2T$ according to the sampling theory. One should also put an upper bound on T_s. Note that a high value of T_s implies a higher storage cost and a higher computational cost for features extraction at the later stage and larger delay in detecting the attack. In our experiments and prototype, we set $T_S = 3$ seconds. Thus we have 300 estimated values of throughput in each sampling period when we set the sampling rate to be 100 samples per second. Another technical issue we have to consider is the *traffic normalization*. Since different link interface may have different line speed, to facilitate feature extraction and comparison at the later stage, the sampled traffic signal of a given link interface will be normalized based on its line speed such that

$$\text{Normalized Throughput} = \frac{\text{Sampled Throughput}}{\text{Maximum Line Capacity}}.$$

C. 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. Note that beside the potential low-rate TCP attack traffic, some other packets may also be included in the sampling process. These packets include:

- Packets that got forwarded to the same interface but they are not designated to the victim site $\mathcal{S}$.
- TCP packets, especially from flows with large RTT, which may be able to survive under the low-rate TCP attack. Please refer to [2].
- UDP packets which will not backoff in the face of low-rate attack or network congestion.

These types of traffic have either a higher frequency or a smaller magnitude, as compared with the burst magnitude of a low-rate attack. To get a clean signal, a low-pass filter can be used to filter the high frequencies and at the same time, clamp all sampled signal to zero if it is less than or equal to a fraction β of the peak value R. In our experiments and prototype, we set it to be less or equal to the maximum value of the normalized background noise N, or $\beta \leq 0.5$.

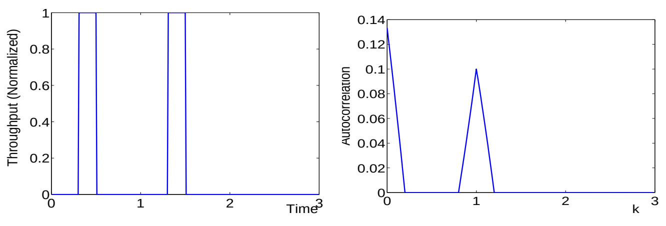
D. Feature Extraction

Auto-correlation is used to extract the periodic signatures of an input signal. Using the auto-correlation measure not only because it is easy to calculate (i.e., 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 the time shifting variable S.

Auto-correlation is calculated with the unbiased internal normalization. The unbiased normalization is necessary if the input signal has a finite sequence. Consider an input signal with n values $(x_0, x_1, \cdots, x_{n-1})$ and all other $x_i = 0$. The unbiased normalized auto-correlation $A(k)$ can be calculated as follows:

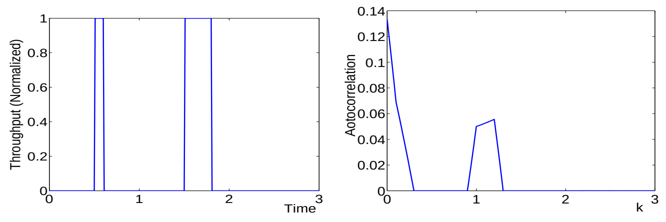
$$A(k) = \frac{1}{n-k} \sum_{i=0}^{n-k+1} x_{i+k} x_i \qquad k = 0, ..., n-1. \tag{1}$$

To illustrate this concept, consider the following auto-correlation plots. Figure 4(a) shows the noise-filtered input signal with time shift $S = 0.3$ sec and periodic property of $T = 1$ and $l = 0.2$ seconds respective. Note that this is the *"classical"* low-rate attack wave. Figure 4(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 on the time shift value S. Consider a more complicated attack wave which is illustrated in Figure 5(a). In this attack, the time shift $S = 0.5$, the first period $T = 1$ second. For the first attack period, the burst length is $l_1 = 0.1$ while the second attack period has the burst length of $l_2 = 0.3$. The auto-correlation plot in Figure 5(b) reveals the existence of a period (e.g., the peak-to-peak distance in the auto-correlation plot) and that bursts may have different durations.



(a) Input sampled traffic (b) Autocorrelation plot

Fig. 4. Auto-correlation of input signal $T = 1, S = 0.2, l = 0.2, R = 1.0$.



(a) Input sampled traffic (b) Autocorrelation plot

Fig. 5. Auto-correlation of input signal $S = 0.5, T = 1, l_1 = 0.1, l_2 = 0.3$.

We extract the feature of auto-correlation plot from an input signal, not only because it captures the periodicity property of the input signal but it also eliminates the problem of time shifting. For the remaining question, we need to address how to compare the auto-correlation plot of an input signal with the auto-correlation plot (or signature) of a low-rate attack.

E. Pattern Matching via the Dynamic Time Warping (DTW) Method

After the first three steps, features are extracted from the sampled input, one has to compare the *similarity* of the extracted features with the signature of the low-rate attack traffic and decide whether there is an on going low-rate attack. Note that an example signature of the low-rate attack is depicted in Figure 4(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 as the signature, for instance, the auto-correlation plot in Figure 5b). Therefore, one has to do proper processing so as to make an accurate decision.

The mechanism we adopted is called the dynamic time warping (DTW) [5], [6]. It is a robust and computational efficient method to compare 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, one can 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

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

A warping path $\mathcal{W}$, is a contiguous set of matrix element $\mathcal{D}$ 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 subjected 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 the path that minimizes the warping cost of $\mathcal{S}$ and $\mathcal{I}$. Formally:

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

In other words, the lower the value of $DTW^*(\mathcal{S},\mathcal{I})$, then the input string $\mathcal{I}$ has higher similarity degree as compare with the signature $\mathcal{S}$. The minimum cost warping path can be found using the *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 distances* 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:

$$\begin{aligned}\gamma(x,y) &= d(x,y) + \\ &\quad \min\{\gamma(x-1,y-1), \gamma(x-1,y), \gamma(x,y-1)\},\end{aligned} \tag{3}$$

where $1 \le x \le n; 1 \le y \le m$. At each step of calculating the value of $\gamma(x,y)$, if the $\min\{\gamma(x-1,y-1), \gamma(x-1,y), \gamma(x,y-1)\} = \gamma(x-1,y)$ 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 $\mathcal{I}$.

From Equation (3), one can see that similar but not identical patterns can match each other with DTW value of 0, i.e, patterns with the same magnitude of burst but different periods like $\{0,0,0,1,1,1,0\}$ and $\{0,0,0,1,1,0,0\}$. Although this scenario is common in other applications like speech recognition and can be viewed as the homology of the input and the template, they should not be treated as identical attack traffic pattern. As a result, we made a modification to the original DTW algorithm that adds some adaptive penalty p for this kind of vertical or horizontal "movement" in the warping path so as to evaluate the similarity while still distinguish the slight difference. Note that the value of the penalty should not be too large since it will increase the DTW value of similar attacks, thus, increase the possibility of false positive or false negative in the detection process. In general, the upper limit of this penalty should not exceed the average value of the template's auto-correlation. As a result, the function of calculating the cumulative distances in our system is:

$$\begin{aligned}\gamma(x,y) &= \parallel s_x - i_y \parallel + \min\{\gamma(x-1,y-1), \\ &\quad \gamma(x-1,y)+p, \gamma(x,y-1)+p\}\end{aligned} \tag{4}$$

After creating this 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 Equation (2).

To illustrate, consider the following example wherein $\mathcal{S} = \{0,0,0,0,1,1,1\}$ and $\mathcal{I} = \{0,0,0,0,1,1,0.8,0.8\}$. The matrix γ and the warping path $\mathcal{W}$ are depicted in Figure 6. In general, a lower value of DTW implies that the input signal $\mathcal{I}$ is very similar to the signature $\mathcal{S}$.

Additionally, from Figure 6, the process of generating the matrix γ by using *dynamic programming* approach to find the minimum DTW value can be seen vividly. The matrix is built

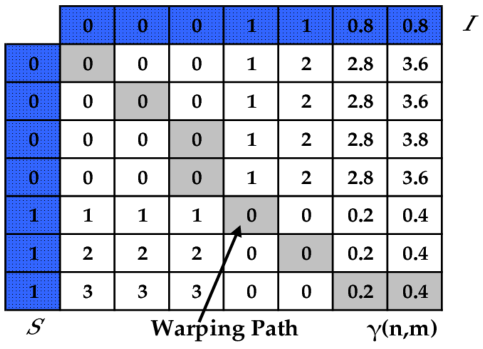


Fig. 6. Distance matrix γ and the warping path $\mathcal{W}$ with $v = h = 0$.

column by column, from left to right and from top down for each column.

The whole procedure of the detection mechanism inside each deployed router can be stated as follows:

Detection Procedure
Assume the capacity of each input port or output port of the router is $C_{\mathcal{P}}$ and the size of sampled input traffic is m.

```
1. Sample the incoming traffic of the current input
     port or output port, call it $I_{real}$;
2. Normalize the throughput: $I_N = \frac{I_{real}}{C_{\mathcal{P}}}$
3. For $i$ = 1 to $m$ { /* remove noise */
     If ($I_N(i) < NoiseThreshold$ )
       $I_F(i) = 0$;
     Else $I_F(i) = I_N(i)$;
   }
4. Calculate the auto-correlation of the filtered input
     $I_A = Auto - correlation(I_F)$
5. Using dynamic programming approach to calculate the
   DTW value of input signal $I_A$ and the template signal $S$
       $D = DTW(S, I_A)$;
6. If ($D <=$ Attack Threshold)
     Low-rate TCP Attack = True;
7. Else Low-rate TCP Attack = False;
```

To implement such detection mechanism, one may choose to put the dynamic detection within a router, or outside a router. To put the detection outside a router, one needs to use a signal splitter so that traffics from a port can be copied to a computing node and the computing node can then perform the dynamic detection on a port by port basis. It is important to point out that the computational complexity of the detection algorithm is very low and it can be carried out in a polynomial time using a dynamic programming approach. In particular, for an input size of m and template size of n, the computational complexity of this DTW is $\Theta(mn)$.

F. Robustness and Accuracy of DTW

Let us consider the robustness and accuracy of using the DTW method to detect a low-rate TCP attack. The experiment setup is as follows. For the template of low-rate attack signature, we consider $T = 1.2$ sec, $l = 0.2$ sec, $R = 1.0$ and $S = N = 0$. Note that although we choose this signature values as the default template in our experiments of the detection, our methodology is general enough for detecting a large family of attack traffic. 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 $\beta_2 = 0.3$, the maximum average throughput of low-rate attack, so that all background traffic that is less than or equal to 30% of the maximum link capacity C will be clamped to zero. Under the DTW, we set the penalty value $p = 0.01$. We consider the following four types of attack traffic:

- ***Strictly Periodic Square Burst (SPSB):*** a strictly periodic signal with a single burst of length l within a period T. The values of l and T are the same for each period.
- ***Random Periodic Square Burst (RPSB):*** a randomly generated periodic signal with a single burst of length l within a period T. The values of l and T between different periods can be different and they are drawn from a uniform distribution (as described below).
- ***Strictly Periodic General burst (SPGB):*** a strictly periodic signal which is generated by a sine wave with period T with an added random noise N. The values of T and N are the same for each period. In reality, the general burst may not be limited to sine wave, and it can be any periodic burst waveform.
- ***Random Periodic General burst (RPGB):*** a randomly generated periodic sine wave with a period of T and with and added random noise N. The values of T and N are drawn from uniform distributions (as described below) and these values may be different from one period to another period.

DTW values for low-rate attack: To generate an input traffic, the period T is uniformly distributed within $[1, 1.5]$. The burst length is uniformly distributed within $(0, 0.5]$, The background noise N 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 around 3000 samples for each of the four types of input traffic discussed above. The results are illustrated in Table I. From the result, one can observe that a large family of low-rate attack has a DTW value which is less than or equal to 35.66.

Values of DTW	**SPSB**	**RPSB**	**SPGB**	**RPGB**
Max DTW	34.88	35.66	34.08	34.69
Min DTW	0	0.80	0.84	1.20
Mean DTW	10.68	9.63	10.89	10.48

TABLE I
DTW VALUES FOR THREE TYPES OF ATTACK TRAFFIC.

DTW values for legitimate traffic (Gaussian): The detection mechanism must distinguish legitimate traffic from the attack stream so as to avoid possible false positive or false negative alert. Therefore, it is desirable to achieve that the *minimum* DTW value of the legitimate traffic be larger than the *maximum* DTW value of any attack traffic so as to reduce the possibility of false positive/negative during the detection.

We carry out the following experiment on legitimate traffic. Based on our assumption before, if there is no low-rate attack, the TCP flows will not back off, all the traffic including TCP and UDP traffic will go through the router properly. We assume that the normal traffic consists of a major constant throughput with some Gaussian noises. In other words: legitimate traffic $= C_1 + \text{random}[0, N]$, where $C_1 \in [0.3, 1]$ and $N \in [0, 0.5]$. We vary the value of C_1 by a step size of 0.01 and for each value of C_1, we generate around 100 different values of N. The results are depicted in Table II. As one can observe, the minimum DTW value for the Gaussian legitimate traffic is above 110 which is much higher than the maximum DTW value of attack traffic reported above. Figure 7 illustrates the *probability density function* of the DTW values for attack and Gaussian flows respectively. From the figure, we observe that there is a clear gap between the Gaussian legitimate traffic and the low-rate attack traffic. Finding a pint to differentiate between legitimate or attack traffic can be easily carried out.

	Gaussian Traffic
Max DTW	286.53
Min DTW	113.50
Mean DTW	236.95

TABLE II
DTW VALUES FOR LEGITIMATE TRAFFIC (GAUSSIAN).

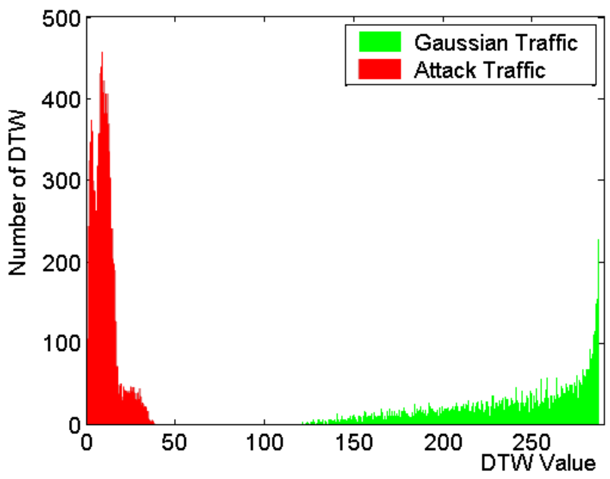


Fig. 7. Probability density functions of DTW values for the attack and the Gaussian legitimate traffic.

DTW values for legitimate traffic (Self-similar): As Gaussian traffic may not perfectly represent all legitimate traffics, we also consider using the self-similar Traffic Model to represent legitimate traffic. It is shown that both the Ethernet local area network [7] and the World Wide Web traffic [8] are statistically self-similar.

Self-similar traffic can be described mathematically. Let $X = (X_t, t = 1, 2, 3, ...)$ be a time series with the mean μ and variance σ^2. The limit of the autocorrelation function $r(k) = E[(X_t - \mu)(X_{t+k} - \mu)]/E[(X_t - \mu)^2]$, $(k = 0, 1, 2, ...)$, when k is approaching infinity, is

$$lim_{k\rightarrow\infty} r(k) = k^{-\beta}, \tag{5}$$

where $0 < \beta < 1$. For each $m = 1, 2, 3, ...$, there is a new time series $X^{(m)} = (X_t^{(m)}, t = 1, 2, 3, ...)$, which is generated by dividing the original series $X = (X_t, t = 1, 2, 3, ...)$ into m non-overlapping segments, where $X_t^{(m)} = 1/m(X_{tm-m+1} + ... + X_{tm})$, $t \geq 1$. If the autocorrelation of $X^{(m)}$ has the same structure as that of X, i.e.,

$$r^{(m)}(k) = r(k),$$

then X is said to be (asymptotically) second order self-similar with degree $H = 1 - \beta/2$, where H is called *Hurst Parameter*. Previous works have shown that the Hurst Parameter H for common Internet traffic is around 0.80.

Based on the definition before, we generate a large number of self-similar traffics using the FARIMA model [9], [10]. We generate the self-similar traffic with *Hurst Parameter* H from 0.75 to 0.85 by the step of 0.01 and 1000 samples for each H with the average rate of throughput ranging from 0.05 to 0.95. The results are depicted in Table III and the *probability density function* of the DTW values for attack and Self-similar flows is illustrated in Figure 8.

	Self-similar Traffic
Max DTW	238.16
Min DTW	28.01
Mean DTW	130.73

TABLE III
DTW VALUES FOR SELF-SIMILAR LEGITIMATE TRAFFIC.

One can observe that the minimum DTW value for self-similar legitimate traffic is less than the maximum DTW value of attack traffic before. Therefore, some false positive and false negative may occur during the detection. However, as shown in Figure 8, the value of self-similar traffic is mainly distributed from 28 to around 238, the intersection area of the attack traffic and the self-similar traffic is rather small compared with both the area of attack traffic and Self-similar traffic separately. Thus the detection mechanism can still be efficient by restricting the false positive and false negative to a small proportion. As depicted in Table IV, among 11000 values of self-similar traffic that we generated only 141 of them are less than the maximum DTW value of the attack traffic. Thus the maximum possible false positive is only a around 1% if

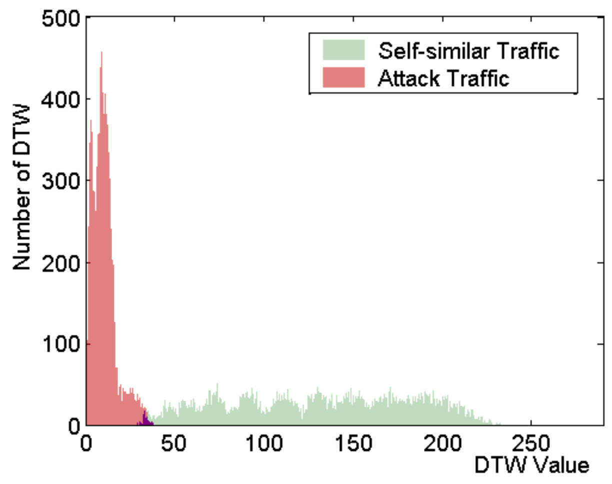


Fig. 8. Probability density functions of DTW values for the attack and the self-similar legitimate traffic.

one sets the attack threshold as the maximum DTW value of attack traffic (i.e., 35.66). Similarly, the maximum possible false negative is around 3.5% if one sets the attack threshold as the minimum DTW value of the self-similar legitimate traffic (i.e., 28.01). In summary, the proposed detection mechanism can successfully distinguish the attack traffic and legitimate traffic with low false positive and/or false negative.

False Self-similar	141	**False Attack**	378
Total Self-similar	11000	**Total Attack**	11492
Max False Positive	1.28%	**Max False Negative**	3.54%

TABLE IV
FALSE DETECTION BETWEEN ATTACK AND SELF-SIMILAR TRAFFIC.

IV. Low-Rate Attack Defense Mechanism

As we discussed in the distributed detection mechanism in Section III, an enabled router $\mathcal{R}$ first determines the existence of low-rate attack on an output port $\mathcal{P}_o$ which it uses to forward packet to a victim site. When a low-rate attack is discovered, $\mathcal{R}$ will then determine the input port that the low-rate attack is coming from. In other word, $\mathcal{R}$ needs to execute the detection algorithm on each of its input port. If the low-rate attack is coming from the input port $\mathcal{P}$, then $\mathcal{R}$ needs to signal all upstream routers which are directly connected to $\mathcal{P}$ to execute the distributed low-rate detection algorithm and execute the defense mechanism. The motivation of this type of push back is to determine the attack as close to the source as possible. This way, we minimize the number of affected TCP flows.

When a router $\mathcal{R}$ discovers the existence of low-rate attack on its output port but *cannot* discover the existence of low rate attack on any of its input ports, this implies that the attack may be using a distributed approach in launching the low-rate attack, for example, sending a short burst at each input port of $\mathcal{R}$ so that these short bursts will converge to a low-rate attack an the output port of $\mathcal{R}$. Under this scenario, the router $\mathcal{R}$ needs to perform the necessary resource management so as to minimize the damaging effect to TCP flows going through the output port $\mathcal{P}_0$.

In our work, we use the deficit round robin (DRR) algorithm to provide the bandwidth allocation and resource protection. The motivation of using DRR is its near perfect isolation of ill-behaved source at a very low implementation cost.

In our case, instead of classifying packet based on its flow, we classify packet according to the input port of $\mathcal{R}$. 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 and packets coming from input port i which are forwarded to output port $\mathcal{P}_0$ will be classified as class i, where $i = 1, \ldots, |\mathcal{P}_i|$. The DRR assigns a Quantum[i] of service to each class i in each round and attempts to serve packets from each class on a per round basis. Each class has a deficit counter, which is deficit_counter[i] and it is initialized to zero, for $i = 1, \ldots, |\mathcal{P}_i|$. At the beginning of a round, deficit counter of each non-empty class i will be increased by the Quantum[i] value (Usually the values of all Quantum[i] are unique and set as Quantum). 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 deficit of the previous rounds gets carried over to the next round and it is only reset to zero whenever there is no packet in that class.

A. *Analysis of Deficit Round Robin Algorithm*

During the traffic scheduling, although packets from different classes (or input ports) can have different sizes, fairness can still be achieved. As shown in [11], [12], the difference in the normalized bytes sent between classes within a certain interval (t_1, t_2) is bounded by a small constant.

We say that class i is *backlogged* during an interval (t_1, t_2) of a DRR execution if the queue for class i is never empty during the interval. We define c_i as the *class share* obtained by the class i that $c_i = \frac{\text{Quantum[i]}}{\text{Quantum}}$ where $\text{Quantum} = Min(\text{Quantum[i]})$. Let $sent_i(t_1, t_2)$ be the total number of bytes sent on the output port by class i in the interval (t_1, t_2). Therefore, the measurement of fairness $FM(t_1, t_2)$ can be expressed as the maximum difference in the normalized bytes sent for class i and j during (t_1, t_2):

$$FM(t_1, t_2) = \max\left(sent_i(t_1, t_2)/c_i - sent_j(t_1, t_2)/c_j\right).$$

Lemma *1: For any class i, during the execution of DRR algorithm, the* deficit_counter[i] *is bounded below by* 0 *and bounded above by Max, where Max is the maximum packet size of all possible packets. Formally, we have*

$$0 \le \text{deficit\_counter[i]} < Max. \tag{6}$$

Proof : Please refer to the appendix. ■

Lemma *2: During any period in which class i is backlogged, the number of bytes sent on the behalf on class i is bounded by*

$$m \cdot \mathsf{Quantum[i]} - Max \leq sent_i(t_1, t_2) \leq m \cdot \mathsf{Quantum[i]} + Max$$

where m is the number of round-robin service opportunities received by class i during this interval.

Proof : Please refer to the appendix. ■

The above two lemmas provide bounds for deficit_counter[i] and $sent_i(t_1, t_2)$. Now we can provide an upper bound on the fairness measure.

Theorem *1: Under the DRR service discipline, for an interval (t_1, t_2), we have the following fairness measure:*

$$FM(t_1, t_2) \leq 2 \cdot Max + \mathsf{Quantum}, \quad where \ \ \mathsf{Quantum} = Min(\mathsf{Quantum[i]}). \tag{7}$$

Proof : Please refer to the appendix. ■

As a result, it is easy to observe that the fairness between classes achieved using DRR algorithm. Additionally, The DRR algorithm is also known to be efficient and can be easily implemented compared with other scheduling algorithm [13]. In general, the processing cost of DRR is $O(1)$ per packet. As a matter of fact, DRR has already been implemented in some of the Cisco's routers [14].

V. Experiments

In this section, we carry out experiments using NS-2 to determine the effectiveness of the proposed defense mechanism.

Experiment 1 (Single TCP flow vs. single source attack):

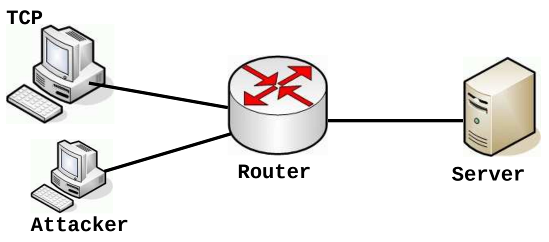


Fig. 9. Single low-rate attack and single TCP flow.

The first experiment is depicted in Figure V. We consider a single low-rate TCP attack and a single TCP flow going through the same router. The latency of each link is 5ms, with the minimum Round Trip Time (RTT) being 20 ms. The capacity of each link is set as 5 Mbps. The low-rate attack is a square burst with $T = 1.0$ sec, burst length $l = 0.2$ sec, burst rate of 5 Mbps or $R = 1$. The low-rate attack uses UDP with packet size of 100 bytes. The packet size of the TCP flow is 500 bytes. Under the DRR, we set the quantum size of each round to be 500 bytes and the buffer size is 5000 bytes. The result is illustrated in Figure V. Note that without the defense mechanism, the router simply uses the conventional scheduling (e.g. drop tail or FCFS) to handle packets. We observe that the TCP flow can only utilize around 4% of the link's bandwidth. On the other hand, when one uses the DRR, we observe an improvement in the TCP's throughput from 224.37 Kbps to 3.402 Mbps, or an improvement from 4.49% to 68.04% of the link capacity.

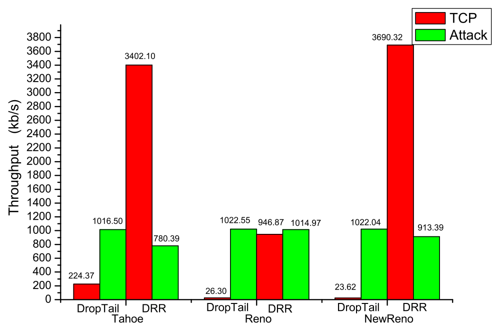


Fig. 10. Result for Low-rate Attack to Single TCP Flow using Tahoe, Reno and New Reno

When we use TCP Reno and new Reno, one may observe that it is not quite effective for TCP Reno, as the throughput can only be increased to less than 20% when DRR is adopted. This is due to the congestion control algorithm of TCP Reno which will have a performance problem when multiple packets are dropped from one transmission window. As mentioned in [15], when TCP Reno incurs multiple packets drop, although it can retransmit the first lost packet after receiving three duplicated ACKs, it is unable to employ Fast Retransmit again and must instead await a retransmission timeout which will then put the sender into the Slow-Start phase. Therefore the DRR will not achieve a good performance for TCP Reno in case there are multiple packets dropped. One possible solution is to increase the DRR buffer. As shown in Table V, we repeat the experiment with different sizes of DRR buffer while all other parameters remain the same. One can observe that the throughput gradually increased to about 85% when the buffer is 30000 bytes.

The result shows the effectiveness of the defense mechanism to protect the TCP flows from the ill-behaved attacking flow.

Experiment 2 (Multiple TCP flows vs. single source attack): The second experiment is depicted in Figure 11. We consider a single low-rate TCP attack and 8 TCP flows going through the same router. Parameters are the same as Experiment 1 except that the buffer size of the DRR-enabled router is 12.5 Kbytes. So the minimum RTT remains the same as 20 ms while the upper bound of RTT is increased to 25 ms. The result is illustrated in Figure 12. Again, using the

Buffer	TCP		Attack flow	
(Bytes)	throughput (Kbps)	% of link capacity	throughput (Kbps)	% of capacity
5000	946.87	18.94%	1014.97	20.30%
15000	1786.92	35.74%	1000.67	20.01%
30000	4286.68	85.73%	656.26	13.13%

TABLE V
RESULT FOR RENO TCP FLOW WITH DIFFERENT DRR BUFFER SIZE.

conventional drop tail scheduling, the total TCP bandwidth is only around 8% of the link's bandwidth. When one uses the DRR, we can improve the throughput of all TCP flows from 423.92 Kbps to 4.390 Mbps, or an improvement from 8.48% to 87.80% of the link capacity. From Figure 12, it is easy to see that flow TCP 4 gains more average throughput than others on the drop tail router. The reason is that TCP 4 has not been completely synchronized by the low-rate attack, and can still transmit several packets during some silent periods between bursts. Figure 13 and 14 depict the same performance gain when we use TCP Reno and new Reno. This shows the effectiveness of the defense mechanism.

Experiment 3 (Multiple TCP flows vs. synchronized distributed low-rate attack): The third experiment is depicted in Figure 15. We consider a distributed low-rate TCP attack and 8 TCP flows going through the same router. Parameters are the same as Experiment 2 except we replace a single attacker by three distributed attackers. Each attacker sends a periodic attack burst every $T = 3.0$ seconds. 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 result is illustrated in Figure 16. One can observe that with DRR, we can improve the throughput of all TCP flows from 469.67 Kbps to 4.296 Mbps, or an improvement from 9.39% to 85.94% of the link capacity. Figure 17 and 18 depict the result when we use TCP Reno and TCP new Reno respectively. Similar observation can be made and this shows the effectiveness of the defense mechanism.

Experiment 4 (Network model of low-rate attack vs. Multiple TCP flows): The fourth experiment is depicted in Figure 19. The transmission bandwidth of all links is 5 Mbps

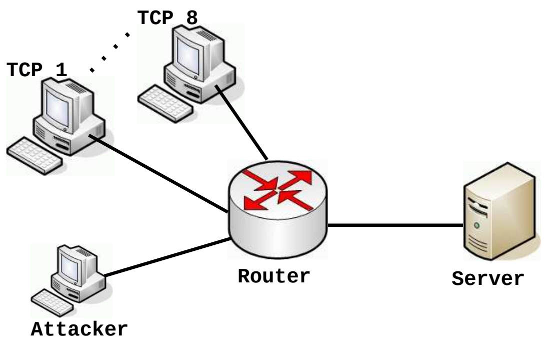


Fig. 11. Single low-rate attack and Multiple TCP flows.

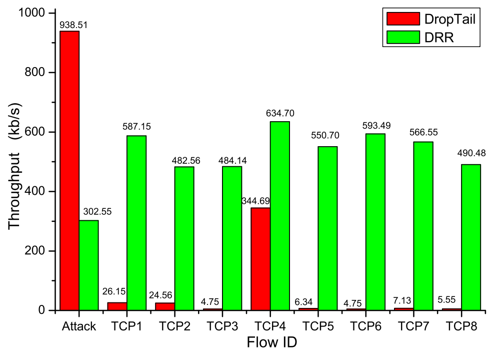


Fig. 12. Result for Single Low-rate Attack to Multiple TCP Flows using Tahoe.

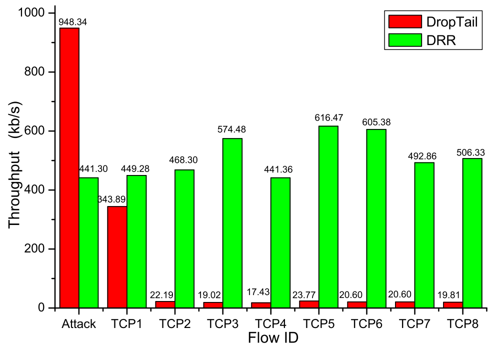


Fig. 13. Result for Single Low-rate Attack to Multiple TCP Flows using Reno

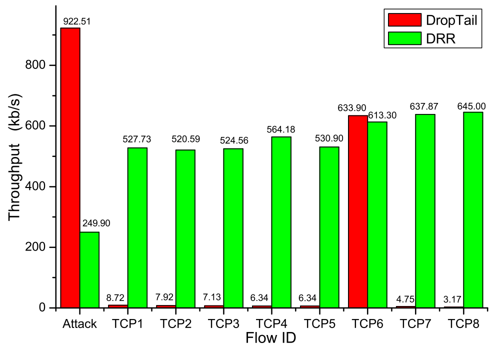


Fig. 14. Result for Single Low-rate Attack to Multiple TCP Flows using New Reno

and the propagation delay is 5 ms. Thus, the minimum RTT for TCP1, TCP2 and the attacker is 50 ms and the RTT for TCP3 and TCP4 are 40 ms and 30 ms respectively. The attacker is located at router R_1 and 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 the TCP 4 is attached to R_7. All of them try to upload files to the server. Table VI shows the throughput of attack and

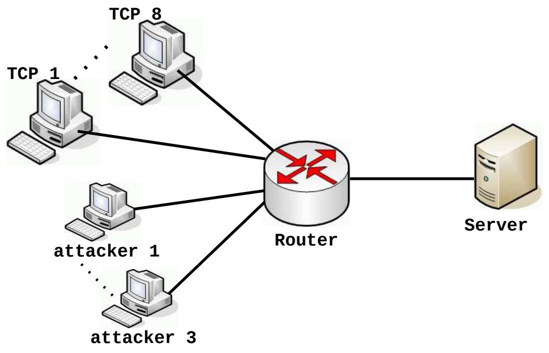


Fig. 15. Distributed low-rate attack and Multiple TCP flows.

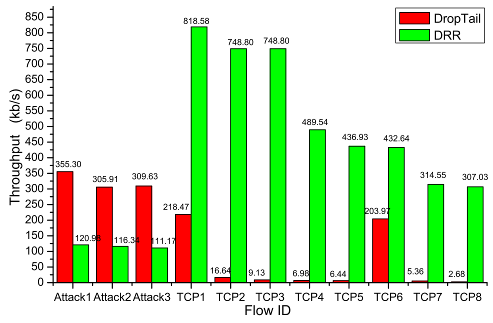


Fig. 16. Result for Synchronized Distribute Low-rate Attack to Multiple TCP Flows using Tahoe

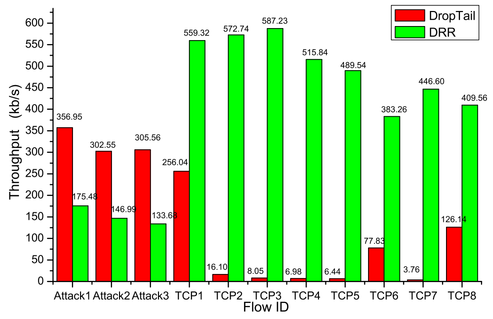


Fig. 17. Result for Synchronized Distribute Low-rate Attack to Multiple TCP Flows using Reno

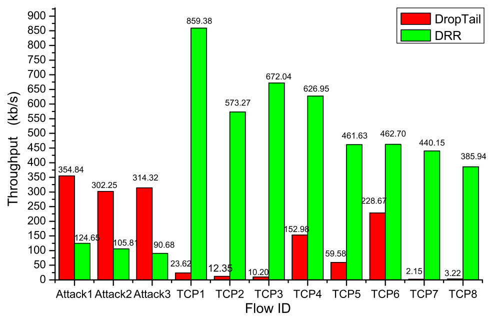


Fig. 18. Result for Synchronized Distribute Low-rate Attack to Multiple TCP Flows using New Reno

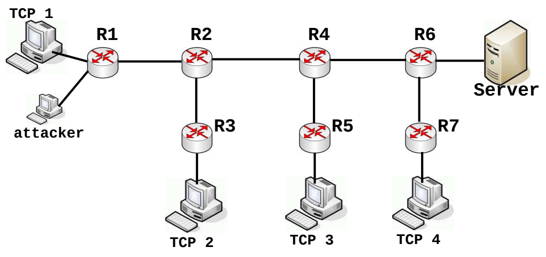


Fig. 19. Network model of Low-rate attack and Multiple TCP flows .

TCP flows when no defense mechanism is deployed (under the drop tail column), as well as the throughput of various flows when DRR is employed at different routing elements. The table shows that enabling the DRR at different routing elements will achieve different TCP throughput. 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 upstream flows (e.g., 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 low-rate attack and they will enable the DRR scheduling. One can observe fairness is achieved wherein all TCP flows will achieve approximately the same amount of bandwidth and they are protected and isolated from the ill-behaved attack flow.

	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
	throughput (kbps)	**throughput** (kbps)	**throughput** (kbps)	**throughput** (kbps)	**throughput** (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 VI
THROUGHPUT OF VARIOUS TCP FLOWS WHEN DIFFERENT ROUTERS ENABLED THE DEFENSE MECHANISM.

Lastly, we like to comment about the practicality of the proposed method. The purpose of our proposed methodology is to provide a practical solution for detecting and defending against the low-rate attack. Consider a victimized core router with 10 interface cards. Although the low-rate attack will converge to one interface card (in which the victim site is attached to that interface card), by performing our defending mechanism, at least 90% TCP flows to the victim site will be isolated from the attack traffic. Additionally, with the cooperation of routers, the pushback mechanism [4] can successfully push the detection and protection as close to the source as possible. Thus more TCP flows will get protected.

VI. Related Work

Network denial of service is a well recognized problem of importance and urgency (e.g., [16], [17]). Various detection and defense approaches have targeted the control of high-rate attacks [3], [18]–[20]. The low-rate TCP attack is first described by Kuzmanovic and Knightly [2], 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 [21]. However, randomizing the RTO requires widespread updates of existing end user software and may reduce the performance of TCP under non-attack conditions [1]. 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 [18], hash-based IP traceback for low volume traffic [22], push-back rate limit [3], [4], and the eXplicit Control Protocol (XCP) [23].

Another work addressing similar problem appeared in [24]. The RoQ attack presents a more general class of adversarial network traffic exploiting the transients of adaptation. A mathematical model was proposed and measurement was carried so as to illustrate the attack potency. Since the attack form presented [24] is similar to this low-rate attack (also periodic burst), we believe, our distributed detection mechanism will shed light towards the detection of such RoQ attack. Note that the general detection of RoQ attack is an ongoing research work.

VII. Conclusion

In this work, we present a distributed and efficient approach 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, and then we 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 attack. In particular, one can achieve very low false positive/negative when compare to legitimate Internet traffics. When the low-rate attack is present, we use a push back mechanism so as to identify the attack as close to the attack source as possible. The rationale of this push back is to minimize the number of affected TCP flows. We show that one can use the deficit round-robin approach to protect the TCP flows and isolate them from the attack traffic. Experiments are carried out to quantify the robustness and accuracy of the proposed detection. Extensive simulations are carried to quantify the merits and effectiveness of the proposed defense mechanism.

Acknowledgment: We wish to thank the anonymous referees for their helpful and insightful comments.

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

Appendix

Note: We follow similar methodology in [11] for the proofs.

Lemma 1: *For any class i, during the execution of DRR algorithm, the* deficit_counter[i] *is bounded below by 0 and bounded above by Max, where Max is the maximum packet size of all possible packets. Formally, we have*

$$0 \leq \mathsf{deficit\_counter[i]} < Max.$$

Proof : At the beginning of the algorithm, we set $\mathsf{deficit\_counter[i]} = 0$, which is obviously less than Max. At the end of the service round of class i, we need to consider two cases:

1) If there is a packet left in the queue of class i, then its size is greater than deficit_counter[i]. Since the size of any packet is no more than Max, we have $\mathsf{deficit\_counter[i]} < Max$ and $\mathsf{deficit\_counter[i]} > 0$.
2) If the queue of class i is empty, then deficit_counter[i] is reset to zero. ∎

Lemma 2: *During any period in which class i is backlogged, the number of bytes sent on the behalf on class i is bounded by*

$$m\cdot\mathsf{Quantum[i]} - Max \leq sent_i(t_1, t_2) \leq m\cdot\mathsf{Quantum[i]} + Max$$

where m is the number of round-robin service opportunities received by class i during this interval.

Proof : Let use use the term "*round*" to denote service opportunities received by class i within an interval (t_1, t_2). We number these rounds from 1 to round m. With loss of generality, we treat t_1, the start of an interval, as the end of round 0. Define $\mathsf{deficit\_counter[i][k]}$ as the value of $\mathsf{deficit\_counter[i]}$ at the end of round k. We also define $bytes_i(k)$ as the bytes sent by class i in round k and $sent_i(k)$ be the bytes sent by class i from round 1 through k. We have $sent_i(k) = \sum_{k=1}^{m} bytes_i(k)$.

It is easily observed that $bytes_i(k) + \mathsf{deficit\_counter[i][k]} = \mathsf{Quantum[i]} + \mathsf{deficit\_counter[i][k-1]}$. As in round k, the accumulated allocation to class i is $\mathsf{Quantum[i]} + \mathsf{deficit\_counter[i][k-1]}$. Therefore, if class i sends $bytes_i(k)$, then the reminder will be stored in deficit_counter[i][k]. Since the queue for class i never empties during the interval (t_1, t_2), we will have:

$$\begin{aligned} bytes_i(k) =& \mathsf{Quantum[i]}+ \\ & \mathsf{deficit\_counter[i][k-1]} - \mathsf{deficit\_counter[i][k]}. \end{aligned}$$

Summing this over all m rounds of servicing of class i, and because $sent_i(k) = \sum_{k=1}^{m} bytes_i(k)$, we have

$$\begin{aligned} sent_i(m) =& m \cdot \mathsf{Quantum[i]}+ \\ & \mathsf{deficit\_counter[i][0]} - \mathsf{deficit\_counter[i][m]}. \end{aligned}$$

Then the result follows because deficit_counter[i] is always non_negative and upper bounded by Max (by Lemma 1). ■

Theorem 1: *Under the DRR service discipline, for an interval* (t_1, t_2)*, we have the following fairness measure:*

$$\begin{aligned} FM(t_1, t_2) \leq& 2 \cdot Max + \mathsf{Quantum}, \\ & where \;\; \mathsf{Quantum} = Min(\mathsf{Quantum[i]}). \end{aligned}$$

Proof : Consider an interval (t_1, t_2) under the DRR algorithm and any two class i and j that are backlogged in this interval. As each class is serviced in a strict round-robin mode, therefore, if we let m be the number of the round-robin opportunities given to class i in the interval (t_1, t_2), and we let m' be the number of round-robin opportunities given to class j in the same interval, then we have $\mid m - m' \mid \leq 1$. From Lemma 2 we get:

$$sent_i(t_1, t_2) \leq m \cdot \mathsf{Quantum[i]} + Max.$$

Based on the definition, c_i, the share given to any class i, is equal to Quantum[i]/Quantum. Therefore the normalized service received by class i is

$$sent_i(t_1, t_2)/c_i \leq m \cdot \mathsf{Quantum} + Max/c_i.$$

Similarly, for class j, we can obtain:

$$\begin{aligned} sent_j(t_1, t_2) &\geq m' \cdot \mathsf{Quantum[j]} - Max \\ &= m' \cdot \mathsf{Quantum[j]} - Max \end{aligned}$$

and

$$sent_j(t_1, t_2)/c_j \geq m' \cdot \mathsf{Quantum[j]} - Max/c_j.$$

Subtracting the equations for the normalized service for class i and j, and using the fact that $m - m' \leq 1$, we get

$$\frac{sent_i(t_1, t_2)}{c_i} - \frac{sent_j(t_1, t_2)}{c_j} \leq \mathsf{Quantum} + \frac{Max}{c_i} + \frac{Max}{c_j}$$

As both c_i and c_j are greater than 1, the theorem follows. ■

References