2011

A Survey of Deployment Information of Delay-based TCP Congestion Avoidance Algorithm for Transmitting Multimedia Data

Peng Yang
University of Nebraska - Lincoln, pyang@cse.unl.edu

Lisong Xu
University of Nebraska - Lincoln, xu@cse.unl.edu

45th Annual Simulation Symposium

2011 IEEE GLOBECOM Workshops (GC Wkshps)
doi: 10.1109/GLOCOMW.2011.6162434

A Survey of Deployment Information of Delay-based TCP Congestion Avoidance Algorithm for Transmitting Multimedia Data

Peng Yang, Lisong Xu
Department of Computer Science and Engineering
University of Nebraska-Lincoln
Lincoln, NE 68588-0115
Email: {pyang, xu}@cse.unl.edu

***Abstract*—Multimedia traffic comprises a significant part of the total Internet traffic. Due to the real-time nature of the multimedia traffic, low queuing delay is critical to many multimedia applications. This requirement makes delay-based TCP congestion avoidance algorithms (or delay-based TCP algorithms for short) a good choice to transmit multimedia data, since they can help keep a low queuing delay in the Internet. However, the Internet traffic is controlled by heterogeneous TCP algorithms and many of them are non-delay-based. Thus, the effort made by the delay-based TCP algorithms to reduce the queuing delay is often offset by the non-delay-based TCP algorithms. Indeed, unless a significant part of the total Internet traffic is controlled by the delay-based TCP algorithms, we probably won't see a big improvement in terms of the queuing delay. This observation motivates us to develop a method to differentiate between the delay-based and the non-delay-based TCP algorithms and use it to investigate the deployment information of the delay-based TCP algorithms on the web servers in the Internet. Our purpose is to gain a preliminary understanding about the amount of the Internet traffic controlled by the delay-based TCP algorithms, and hence the impact of the delay-based TCP algorithms on the Internet queuing delay. Such information is valuable for those people who plan to use the delay-based TCP algorithms to transmit their multimedia data.**

I. Introduction

Multimedia traffic comprises a significant part of the total Internet traffic. Due to its importance, many methods have been proposed during the past few years to transmit multimedia data efficiently. Particularly, delay-based TCP congestion avoidance algorithms (or delay-baed TCP algorithms for short), e.g., TCP-Vegas [1], appear to be a good solution. A TCP algorithm is delay-based if it adjusts the congestion window size according to the queuing delay measured. Specifically, it increases the congestion window size fast when the queuing delay is low, and increases the congestion window size slowly or even decreases it when the queuing delay is high in the congestion avoidance state. According to this definition, TCP-Compound [2], TCP-Illinois [3], TCP-Vegas [1], TCP-Veno [4] and TCP-Yeah [5] can all be considered as delay-based. TCP-AIMD [6], TCP-CUBIC [7], TCP-BIC [8], TCP-HighSpeed [9], HTCP [10], TCP-Scalable [11] and TCP-Westwood+ [12] can be considered as non-delay-based.

Due to the real-time nature of the multimedia traffic, low queuing delay is critical to the performance of many multimedia applications. This requirement makes the delay-based TCP algorithms a good choice to transmit multimedia data, since they can help keep a low queuing delay in the Internet. However, the effectiveness of the delay-based TCP algorithms also depends on other competing TCP algorithms. The Internet traffic is controlled by heterogenous TCP algorithms, among which some TCP algorithms are non-delay-based. Thus, the effort made by the delay-based TCP algorithms to lower the queuing delay is often offset by those non-delay-based TCP algorithms. Indeed, unless a significant part of the total Internet traffic is controlled by the delay-based TCP algorithms, we probably won't see a big improvement in terms of the queuing delay. This observation leads us to investigate the deployment information of the delay-based TCP algorithms in the Internet. We would like to gain a preliminary understanding about the amount of the Internet traffic controlled by the delay-based TCP algorithms. This information not only lets us understand the impact of the delay-based TCP algorithms on the queuing delay in the Internet but also is valuable for those people who plan to use the delay-based TCP algorithms to transmit their multimedia data.

The contribution of the paper is as follows.

- We propose a method, called Delay-based TCP Congestion Avoidance Algorithm Differentiation (DCAAD), to differentiate between the delay-based and the non-delay-based TCP algorithms.
- We investigate the deployment information of the delay-based TCP algorithms on more than 1800 web servers in the Internet. We find that the number of the web servers using the delay-based TCP algorithms is very small. Thus, we believe that, at the present time, the impact of the delay-based TCP algorithms on the queuing delay in the Internet is small, and hence the multimedia traffic controlled by the delay-based TCP algorithms probably won't experience a big improvement in terms of the queuing delay.

Note that, the conclusion does not object to the use of the

delay-based TCP algorithms for transmitting the multimedia data. It merely means the advantage of using the delay-based TCP algorithms is small at the present time. As more and more Internet nodes use the delay-based TCP algorithms, we will see bigger impact of the delay-based TCP algorithms on the queuing delay in the Internet.

As far as we know, this is the first paper on distinguishing the delay-based and the non-delay-based TCP algorithms. Most papers on measuring TCP behaviors, such as [13] and [14], focus on the TCP components other than the TCP algorithm. The most relevant paper is [15], where we try to identify which one of the 15 TCP algorithms of Windows and Linux is used by a web server. However, in this paper, we determine whether the TCP algorithm of a web server is delay-based or not, and the TCP algorithm could be any TCP algorithm (i.e., not limited to those 15 TCP algorithms) and could even be an unknown TCP algorithm (i.e., not publicly announced).

The rest of the paper is organized as follows: in Sections II, III, IV and V, we present our method for differentiating between the delay-based and the non-delay-based TCP algorithms. In Section VI, we evaluate the differentiation method using cross validation and the Internet experiments. Finally, in Section VII, we conclude the paper.

II. Differentiation Between the Delay-based and the Non-delay-based TCP Algorithms

A. Design Goals

In this section, we present an overview of DCAAD. DCAAD collects traces of the congestion window sizes (called congestion window traces) from a web server and infers a delay feature vector for differentiating between the delay-based and the non-delay-based TCP algorithms. The goal of DCAAD is to differentiate all delay-based TCP algorithms officially implemented in Windows and Linux operating systems and some unknown and proprietary delay-based TCP algorithms from the non-delay-based TCP algorithms.

We consider TCP-Compound, TCP-Illinois, TCP-Vegas, TCP-Veno and TCP-Yeah, a total of 5 well-known delay-based TCP algorithms. We also employ a machine learning algorithm, i.e., Support Vector Machine (SVM), so that we can differentiate some unknown and proprietary delay-based TCP algorithms from the non-delay-based TCP algorithms.

B. Delay Feature Vectors for Differentiating Between the Delay-based and the Non-delay-based TCP Algorithms

There are two features that can be used to differentiate between the delay-based and the non-delay-based TCP algorithms. The first feature is multiplicative decrease parameter (denoted by β). Let $loss\_cwnd$ denote the congestion window size just before a congestion event. In case of 3 duplicated ACK packets, TCP sets both its slow start threshold (denoted by *ssthresh*) and its congestion window size to $\beta \times loss\_cwnd$; in case of a timeout, TCP sets *ssthresh* to $\beta \times loss\_cwnd$ and sets its cognestion window size to usually 1 packet. For some delay-based TCP algorithms, parameter β is affected by the queuing delay. For example, TCP-Veno sets parameter β to 0.8 when the queuing delay is small and set it to 0.5 when the queuing delay is large.

TABLE I
VECTOR ELEMENTS FOR THE DELAY-BASED AND THE NON-DELAY-BASED TCP ALGORITHMS

	$\frac{\beta_h}{\beta_l}$	$\frac{\alpha_h}{\alpha_l}$	$\frac{\alpha_h}{\hat{\alpha}_l}$
Delay-based TCP algorithms	≤ 1.0	< 1.0	< 1.0
Non-delay-based TCP algorithms	$= 1.0$	≥ 1.0	≥ 1.0

The second feature is the congestion window growth speed (denoted by α) in the congestion avoidance state. For the delay-based TCP algorithms, the congestion window increases fast when the queuing delay is small and increases slowly when the queuing delay is large. For example, TCP-Illinois increases the congestion window size by 10 packets every round trip time (RTT) when the queuing delay is small and increases the congestion window size by 0.3 packets every RTT when the queuing delay is large.

Thus, we can use the change of parameters β and α under different queuing delays to differentiate between the delay-based and the non-delay-based TCP algorithms. In DCAAD, this change is represented by a delay feature vector. DCAAD measures parameters β and α in two different network environments. In network environment A, DCAAD maintains a fixed RTT between itself and a web server (thus, approximately 0 queuing delay). Let's refer to parameter β as β_l and refer to parameter α as α_l in network environment A, respectively. In network environment B, DCAAD increases the queuing delay from 0 second to 0.2 seconds twice during its communication with a web server. The first queuing delay increase is to measure parameter β, referred to as β_h, under a large queuing delay; the second queuing delay increase is to measure the change of the parameter α when the queuing delay becomes larger. Let's refer to parameter α before the second queuing delay increase as $\hat{\alpha}_l$ and refer to parameter α after the second queuing delay increase as α_h, respectively in network environment B. A delay feature vector can be defined as $< \frac{\beta_h}{\beta_l}, \frac{\alpha_h}{\alpha_l}, \frac{\alpha_h}{\hat{\alpha}_l} >$.

The delay feature vector can be used to distinguish between the delay-based and the non-delay-based TCP algorithms in that the changes of parameters β_h, β_l, α_h, α_l and $\hat{\alpha}_l$ in the two network environments are different for these two types of TCP algorithms. This leads to the different values of the vector elements for the delay-based and the non-delay-based TCP algorithms, which is show in Table I.

Note that the relationships shown in Table I do not holds for some non-delay-based TCP algorithms, where parameters β and α are affected by the factors related or unrelated to the queuing delay in some situations. The details for handling those exceptions are discussed in Section IV.

C. Differentiation Steps

The differentiation includes three steps.

- Step 1: Trace Gathering. In this step, we collects the congestion window traces from a web server in network

environments A and B.

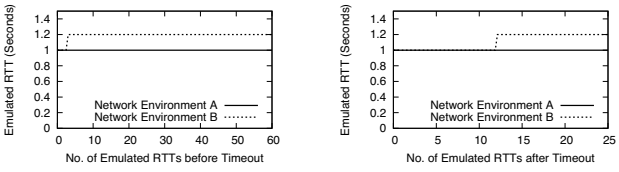
- Step 2: Feature Extraction. We extract parameters β_l, β_h, α_l, α_h and $\hat{\alpha}_l$ from the congestion window traces and calculate the delay feature vector.
- Step 3: Differentiation. We use an SVM to analyze a delay feature vector and determine whether a web server uses a delay-based TCP algorithm or not.

III. Step 1: Trace Gathering

In this section we discuss DCAAD trace gathering that collects the congestion window traces from a web server in network environments A and B. A network environment specifies how DCAAD communicates with a web server. Basically, it is a standard HTTP session, where DCAAD downloads a file from a web server. However, it differs from a standard HTTP session in serveral places to faciliate the extraction of parameters β_l, β_h, α_l, α_h and $\hat{\alpha}_l$ from the congestion window traces collected.

Firstly, in both network environments, DCAAD creates a timeout event so that the TCP of a web server can change its state from the inital slow start state (before timeout) to the second slow start state (after the timeout), and to the congestion avoidance state. The reason for this state transition is that we have to collect the congestion window traces in the congestion avoidance state to extract parameters α_l, α_h and $\hat{\alpha}_l$. Also, we need the congestion window traces during the second slow start state and the congestion avoidance state to extract paramters β_l, β_h.

Secondly, a network environment should maintain specific RTTs and queuing delays between DCAAD and a web server. As dicussed in Section II-B, the reason for specific RTTs and queuing delays is to observe the change of parameters β_l, β_h, α_l, α_h and $\hat{\alpha}_l$ under different queuing delays. In addtion, in both network environments, we emulate very long RTTs between DCAAD and a web server. Long RTTs means large bandwidth-delay product, which can help us figure out how many data packets a web server sends in one RTT, i.e. the congestion window size. In network environment A, DCAAD maintains the RTT of 1.0 second between itself and a web server; in network environment B, DCAAD varies the RTT to either 1.2 or 1.0 seconds as specified in Figure 1. As there



(a) Before Timeout (b) After Timeout

Fig. 1. The RTTs of the two network environments A and B.

is no variation in the RTT perceived by a web server, the queuing delay measured by the web server is approximately 0 in network environment A. However, in network environment B, a web server observes a queuing delay of 0.2 seconds both before the timeout and after the timeout.

There are two important parameters for both network environments: the size of the data packets sent by a web server, which is denoted by mss_{em}; and the congestion window size of a web server when DCAAD creates a timeout event. We require that this particular congestion window size is greater than or equal to a predefined parameter denoted by $timeout_{em}$. The guidline for setting these two parameters is to grow the congestion window of a web server to a large value, which is helpful for us to distinguish between the delay-based and the non-delay-based TCP algorithms. For parameter mss_{em}, we want to set it to a small value, since given a fixed amount of data to download, a small parameter mss_{em} can help grow the congestion window size of a web server to a large value. We first set parameter mss_{em} to 100 bytes. If a web server cannot send the data packets with this size, we gradually grow parameter mss_{em} to 300, 536 and finally 1460 bytes. For parameter $timeout_{em}$, we want to set it to a large value. However, large parameter $timeout_{em}$ requires us to find a large file on a web server to download. Thus, we first set parameter $timeout_{em}$ to 512 packets. If we cannot find a large enough file, we reduce parameter $timeout_{em}$ to 256 packets. It is hard to differentiate some delay-based TCP algorithms from the non-delay-based TCP algorithms if parameter $timeout_{em}$ is smaller than 256 packets.

IV. Step 2: Feature Extraction

A. Delay Feature Vector Construction

Parameters β_l and β_h are extracted by figuring out the slow start threshold (i.e., *ssthresh*) in the two congestion window traces, respectively. Once we obtain the *ssthresh*, these two parameters can be easily computed as the ratio between the *ssthresh* and the congestion window size immediately before the timeout event. Interested readers may refer to [15] for the details. Then, we can calculate the first element of a delay feature vector, i.e. the ratio $\frac{\beta_h}{\beta_l}$.

Parameters α_l, α_h and $\hat{\alpha}_l$ in network environments A and B can be calculated as the coefficients in the linear functions fitting the sub-traces in the congestion window traces. Let's refer to the congestion window trace in network environment A as trace A and refer to the congestion window trace in network environment B as trace B. We fit a sub-trace in the trace A using a linear function $g_l(x) = m_l x + n_l$. The sub-trace should be taken in the congestion avoidance state in order to measure congestion window growth speed α_l. Furthermore, since we compare parameter α_l with parameter α_h, the sub-trace should start from the point after the second queuing delay increase in network environment B and end with the last congestion window size in the trace A. We set parameter α_l to m_l. Parameters α_h and $\hat{\alpha}_l$ are obtained by fitting two sub-traces in the trace B. The sub-trace for parameter $\hat{\alpha}_l$ begins with the first congestion window size in the congestion avoidance state and ends with the last congestion window size before the second queuing delay increase. The sub-trace for parameter α_h starts with the last congestion window size in the sub-trace for parameter $\hat{\alpha}_l$ and ends with the last congestion window size in the trace B. Let's denote the linear function

fitting the sub-trace for parameter $\hat{\alpha}_l$ by $\hat{g}_l(x) = \hat{m}_l x + \hat{n}_l$ and denote the linear function fitting the sub-trace for parameter α_h by $g_h(x) = m_h x + n_h$. We set parameter $\hat{\alpha}_l$ to $\hat{m}_l$ and set parameter α_h to m_h, respectively. Then, we can calculate the second element and the third element of a delay feature vector, i.e. the ratio $\frac{\alpha_h}{\alpha_l}$ and the ratio $\frac{\alpha_h}{\hat{\alpha}_l}$.

B. Handling Exceptions

As said previously, $\frac{\beta_h}{\beta_l}$ is less than or equal to 1.0 and $\frac{\alpha_h}{\alpha_l}$ is smaller than 1.0 for the delay-based TCP algorithms. For most non-delay-based TCP algorithms, the first ratio is 1.0 and the second ratio is greater than or equal to 1.0. However, there are exceptions where parameters β_l, β_h, α_l and α_h are affected by some other factors related or unrelated to the queuing delay.

1) The Exceptions for Parameters β_l and β_h: (1) *TCP-HighSpeed*. In TCP-HighSpeed [9] parameter β is affected by parameter $loss\_cwnd$, if the timeout happens in the congestion avoidance state. If the network environments are not carefully designed, we may end up with the ratio $\frac{\beta_h}{\beta_l} \neq 1.0$ if parameter $loss\_cwnd$ is different in network environments A and B. We handle this problem by creating a timeout in the slow start state in the network environments. In this way, parameter β always equals 0.5. Thus, the ratio $\frac{\beta_h}{\beta_l}$ is 1.0, which is consistent with most non-delay-based TCP algorithms.

(2) *HTCP*. For HTCP [10], parameter β is affected by the achieved throughput after the first congestion event. Thus, it is possible that the ratio $\frac{\beta_h}{\beta_l} \neq 1.0$ if we have different throughput in carelessly designed network environments. However, in our network environments, there is only one timeout. At this point, parameter β always equals 0.5. Thus, the ratio $\frac{\beta_h}{\beta_l}$ is 1.0, which is consistent with most non-delay-based TCP algorithms.

(3) *TCP-Westwood+*. For TCP-Westwood+ [12], parameter β is affected by the estimated bandwidth, which means the ratio $\frac{\beta_h}{\beta_l}$ may not be equal to 1.0 in carelessly designed network environments. However, in our network environments, the ratio $\frac{\beta_h}{\beta_l}$ is close to 1.0.

2) The Exceptions for Parameters α_l and α_h: (1) *TCP-CUBIC and HTCP.* For TCP-CUBIC [7] and HTCP, the congestion window growth speed depends on the elapsed time from the last congestion event. The speed becomes faster when the elapsed time is longer. If the elapsed time is longer in network environment A than in network environment B, we may end up with the ratio $\frac{\alpha_h}{\alpha_l} < 1.0$, which is not consistent with most non-delay-based TCP algorithms. We handle this problem by using longer RTTs in network environment B than in network environment A. In this way, the elapsed time is longer in network environment B than in network environment A and the ratio $\frac{\alpha_h}{\alpha_l}$ for TCP-CUBIC and HTCP is bigger than 1.0, which is consistent with most non-delay-based TCP algorithms.

(2) *TCP-HighSpeed and TCP-Scalable* In TCP-HighSpeed and TCP-Scalable [11], the congestion window growth speed depends on the congestion window size. As the congestion window size becomes larger, the speed becomes faster. In our network environments, the congestion window size in the congestion avoidance state depends on parameter $loss\_cwnd$, i.e. the congestion window size right before the timeout. For network environments A and B, parameter $loss\_cwnd$ may not the same. Thus, it is possible that the ratio $\frac{\alpha_h}{\alpha_l} < 1.0$, which is not consistent with most non-delay-based TCP algorithms. We handle this problem by using the third element in a delay feature vector, i.e. the ratio $\frac{\alpha_h}{\hat{\alpha}_l}$. Parameters α_h and $\hat{\alpha}_l$ are obtained in the same congestion window trace and have the same parameter $loss\_cwnd$. If the ratio $\frac{\alpha_h}{\hat{\alpha}_l} >= 1.0$, the TCP algorithm can not be delay-based, because this means the congestion window grows faster or at least at the same speed after the second queuing delay increase than before the increase in network environment B. Fortunately, for both TCP-HighSpeed and TCP-Scalable, the ratio $\frac{\alpha_h}{\hat{\alpha}_l} > 1.0$. Thus, we can easily distinguish them from the delay-based TCP algorithms. Actually, in the differentiation of the delay-based TCP algorithms, the first step we conduct is to check the ratio $\frac{\alpha_h}{\hat{\alpha}_l}$. If it is greater than or equal to 1.0, we classify the TCP algorithm as non-delay-based.

V. Step 3: Differentiation

The differentiation of the delay-based TCP algorithms involves two sub-steps.

- Sub-step 1: Check the third element of a delay feature vector, i.e. the ratio $\frac{\alpha_h}{\hat{\alpha}_l}$. If it is greater than or equal to 1.0, then classify the TCP algorithm as non-delay-based, because this means the congestion window grows faster or at the same speed after the second queuing delay increase than before the increase in network environment B.
- Sub-step 2: If $\frac{\alpha_h}{\hat{\alpha}_l} < 1.0$, then analyze the first two elements of a delay feature vector using an SVM and decide whether a TCP algorithm is delay-based or not.

Why not just use the ratio $\frac{\alpha_h}{\hat{\alpha}_l}$ to distinguish the delay-based and the non-delay-based TCP algorithms? The ratio $\frac{\alpha_h}{\hat{\alpha}_l}$ alone is not sufficient. For example, even if the ratio $\frac{\alpha_h}{\hat{\alpha}_l} < 1.0$, we still do not know whether this change of the congestion window growth speed is due to the queuing delay increase or not. Actually, there are some non-delay-based TCP algorithms we discovered with the ratio $\frac{\alpha_h}{\hat{\alpha}_l} < 1.0$. We plot one such non-delay-based TCP algorithm in Figure 2.

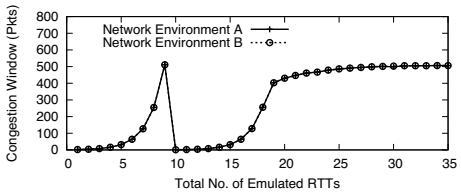


Fig. 2. A non-delay-based TCP algorithm with the ratio $\frac{\alpha_h}{\hat{\alpha}_l} < 1.0$. Its congestion window size grows faster when it just enters the congestion avoidance state than thereafter.

The first two elements of a delay feature vector is analyzed using an SVM, which can be formally expressed by the following expression.

$$y(x) = w^T \phi(x) + b \tag{1}$$

where x is the input to the SVM; ϕ is a function that maps input x to another space suitable for classification; w is the weight associated with the input, the subscript T means transpose; and b is a constant. In our case, input x is the first two elements of a delay feature vector. If $y(x) \geq 1$, then we classify the corresponding TCP algorithm as delay-based; if $y(x) \leq -1$, then we classify the TCP algorithm as non-delay-based.

Parameters w and b are determined by minimizing the classification error on some training data. The training data contains the delay feature vectors for various delay-based and non-delay-based TCP algorithms obtained on our local test-bed. We label each piece of training data (i.e. a delay feature vector). If a piece of training data corresponds to a delay-based TCP algorithm, the label is 1; if it corresponds to a non-delay-based TCP algorithm, the label is -1. Let's denote the entire training data by $x_i, i = 1, ..., l$ and denote the corresponding labels by $y_i, i = 1, ..., l$. Parameters w and b can be obtained by solving the following optimization problems.

$$\min 1/2 w^T w + c \sum_{i=1}^{l} s_i \tag{2}$$

subject to

$$y_i(w^T \phi(x_i) + b) \geq 1 - s_i, i = 1, ..., l \tag{3}$$

$$s_i \geq 0, i = 1, ..., l \tag{4}$$

where parameter $c > 0$, and $s_i, i = 1, ..., l$ are slack variables represening the classification error.

We have several choices of mapping function ϕ and parameter c. Mapping function ϕ is usually specified by a corresponding kernel function Ke, which is defined as $Ke(x_i, x_j) = \phi(x_i)\phi(x_j)$. We choose the best kernel function and parameter c according to the distribution of delay feature vectors and the result of cross validation (see Section VI-B).

VI. Evaluation

In this section, we evaluate our differentiation method, including training data collection, cross validation and the Internet experiments targeting real-world web servers.

A. Collecting Training Data

The training data used to determine the parameters in the SVM is collected on our local test-bed. The test-bed consists of four computers: one DCAAD computer, one Linux web server, and one Windows web server, all connected to a Linux router. The DCAAD computer is running OpenSUSE 11.1 operating system (kernel version 2.6.27). The Linux web server is also running OpenSUSE 11.1 with an Apache web server installed; and the Windows web server is running Windows Server 2008 with an IIS web server installed.

We run Netem [16] on the Linux router to emulate the two network environments discussed above with different packet loss rates and different actual RTTs between the DCAAD computer and the web servers. We emulate 7 packet loss rates for both data packets and ACK packets: 0%, 0.01%, 0.05%, 0.1%, 0.5%, 1%, and 5%, and 2 actual RTTs: 50ms and 250ms. Combining the network environments, different packet loss rates and different actual RTTs there are $2 \times 7 \times 2 = 28$ unique network conditions between the DCAAD computer and the web servers. For each network condition, we run DCAAD trace gathering with different parameter $timeout_{em}$: 512 and 256. In total, we run DCAAD trace gathering $28 \times 2 = 56$ times for every TCP algorithms officially supported by the Windows and Linux operating systems [1]. We plot the first two elements of the delay feature vectors generated in Figure 3.

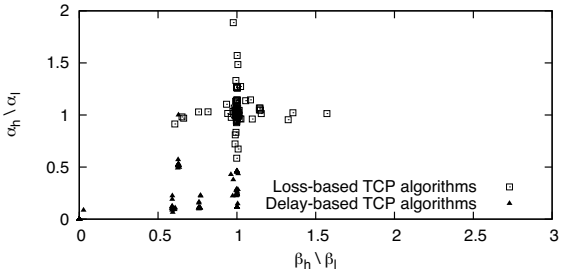


Fig. 3. The first two elements of the delay feature vectors obtained on the local test-bed. The distribution shows that the first two elements of the delay feature vectors for the delay-based and the non-delay-based TCP algorithms are almost separated. Note that there are ratios $\frac{\alpha_h}{\alpha_l} < 1.0$ and ratios $\frac{\beta_h}{\beta_l} \neq 1.0$ for some non-delay-based TCP algorithms due to packet losses and measurement errors.

B. Cross Validation

There are two parameters in the SVM should be determined by conducting cross validation. One is the kernel function and the other one is parameter c. According to Figure 3, the first two elements of the delay feature vectors for the delay-based and the non-delay-based TCP algorithms are almost separated. This means simple kernel functions, such as linear kernel function, should produce fairly good result. Thus, we choose the linear kernel function. We also consider the following values of parameter c: 2^{-5}, 2^{-3}, ..., 2^{13}, 2^{15}. We determine the best parameter c using K-fold cross-validation (K = 10) implemented by C.-C. Chang et al. [19] on the training data.

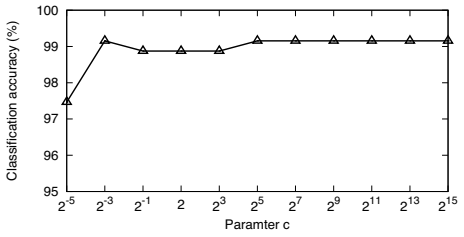


Fig. 4. SVM cross-validation accuracy.

[1]Among all TCP algorithms officially supported by the Linux operating systems, we do not consider TCP-Hybla [17] and TCP-LP [18], which is designed for satellite communication and background file transfer, respectively. Also, we consider two versions of TCP-CUBIC: Linux kernel 2.6.25 and before, and Linux kernel 2.6.26 and after

Figure 4 shows that the cross-validation accuracy for the linear kernel function can be as high as 99.15% at several parameter selections. We choose one of them, i.e. parameter $c = 0.125$. Using this best parameter c, we determine the SVM to be used in the Internet Experiments.

C. Internet Experiments

We use the SVM to differentiate the TCP algorithms of the top 5000 web servers according to the Alexa traffic rank [20] and successfully get the congestion window traces of 1842 web servers. For those web servers without valid congestion window traces, the main reason is that we cannot obtain the congestion window traces that are long enough. Overall, we found that 4.75% of the web servers use some kind of the delay-based TCP algorithms. We show three of such traces in Figure 5 to Figure 7.

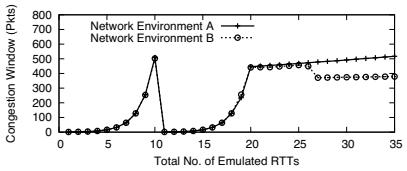


Fig. 5. A web server using TCP-Yeah. Web server indicated in HTTP headers: CacheFlyServe v26b. Operating System reported by NMAP [21]: Linux.

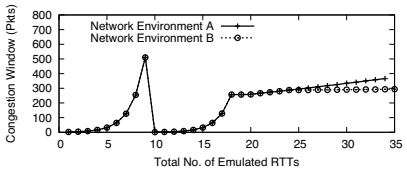


Fig. 6. A web server using TCP-Compound. Web server indicated in HTTP header: Microsoft-IIS/7.5. Operating System reported by NMAP: not available.

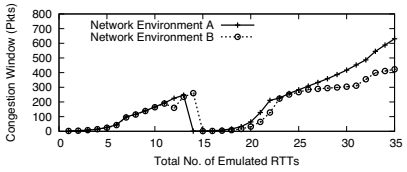


Fig. 7. A web server using a unknown delay-based TCP algorithm. Web server indicated in HTTP header: PWS/1.7.1.5. Operating System reported by NMAP: Linux.

We do find some unknown delay-based TCP algorithms. We show one in Figure 7.

VII. Conclusion

In this paper, we present our method DCAAD for differentiating between the delay-based and the non-delay-based TCP algorithms. DCAAD infers key features of a TCP algorithm according to the congestion window traces collected from a web server and then classifies the TCP algorithm using a SVM and the features inferred.

We evaluate DCAAD using local test-bed cross validation, which shows that DCAAD can achieve good differentiation accuracy. Using DCAAD, we conduct the Internet experiments to differentiate the TCP algorithms used by the web servers in the Internet. The result indicates that the percentage of the web servers using the delay-based TCP algorithms is very small. Thus, we believe that the impact of the delay-based TCP algorithms on the Internet queuing delay is small and the multimedia traffic controlled by the delay-based TCP algorithms probably won't experience a big improvement in terms of the queuing delay.

References

[1] L. Brakmo, S. O'Malley, and L. Peterson, "TCP Vegas: New techniques for congestion detection and avoidance," in *Proceedings of ACM SIGCOMM*, August 1994, pp. 24–35.

[2] K. Tan, J. Song, Q. Zhang, and M. Sridharan, "A compound TCP approach for high-speed and long distance networks," in *Proceedings of IEEE INFOCOM*, Barcelona, Spain, April 2006.

[3] S. Liu, T. Basar, and R. Srikant, "TCP-Illinois: A loss and delay-based congestion control algorithm for high-speed networks," in *Proceedings of International Conference on Performance Evaluation Methodologies and Tools (VALUETOOLS)*, Pisa, Italy, October 2006.

[4] C. Fu and S. Liew, "TCP Veno: TCP enhancement for transmission over wireless access networks," *IEEE Journal on Selected Areas in Communication*, vol. 21, no. 2, pp. 216–228, February 2003.

[5] A. Baiocchi, A. Castellani, and F. Vacirca, "YeAH-TCP: Yet another highspeed TCP," in *Proceedings of the fifth PFLDNET Workshop*, Los Angeles, CA, February 2007.

[6] V. Jacobson, "Congestion avoidance and control," in *Proceedings of ACM SIGCOMM*, Stanford, CA, August 1988.

[7] S. Ha, I. Rhee, and L. Xu, "Cubic: A new tcp-friendly high-speed tcp variant," *ACM SIGOPS Operating System Review*, vol. 42, no. 5, pp. 64–74, July 2008.

[8] L. Xu, K. Harfoush, and I. Rhee, "Binary increase congestion control for fast long-distance networks," in *Proceedings of IEEE INFOCOM*, Barcelona, Spain, April 2006.

[9] S. Floyd, "HighSpeed TCP for large congestion windows," *RFC 3649*, December 2003.

[10] R. N. Shorten and D. J. Leith, "H-TCP: TCP for high-speed and long-distance networks," in *Proceedings of the Second PFLDNet Workshop*, Argonne, IL, February 2004.

[11] T. Kelly, "Scalable TCP: Improving performance in highspeed wide area networks," *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 2, pp. 83–91, April 2003.

[12] C. Casetti, M. Gerla, S. Mascolo, M. Y. Sanadidi, and R. Wang, "TCP Westwood: Bandwidth estimation for enhanced transport over wireless links," in *Proceedings of ACM Mobicom*, Rome, Italy, July 2001.

[13] J. Padhye and S. Floyd, "On inferring TCP behavior," in *Proceedings of ACM SIGCOMM*, San Diego, CA, August 2001.

[14] D. Comer and J. Lin, "Probing TCP implementations," in *Proceedings of USENIX Summer Conference*, Boston, MA, June 1994.

[15] P. Yang, W. Luo, L. Xu, J. Deogun, and Y. Lu, "TCP congestion avoidance algorithm identification," in *Proceeding of IEEE ICDCS*, Minneapolis, MN, June 2011.

[16] S. Hemminger, "Network emulation with NetEm," in *Proceedings of the 6th Australia's National Linux Conference*, Australia, April 2005.

[17] C. Caini and R. Firrincieli, "TCP-Hybla: A TCP enhancement for heterogeneous networks," *International Journal of Satellite Communications and Networking*, vol. 22, no. 5, pp. 547–566, September 2004.

[18] A. Kuzmanovic and E. W. Knightly, "TCP-LP: A distributed algorithm for low priority data transfer," in *Proceedings of IEEE INFOCOM*, San Francisco, April 2003.

[19] C.-C. Chang and C.-J. Lin, "LIBSVM: a library for support vector machines." [Online]. Available: http://www.csie.ntu.edu.tw/~cjlin/papers/libsvm.pdf

[20] Alexa Internet Inc. The top 1,000,000 sites on the web. [Online]. Available: http://www.alexa.com/topsites

[21] Network Mapper (NMAP). [Online]. Available: http://nmap.org/