# PathRakeTCP: Available Bandwidth Estimation Using Multiple TCP Connections for Passing through Firewalls

Naoyuki Ito
*Nagoya University*
Japan

Takashi Oshiba
*NEC Corporation*
Japan

Kozo Satoda
*NEC Corporation*
Japan

Tutomu Murase
*Nagoya University*
Japan

***Abstract*—In this paper, we propose PathRakeTCP, a method to estimate available bandwidth even if there are firewalls on the communication path that reject UDP communication. PathRakeTCP is one of the so-called packet train methods that estimates the available bandwidth by transmitting multiple probing packets. Firewalls often allow TCP communication to pass through. PathRakeTCP can pass through the firewalls because the packet train is composed of TCP packets. Many conventional packet train methods use UDP packets, which allow precise control over the transmission timing of each probing packet. This is because precise transmission timing control is essential for accurate available bandwidth estimation. On the other hand, if a packet train of TCP packets is simply transmitted, the transmission timing of each probing packet will be disrupted by TCP congestion control. This is a new problem not encountered in the conventional methods that use UDP packets. To solve the above problem, PathRakeTCP establishes a TCP connection for each probing packet and transmits only one packet at each TCP connection. To demonstrate that PathRakeTCP can precisely control the transmission timing of each TCP packet, an experimental evaluation is conducted with a real testbed. The experimental results show that the estimation error of PathRakeTCP is comparable to that of the packet train method with UDP packets.**

***Keywords—available bandwidth estimation, packet train, TCP congestion control, firewall***

## I. Introduction

The COVID-19 pandemic has increased the use of online video conferencing applications, resulting in a trend of increasing network congestion [1]. It is likely that these applications will continue to be used even after the pandemic is over. Even under such circumstances, network administrators in universities or enterprises are required to provide a comfortable network environment for users. To this end, it is useful to periodically monitor the available bandwidth [2] (i.e., unused capacity of an end-to-end path) over a long period of time. The collected available bandwidth data can be used to identify congested network paths for troubleshooting when congestion occurs. Additionally, the data can be used for the efficient future renewal of network infrastructure. To monitor the available bandwidth, packet train methods, in which multiple probing packets are transmitted, are useful. One of the packet train methods we have developed in the past is PathQuick3 [3]. PathQuick3 can estimate the available bandwidth accurately, quickly and with low network load. Most conventional methods, including PathQuick3, use UDP packets to form a packet train. However, a firewall, which is one of the typical middleboxes [4] in university or enterprise networks [5], often rejects UDP communication [6][7]. Thus, network administrators often cannot monitor the available bandwidth due to the firewall.

In this paper, we propose a packet train method, PathRakeTCP, which can estimate the available bandwidth even if there are firewalls on the communication path that reject UDP communication. Firewalls typically allow some TCP communications to pass through. PathRakeTCP can pass through firewalls because it configures packet trains with TCP packets on port numbers that are allowed to pass through. Network administrators, who are typical users of PathRakeTCP, can use it frequently to estimate network environments such as workplaces and cafeterias to identify where and when network congestion is occurring.

Section II describes the conventional methods, Section III describes the problems of the conventional methods, Section IV outlines the PathRakeTCP mechanism by comparing it with PathQuick3, Section V evaluates the estimation error of PathRakeTCP using experiments with a real testbed, Section VI discusses an intrusion detection system (IDS), an intrusion prevention system (IPS) and PathRakeTCP and Section VII provides a conclusion and future work.

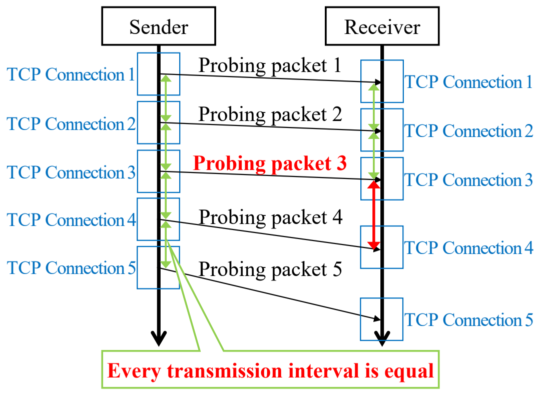

Fig. 1. Overview of PathRakeTCP. TCP connections are established for each probing packet, and only one packet is transmitted on each TCP connection to precisely control the transmission timing of each probing packet.

## II. Conventional Packet Train Methods and the Conventional Work of Middleboxes

### *A. Principles of Available Bandwidth Estimation in the Conventional Packet Train Methods*

Conventional packet train methods such as pathChirp [8], Pathload [9] and PathQuick3 transmit multiple probing packets consisting of UDP packets and estimate the available bandwidth based on the principle described below.

If the probing rate at the sender of a packet train is less than the available bandwidth, there is no queuing delay for the probing packet at the router or switch that is the bottleneck on the communication path. Therefore, the receiving interval of the probing packet of the packet train at the receiver is equal to the transmission interval at the sender.

However, if the probing rate exceeds the available bandwidth, queuing delays for the probing packets occur at the router or switch at the bottleneck location. As a result, the receiving interval of the probing packets at the receiver begins to be longer than the transmission interval at the sender. The receiver finds the probing packet at which the receiving interval begins to increase, and uses the probing rate of the immediately previous probing packet as an estimated available bandwidth.

### *B. Mechanism for Estimating Available Bandwidth in PathQuick3*

PathQuick3 (1) linearly increases the probing rate for each probing packet in a single packet train; (2) finds the probing rate of the probing packet at which the queuing delays begin to occur, and uses the probing rate of the probing packet immediately before that point as an estimated available bandwidth. The mechanisms of (1) and (2) are explained below.

(1) As shown in Fig. 2, the per-packet probing rate is linearly increased by keeping the transmission interval constant and by linearly increasing the packet size.

(2) In Fig. 3, it is assumed that the per-packet probing rate exceeds the actual available bandwidth for the first time at probing packet 4. In this case, since no queuing delay occurs up to probing packet 3, the receiving interval is equal to the transmission interval. On the other hand, since the probing rate of probing packet 4 exceeds the available bandwidth for the first time, a queuing delay occurs at the router or switch, which is the bottleneck on the communication path. As a result, the receiving interval is longer than the transmission interval. Since probing packet 4 is the packet at which the receiving interval begins to increase for the first time, the probing rate of probing packet 3 is used as the estimated available bandwidth.

Note that we can adjust the maximum probable bandwidth, the minimum probable bandwidth and the resolution of estimation by changing the packet transmission interval, the packet size of the first probing packet, the increased amount of the packet size and the number of probing packets. The above maximum and minimum probable bandwidth and resolution can be derived from Equation (3) in [3], which is a formula for calculating the probing rate for each probing packet.

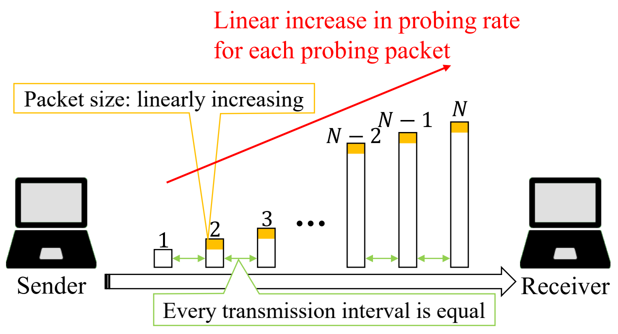
Linear increase in probing rate
for each probing packet
Packet size: linearly increasing
1
2
3
$N-2$
$N-1$
$N$
Sender
Receiver
Every transmission interval is equal

Fig. 2. Linear increase in probing rate in PathQuick3.

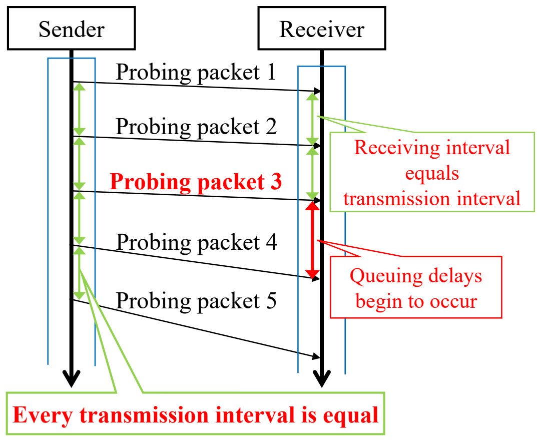
Sender
Receiver
Probing packet 1
Probing packet 2
Probing packet 3
Probing packet 4
Probing packet 5
Receiving interval equals transmission interval
Queuing delays begin to occur
Every transmission interval is equal

Fig. 3. Increasing the probing rate causes a queuing delay. The sender of PathQuick3 utilizes a single UDP socket to transmit all the probing packets.

### *C. Conventional Work of Middleboxes*

A firewall and an IDS/IPS are typical middleboxes that inhibit communication. Most of the research on these middleboxes concern security issues, and few studies have examined the relationship between network estimation, which is the subject of this paper, and these middleboxes.

First, to safely pass through a firewall, [10] proposes a trust model of cloud computing that can be controlled in the presence of firewalls without being inhibited by them. However, it is far from a network estimation.

Second, to evade an IDS/IPS, [11] surveys IDS/IPS evasion techniques and tests each technique in a real-world environment. However, these evasion techniques are from the attacker's perspective and do not extend to network estimation.

Other studies related to an IDS/IPS include [12] and [13] but they are concerned with IDS/IPS attack detection performance from a security point of view and not with passing through an IDS/IPS. [12] used machine learning to support an IDS/IPS to detect distributed denial-of-service (DDoS) attacks. [13] proposed a method to detect TCP SYN flood attacks, a type of DDoS attack, by detecting anomalous traffic patterns.

## III. Problems with Conventional Methods

### *A. Packet Train Methods Using UDP Packets*

Conventional packet train methods have the problem that they cannot pass through firewalls. As described in Section I, conventional packet train methods using UDP probing packets cannot estimate the available bandwidth because the UDP probing packets are rejected by the firewall, as shown in Fig. 4.

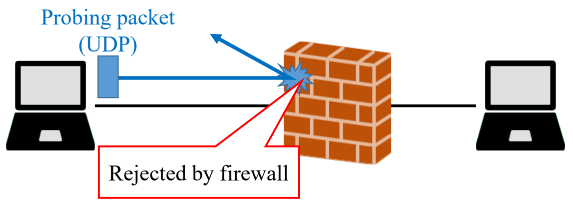

Fig. 4. The UDP packet is rejected by the firewall, and thus, the available bandwidth cannot be estimated.

## B. Packet Train Methods Using TCP Packets

The conventional methods using UDP packets cannot be simply applied to TCP packets. The reason is, that simply transmitting a packet train of TCP packets similar to UDP packets causes a new problem not seen in the conventional methods, the transmission timing of each probing packet is disrupted by TCP congestion control. The congestion window size at the beginning of TCP communication is either 2, 4, or 10 [14], and the number of probing packets exceeds the congestion window size because it is common for the packet train method to transmit several dozen or more probing packets. As shown in Fig. 5, when the number of probing packets sent already reaches the congestion window size, the next probing packet cannot be transmitted until the ACK packet is returned to the sender. Therefore, the transmission timing is disturbed by TCP congestion control, and thus the estimation accuracy of the available bandwidth is heavily degraded because the preciseness of the transmission timing is essential in the packet train methods.

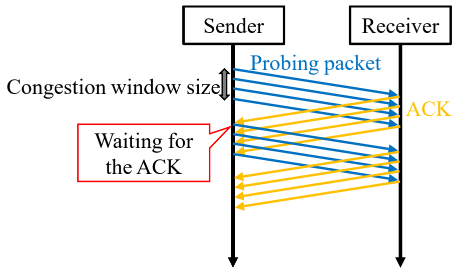

Fig. 5. The transmission timing of each probing packet is disrupted by the TCP congestion control.

## C. Available Bandwidth Estimation Methods Using TCP Communication

ImTCP [15] and [16][17][18] are available bandwidth estimation methods using TCP communication. However, ImTCP requires modification of the TCP stack of OS/kernel. Therefore, network administrators in universities or enterprises cannot easily use ImTCP in an out-of-box manner.

Note that the methods in [16][17][18] are passive measurement methods that capture and analyze packets from general users. Namely, these methods are not packet train methods, in which network administrators actively transmit probing packets to monitor the available bandwidth.

## IV. Proposal of PathRakeTCP, an Available Bandwidth Estimation Method for Passing through Firewalls

Here, we propose a method for estimating the available bandwidth, PathRakeTCP[1], which solves the following three problems of conventional methods: (A) passing through firewalls, (B) precise control of transmission timing, and (C) ease of use.

For (A), the firewall can be passed through by using TCP communication. Network administrators have access to the firewall settings in the network they manage. Therefore, they know the TCP port numbers that can pass through their firewalls, and thus PathRakeTCP can use these port numbers to pass through firewalls and estimate the available bandwidth.

For (B), PathRakeTCP establishes a TCP connection for each probing packet, and each TCP connection transmits only one packet. Since the congestion window size at the start of TCP communication is either 2, 4, or 10, the number of probing packets never exceeds the congestion window size, and thus the transmission timing of each probing packet is not affected by the TCP congestion control. Therefore, the transmission timing of each probing packet can be precisely controlled.

Regarding (C), PathRakeTCP does not require modification of the TCP stack of the OS/kernel and can be implemented on user land and therefore can be easily used by network administrators in an out-of-box manner.

In summary, to the best of our knowledge, *PathRakeTCP is the first packet train method in the literature that can solve all the problems of (A), (B) and (C) above*.

We have implemented PathRakeTCP as a user land application using C language packages for Linux. We have used Cygwin [19] to work PathRakeTCP on Windows OS.

## A. Available Bandwidth Estimation Method Using Multiple TCP Connections

PathRakeTCP can control the transmission timing of each probing packet with the same preciseness as the conventional UDP method. This is illustrated in Fig. 1 and Fig. 3 as a comparison of PathRakeTCP and PathQuick3. When a sender transmits multiple probing packets as a packet train, the sender of PathRakeTCP utilizes multiple TCP connections in Fig. 1, while the sender of PathQuick3 utilizes a single UDP socket in Fig. 3. As shown in Fig. 1, PathRakeTCP can transmit each probing packet at an equal transmission interval without being affected by TCP congestion control. This is because the number of probing packets does not exceed the congestion window size since a TCP connection is established for each packet, as explained in (B) at the beginning of Section IV. Therefore, comparing Fig. 1 and Fig. 3, it can be seen that the transmission of probing packets can be controlled in exactly the same way from the viewpoint of transmission timing. This means that PathRakeTCP can control the transmission timing of the TCP

[1] The *Rake* in PathRakeTCP is derived from the resemblance of the shape of the teeth of a rake to the way probing packets are transmitted from multiple TCP connections in a comb-like fashion.

probing packets as precisely as PathQuick3 with the UDP probing packets.

## V. Comparative Experimental Evaluation of Estimation Error Using a Real Testbed

Although PathRakeTCP and PathQuick3 use different transport protocols, they have the same packet train structure. Therefore, if PathRakeTCP can control the transmission timing of each probing packet with the same preciseness as PathQuick3, the estimation error of both methods is expected to be approximately the same (strictly speaking, the header size of the TCP and UDP packets is different but this difference has a negligible impact on the estimation error). Therefore, we conduct an experimental evaluation to compare the estimation error of both methods using a real testbed.

### A. Experimental Setup

Fig. 6 shows the equipment and network topology of the real testbed for the comparative experimental evaluation. This experiment assumes a situation in which a packet train of PathRakeTCP or PathQuick3 is transmitted from a PC for sending a packet train to a PC for receiving a packet train while other users are communicating. The other users' communication is simulated by the UDP cross-traffic transmitted from a PC for sending the cross-traffic to a PC for sending the cross-traffic using iperf (one of the standard tools for network performance measurement). The cross-traffic rate is $c$ Mbps. While varying $c$ from 0 Mbps to 90 Mbps in 10 Mbps increments, the packet trains are transmitted, and the estimated available bandwidth is recorded.

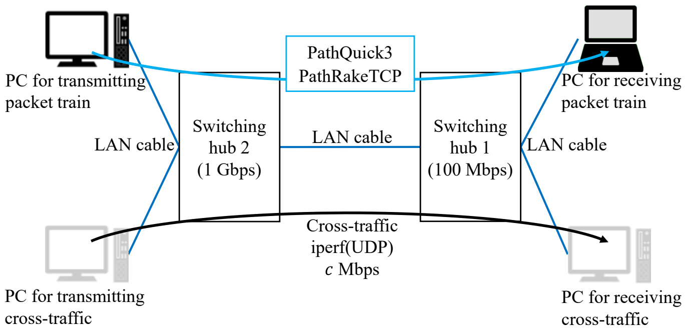
PathQuick3
PathRakeTCP
PC for transmitting packet train
PC for receiving packet train
LAN cable
Switching hub 2 (1 Gbps)
LAN cable
Switching hub 1 (100 Mbps)
LAN cable
Cross-traffic iperf(UDP) c Mbps
PC for transmitting cross-traffic
PC for receiving cross-traffic

Fig. 6. Equipment and network topology of the real testbed for the comparative experimental evaluation.

The actual available bandwidth, i.e., the ground truth, is

$$\text{Actual available bandwidth} = \text{Physical capacity at bottleneck} - c \quad (1)$$

The estimation error is

$$\text{Estimation error} = \text{Actual available bandwidth} - \text{Estimated available bandwidth} \quad (2)$$

We quantitatively compare PathRakeTCP with PathQuick3 by using the mean absolute error (MAE), which is the mean of the estimation errors.

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N} |(\text{Estimation error})_i| \qquad (3)$$

where $N$ is the total number of estimations.

The physical capacity of switching hub 1, which is the bottleneck point, is 100 Mbps. The reason for setting the physical capacity of the bottleneck at 100 Mbps in this experiment is that we assume that the network administrators provide a network environment in which the users can comfortably use video distribution services and video conferencing services (representative examples of such network applications and their required bandwidth are shown in Table I), which requires a broadband network among the various network applications.

TABLE I. Bandwidth Required for Each Network Application.

| **Application name** | **Required bandwidth (Maximum value)** |
|---|---|
| YouTube | 20 Mbps [20] |
| Netflix | 15 Mbps [21] |
| Zoom | 4 Mbps [22] |
| Microsoft Teams | 4 Mbps [23] |

The transmission interval, packet size of the first packet, increase in packet size and number of probing packets are 0.1 ms, 32 Bytes, 12 Bytes and 120 packets, respectively, and the minimum and maximum probable bandwidths calculated from these values are 2.6 Mbps and 116.0 Mbps, respectively (for the calculation method described in [3]). The reason for adopting these values is that the minimum and maximum probable bandwidth can include all the actual available bandwidths, i.e., 10, 20, …, 100 Mbps. Another reason is to have a resolution of approximately 1 Mbps, which is considered sufficient for network administrators to monitor the available bandwidth.

PathRakeTCP requires the same number of TCP connections as the number of probing packets (i.e., 120 in this case) to be established before transmitting the packet train. This process is completed instantly on the PC used to send and receive the packet train (Table II shows the detailed specifications), so PathRakeTCP can immediately transmit the packet train without a long waiting time for the establishment of multiple TCP connections (we will discuss the multiple connection establishment process in detail in Section VI).

TABLE II. The Spec of the PCs for Transmitting and Receiving Packet Trains.

| | **PC for transmitting packet trains** | **PC for receiving packet trains** |
|---|---|---|
| CPU | AMD Ryzen 5 PRO 4650G with Radeon Graphics 3.70 GHz | Intel Core i5-10210U CPU @ 1.60 GHz 2.11 GHz |
| Memory | 32 GBytes | 16 GBytes |
| OS | Windows 10 Home | Windows 10 Home |

### B. Experimental Results of Estimation Error

Fig. 7 and Fig. 8 show the estimation results for PathRakeTCP and PathQuick3, respectively. Each dot in the figure represents the median of the estimated available bandwidth when varying the cross-traffic (the median value of five estimation runs for each cross-traffic $c$). When the estimated available bandwidth exactly matches the actual available bandwidth, a dot is plotted on a straight line at an angle of 45°.

The shorter the distance between this diagonal line and the dot (the length of the two-way arrow in Fig. 7), the smaller the estimation error. Fig. 7 and Fig. 8 show that the estimation error is comparable between the two methods.

Next, we compare the estimation errors quantitatively. We transmit packet trains five times for each of the 10 cross-traffic rates $c$, i.e., 50 times in total ($N = 50$ in Equation (3)). The MAE of PathRakeTCP is 15.6 Mbps (the ratio of MAE to the physical capacity at the bottleneck is $15.6/100 = 15.6\ \%$) and that of PathQuick3 is 18.2 Mbps (the ratio of MAE to the physical capacity at the bottleneck is $18.2/100 = 18.2\ \%$). The MAEs of the two are approximately the same. This confirms that PathRakeTCP can precisely control the timing of the TCP packet transmission without being disturbed by TCP congestion control.

## VI. Discussion of an IDS/IPS and PathRakeTCP

As described in Section IV, PathRakeTCP can pass through a firewall. Here, we discuss the ability of PathRakeTCP to pass through an IDS/IPS, another typical middlebox. In the practical use of PathRakeTCP, since PathRakeTCP establishes many TCP connections, we need to consider the possibility that IDS/IPS may misunderstand it as a DoS/DDoS attack and reject the communication. However, as shown below, such a possibility is considered to be low because the traffic patterns of the PathRakeTCP and DoS/DDoS attacks can be quite different.

Many IDSs/IPSs detect attacks by matching new packets against a signature database using signatures defined from the logs of previous attacks [11]. Therefore, the more different the traffic pattern of PathRakeTCP is from that of a DoS/DDoS attack, the less likely it is that PathRakeTCP will be misunderstood as a DoS/DDoS attack. The type of DoS/DDoS attack in which PathRakeTCP is misunderstood would be the TCP SYN flood attack [13], which (1) transmits many SYN packets in a short period of time but (2) does not return an ACK packet in the 3-way handshake in the connection establishment process, resulting in leaving a large number of TCP connections half-open.

The traffic pattern of PathRakeTCP can be quite different from that of a TCP SYN flood attack as follows. First, when PathRakeTCP starts to establish the same number of TCP connections as the number of probing packets between a sender and a receiver, instead of transmitting all SYN packets at the same time, the sender inserts a small amount of waiting time between transmitting each SYN packet. By setting the waiting time longer than the duration, the IDS/IPS will not misunderstand as a TCP SYN flood attack, and the above (1) can be prevented. Network administrators have access to the IDS/IPS settings in the network they manage. Therefore, they know the appropriate waiting time based on the IDS/IPS settings. Next, because the sender returns an ACK packet immediately after receiving the SYN-ACK packet from the receiver in the 3-way handshake, the above (2) can be prevented.

Furthermore, PathRakeTCP is unlikely to be misunderstood as a DDoS attack from the viewpoint of IP address spoofing, since the IP address is the same for all the packets and is not a spoofed IP address.

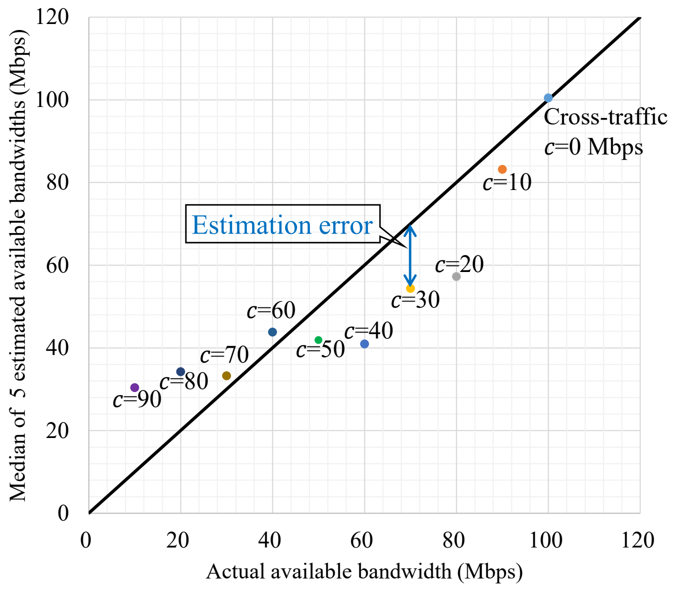

Fig. 7. Estimation results of PathRakeTCP.

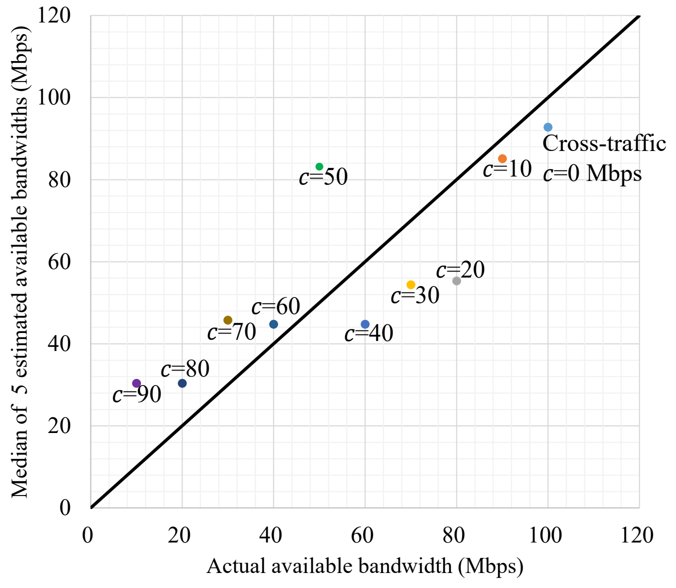

Fig. 8. Estimation results of PathQuick3.

## VII. Conclusion and Future Work

In this paper we propose PathRakeTCP, which can estimate the available bandwidth even if there is a firewall on the communication path that rejects UDP communication. PathRakeTCP establishes a TCP connection for each probing packet in the packet train and transmits only one packet on each TCP connection.

The main research contributions of this paper are threefold:

(1) Firewalls often allow TCP communications to pass through, and PathRakeTCP can pass through the firewalls because it consists of TCP packets in a packet train.

(2) Since a TCP connection is established for each probing packet, the transmission timing of each probing packet is not disrupted by the TCP congestion control, and hence PathRakeTCP can precisely control the transmission timing of each probing packet.

(3) The TCP stack in the OS/kernel does not need to be modified, and PathRakeTCP can be implemented on user land, making it easy to use in an out-of-box manner.

To the best of our knowledge, *PathRakeTCP is the first packet train method in the literature that has all the advantages of (1), (2) and (3) above*.

To demonstrate that PathRakeTCP can precisely control the transmission timing of each probing packet, we evaluate the estimation error of PathRakeTCP using a real testbed. As a result, the estimation error of PathRakeTCP is approximately the same as that of PathQuick3, a packet train method for UDP probing packets that we have developed in the past, and thus, we confirm that PathRakeTCP can precisely control the transmission timing of each TCP probing packet.

Although a fully wired network was used in our real testbed, wireless networks, e.g., wireless LAN (Wi-Fi), are widely used in modern campus and enterprise networks. Additionally, preliminary experimentation results with a small testbed are shown in this paper. Also, the baseline method in this paper, PathQuick3, is the method that we have developed in the past. Therefore, in future work, we will conduct large-scale experiments over wireless campus and enterprise networks to compare the estimation accuracy of the various packet train methods developed by third-party research groups and PathRakeTCP.

Our investigation into the complex behavior of firewalls (such as stateful and deep packet inspection [24]), IDSs/IPSs, and other types of middleboxes is still in progress. Additionally, we have not yet confirmed that PathRakeTCP can pass through operational middleboxes in the wild. Consequently, we also plan to conduct large-scale experiments over operational campus and enterprise networks to confirm that PathRakeTCP can pass through various types of firewalls, IDS/IPS and other types of middleboxes.

## REFERENCES

[1] M. S. Elsayed, N. A. Le-Khac, and A. D. Jurcut, “Dealing with COVID-19 network traffic spikes,” *IEEE Security & Privacy*, Vol. 19, Issue 1, pp. 90–94, 2021.

[2] R. Prasad, C. Dovrolis, M. Murray, and K. Claffy, “Bandwidth estimation: Metrics, measurement techniques, and tools,” *IEEE Network*, Vol. 17, Issue 6, pp. 27–35, 2003.

[3] T. Oshiba, K. Nogami, K. Nihei, and K. Satoda, “Robust available bandwidth estimation against dynamic behavior of packet scheduler in operational LTE networks,” *IEEE Symposium on Computers and Communication (ISCC)*, pp. 1276–1283, 2016.

[4] B. Carpenter and S. Brim, “Middleboxes: Taxonomy and issues,” IETF RFC 3234, 2002.

[5] J. Sherry and S. Ratnasamy, “A survey of enterprise middlebox deployments,” Technical Report, UCB/EECS-2012-24, University of California at Berkeley, 2012.

[6] T. Reddy, P. Patil, D. Wing, and B. V. Steeg, “WebRTC UDP firewall traversal,” *IAB Workshop on Stack Evolution in a Middlebox Internet (SEMI)*, 2015.

[7] H. Y. Yang, K. H. Lee, and S. J. Ko, “Communication quality of voice over TCP used for firewall traversal,” *IEEE International Conference on Multimedia and Expo (ICME)*, pp. 29–32, 2008.

[8] V. J. Ribeiro, R. H. Riedi, R. G. Baraniuk, J. Navratil, and L. Cottrell, “pathChirp: Efficient available bandwidth estimation for network paths,” *Passive and Active Measurement (PAM) workshop*, 2003.

[9] M. Jain and C. Dovrolis, “End-to-end available bandwidth: Measurement methodology, dynamics, and relation with TCP throughput,” *ACM SIGCOMM*, pp. 295–308, 2002.

[10] Z. Yang, L. Qiao, C. Liu, C. Yang, and G. Wan, “A collaborative trust model of firewall-through based on cloud computing,” *International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pp. 329–334, 2010.

[11] H. Kılıç, N. S. Katal, and A. A. Selçuk, “Evasion techniques efficiency over the IPS/IDS technology,” *International Conference on Computer Science and Engineering (UBMK)*, pp. 542–547, 2019.

[12] J. D. Ndibwile, A. Govardhan, K. Okada, and Y. Kadobayashi, “Web server protection against application layer DDoS attacks using machine learning and traffic authentication,” *IEEE Annual Computer Software and Applications Conference (COMPSAC)*, pp. 261–267, 2015.

[13] S. H. C. Haris, R. B. Ahmad, and M. A. H. A. Ghani, “Detecting TCP SYN flood attack based on anomaly detection,” *International Conference on Network Applications, Protocols and Services (NETAPPS)*, pp. 240–244, 2010.

[14] J. Chu, N. Dukkipati, Y. Cheng, and M. Mathis, “Increasing TCP’s initial window,” IETF RFC 6928, 2013.

[15] C. L. T. Man, G. Hasegawa, and M. Murata, “ImTCP: TCP with an inline measurement mechanism for available bandwidth,” *Computer Communications (COMCOM)*, pp. 1614–1626, 2006.

[16] F. Ciaccia, I. Romero, O. A. Abella, D. Montero, R. S. Gracià, and M. Nemirovsky, “SABES: Statistical available bandwidth estimation from passive TCP measurements,” *IFIP Networking Conference*, pp. 743–748, 2020.

[17] S. K. Khangura and M. Fidler, “Available bandwidth estimation from passive TCP measurements using the probe gap model,” *IFIP Networking Conference and Workshops*, pp. 1–9, 2017.

[18] M. Zangrilli and B. B. Lowekamp, “Applying principles of active available bandwidth algorithms to passive TCP traces,” *International Workshop on Passive and Active Network Measurement (PAM)*, pp. 333–336, 2005.

[19] Cygwin, “Cygwin. Get that Linux feeling - on Windows,” https://www.cygwin.com/, (accessed in November 8, 2022).

[20] Google, “System requirements,” (in Japanese) [online] https://support.google.com/youtube/answer/78358?hl=ja, (accessed in April 15, 2022).

[21] Netflix, “Internet connection speed recommendations,” (in Japanese) [online] https://help.netflix.com/ja/node/306, (accessed in April 15, 2022).

[22] Zoom, “Zoom system requirements: Windows, macOS, Linux,” https://support.zoom.us/hc/en-us/articles/201362023-Zoom-system-requirements-Windows-macOS-Linux, (accessed in April 15, 2022).

[23] Microsoft, “Prepare your organization's network for Microsoft Teams,” (in Japanese) [online] https://docs.microsoft.com/ja-jp/microsoftteams/prepare-network, (accessed in April 15, 2022).

[24] Y. Guo, C. Wang, and X. Jia, “Enabling secure and dynamic deep packet inspection in outsourced middleboxes,” *International Workshop on Security in Cloud Computing (SCC)*, ACM, pp. 49–55, 2018.