PerfTrace: A New Multi-metric Network Performance Monitoring Tool

Yaozhong Liu, Long Pan, Chenglong Li†‡, Lin He†‡, Yirui Luo, Guanglei Song, Jiahai Yang†‡, Zhiliang Wang†‡
Institute for Network Sciences and Cyberspace, BNRist, Tsinghua University, Beijing, China
†Zhongguancun Laboratory, Beijing, China
‡Quan Cheng Laboratory, Jinan, Shandong, China

***Abstract*—We present `PerfTrace`, an end-to-end tool for efficient, real-time, and multi-metric network performance monitoring. `PerfTrace` provides a high integration of different existing measurement functions, supporting the measurement of essential metrics such as latency, jitter, packet loss, and available bandwidth. More importantly, innovative schemes and algorithms are proposed to address the weaknesses of existing tools.**

After conducting comprehensive evaluations, we find that (i) `PerfTrace` measures one-way and two-way latency, jitter, and packet loss ∼9.4× faster and ∼3.6× more data-efficiently; (ii) `PerfTrace` measures available bandwidth in our testbed with minimal mean relative error (5.22%), outperforming all the tools compared (ranging from 8.17% to 37.24%). Meanwhile, `PerfTrace` consumes a more constant percentage of bandwidth resources than other tools when monitoring available bandwidth. `PerfTrace`'s data overhead is always only about 1/600 of the total bandwidth for a measurement frequency once per minute.

I. Introduction

Network performance refers to the overall measures of network quality of service as perceived by users. With the rapid growth of networks, network performance monitoring has become more critical than ever, especially in the following scenarios: (i) Networking researchers need to evaluate the performance of new network protocols and algorithms based on specific performance metrics [1]. (ii) Internet service providers (ISPs) need to monitor the real-time network performance to detect, diagnose and fix problems [2]. (iii) Internet technology companies need to monitor their service access delay, which is strongly related to the company profits [3]. (iv) Customers compare the network performance of different commercial cloud services to make choices [4].

Network performance measurements can be dated back to the 1970s when ARPANET initiated several projects on network behavior [5]. After decades of development, researchers have proposed many approaches to monitor network performance. Network measurements usually fall into two categories: passive and active measurements [6]. Passive measures analyze the performance based on user application traffic without generating extra artificial traffic to the networks. However, passive measurements rely on collecting the traffic and usually require administrator privileges. In comparison, active measures, which push probe packets specifically designed into the network and then analyze network performance, are easier to deploy and perform. Since the probe traffic may interfere with user traffic, probe packets should be properly scheduled to minimize their impact. According to the location of vantage points, network measurements can also be divided into end-to-end measurements and interior measurements [7]. Interior measurements require specialized forwarding equipment (e.g., programmable switches [8]), limited by network type and network size [9].

Among the above categories, active end-to-end measurements are popular as they are easy to deploy, including commonly-used tools `ping` and `iPerf`. However, when it comes to performance monitoring, problems and weaknesses of those tools emerge: (i) A single measurement usually takes a long time, i.e., dozens of seconds. (ii) Existing approaches cannot adapt well to the new network environments. For instance, ICMP filtering [10], which is common for network devices nowadays, reduces the availability of those tools like `ping` and `traceroute`. (iii) Some measurement tools or methods can be intrusive, e.g., `iPerf` and `FastBTS` [11] measure the TCP bulk transfer capacity (BTC) with TCP flooding, not considered to be a good choice for long-lasting performance monitoring. (iv) Existing tools usually focus on one metric, which means a combination of tools is required to monitor network performance comprehensively, potentially leading to increased network pressure.

To address those challenges, we propose `PerfTrace`, an active end-to-end tool for performance monitoring. `PerfTrace` provides a high integration of different functions of existing popular tools, and innovative schemes and algorithms are proposed (§IV) to address the above problems and weaknesses. In detail, `PerfTrace` measures various performance metrics with probe traffic that is as less as possible.

This paper presents the methodology, design, and implementation of `PerfTrace`, which includes the following features:

- **Packet multiplexing.** `PerfTrace` is designed with the idea of packet multiplexing, i.e., one probe packet for several metrics, which covers more performance metrics with less traffic, making the measurements less intrusive.

This work is supported by Tsinghua University Initiative Scientific Research Program (2021Z11GHX010) and Beijing Natural Science Foundation under Grant No.4222026. Corresponding Author: Chenglong Li (email: lichenglong@tsinghua.edu.cn).

- **Better bandwidth measurement algorithm.** We design and implement a new available bandwidth measurement algorithm in PerfTrace, which is better than the existing tools involved in the comparison in terms of adaptability, accuracy, and real-time performance.
- **Flexible expandability.** The light-weight PerfTrace is built with a generic framework and flexible expandability, making it easy for community developers to extend new measurement schemes. We hope our work is just the beginning.

PerfTrace is open-source and can be found at https://github.com/LiuNotTwo/PerfTrace. The rest of the paper is structured as follows. We discuss the necessary background and related work (§II) and present the design (§III). Then we evaluate PerfTrace and compare it with existing tools (§IV). Finally, we have a conclusion of our work (§V).

II. Background and Related Work

The quality and speed of data transmission are Internet users' two most essential concerns, the former indicating the stability and reliability and the latter indicating the ability of high-speed processing and transmission of the network devices. As for performance monitoring, the following metrics are considered most common and valuable: latency, packet loss rate, jitter, and available bandwidth.

We put these three metrics together: latency, packet loss rate, and jitter because we can measure them together in one measurement appropriately. By sending multiple probe packets resulting in multiple latencies, we can calculate the jitter and packet loss rate based on the latencies and the number of packets lost.

In §III, we introduce PerfTrace's implementations of latency, packet loss rate, and jitter measurements. Compared with these three well-known tools above-mentioned, PerfTrace measures more metrics with less traffic.

For a network link or path, available bandwidth refers to the unused part of its bandwidth capacity. When measuring available bandwidth, existing bandwidth measurement tools usually make the following assumptions: (i) probe traffic and cross traffic have the same priority; (ii) cross traffic and probe traffic transmit in the paths just in the same way as fluid (**fluid model** [12]). PerfTrace differs in that it no longer relies on the assumption (ii) and is, therefore, more adaptable to heterogeneous networks.

A. Taxonomy of Available Bandwidth Measurement Tools

We can roughly sort the existing tools for available bandwidth measurements into the following categories based on their traffic modes, measurement models and technologies (TABLE I).

Packet pair vs. packet train. We can divide the tools into two categories based on whether a single probe only sends two packets (packet pair) or multiple packets (much more than two packets, i.e., packet train). Generally, the noises easily affect packet pair-based measurements, while the packet train-based measures handle the noises better but are more intrusive. Therefore, the trade-off between robustness and intrusiveness remains a critical problem.

Probe gap model (PGM) vs. probe rate model (PRM). PGM infers the available bandwidth based on the difference in intervals of two packets. The idea behind PGM is that the interval between two packets will be affected by the cross traffic in the paths, so we can indirectly get the available bandwidth by perceiving the cross traffic rate using PGM. This technique is called packet pair/train dispersion (**PPTD**). Unlike PGM, PRM sends probe packets with a well-designed rate and estimates the bandwidth by observing whether the probe packets will cause **self-induced congestion** [12].

B. Accuracy, Overhead and Intrusiveness

Accuracy and Overhead. Time measurements remain the core of bandwidth measurements for all the existing tools ranging from the **PRM**-based tools to **PGM**-based tools. We hope that the arrival time of the probe packets will only be influenced by the cross traffic on the links when measuring bandwidth. However, the reality is that many other factors, namely the noises, can also make a difference. Noises include but are not limited to: (i) interrupt coalescing usually brings clock offsets of $\sim$100 microseconds [18]; (ii) virtual interrupts of cloud servers also result in millisecond errors [19]; (iii) complicated rate-limiting mechanism of ISPs may affect the sequential features of the probe packets.

As a typical example, we explain why Pathload does not work well in high-bandwidth cloud networks. Pathload sends a packet train consisting of 100 packets. Supposing each packet is 1500 bytes (cannot exceed the MTU), this sequence of packets will not last for more than 1.2 ms when the probing rate is 1 Gbps, thus can be easily affected by the noises of interrupt coalescing and virtual interrupts.

Overhead and Intrusiveness. People may take it for granted that high traffic of probe packets always lead to high intrusiveness. However, the relative amount of traffic is also an essential metric in addition to the absolute amount. For instance, 1MB traffic for 10Mbps links is much more intrusive than 10MB traffic for 1Gbps links. While other tools usually use the total traffic of probe packets as the representative of the intrusiveness, PerfTrace mainly takes the relative value into account because today's link bandwidth is becoming bigger.

III. Design of PerfTrace

This section discusses the design and implementation of PerfTrace. PerfTrace draws wisdom from previous work [20], [17] and proposes new algorithms and methods.

A. Architecture

Figure 1 presents an architectural overview of PerfTrace, which includes four modules: Local Controller, Probe Sender, Local Database, and Probe Responder.

Local Controller. Local Controller parses the arguments of monitoring tasks (measurement mode, packet count, etc.), determining the types, sizes, amount, and intervals of probe

TABLE I: Information about some typical bandwidth measurement tools.

Tools for AB	traffic mode [13]	model	technology
Spruce [12]	Packet pair	PGM (Link capacity required)	PPTD
Abing [14]	Packet pair	PGM	PPTD
Pathchirp [15]	Exponentially spaced packet train	PRM	self-induced congestion
Assolo [16]	Reflected exponentially spaced packet train	PRM	self-induced congestion
Pathload [17]	Uniformly spaced packet train	PRM	self-induced congestion

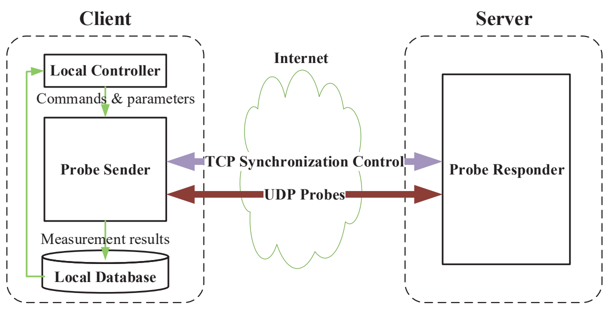


Fig. 1: An overview of PerfTrace.

packets to be sent. A timer will be set for a scheduled task and trigger Probe Sender.

Probe Sender. Being the de-facto performer of measurements, Probe Sender establishes TCP connection with target-side Probe Responder and synchronizes the parameters. After that, it sends fabricated probe packets to the open UDP port of the targets and receives corresponding replies.

Local Database. Local Database stores the measurement results of `PerfTrace`. The historical data can help determine the probe rate when measuring available bandwidth.

Probe Responder. Probe Responder opens particular TCP ports and waits for measurement tasks. Once a task is received, it will open specific UDP ports and inform the Probe Sender.

We introduce a complete workflow of a `PerfTrace` measurement task. Local Controller parses the task arguments and starts corresponding measurements. Probe Sender then generates sequences of fabricated probe packets based on related parameters. As for the measurements of available bandwidth, several iterations are required. The measurement results of the previous iteration are necessary for determining the probe rate of the next iteration. Probe Sender synchronizes necessary information using TCP, and then the UDP-based actual measurements will be performed.

B. Basic Mode: Monitoring Latency, Jitter, and Packet Loss

Considering that the available bandwidth measurements usually lead to high traffic costs, the measurement modes can be categorized into two types based on whether the bandwidth measurement is required. Basic Mode of `PerfTrace` is designed for monitoring latency, jitter, and packet loss with only a little traffic. We develop Basic Mode by taking a page from both `ping` and `OWping`: (i) similar to `ping`, the target will send a response at once as long as it receives the probe packets; (ii) similar to `OWping`, which establishes both TCP-based control channel and UDP-based measurement channel, Probe Sender measures the latency and jitter with timestamp embedded in the probe packets, and the packet loss rate is calculated according to the arrival packet number informed by the Probe Responder through the TCP control channel.

C. AB Mode: Monitoring Available Bandwidth

The measurements of available bandwidth (AB) usually require many packets to be sent. Therefore, in AB Mode of `PerfTrace`, Probe Responder no longer sends replies in response to each probe packet but only records the receiving time of each packet and sends the overall statistical data to the Probe Sender through the TCP-based control channel. Because AB Mode inherits the probing packet format of Basic Mode (changing the packet size by adding additional random load), it also supports one-way delay, one-way jitter, and one-way packet loss measurements along with bandwidth measurements. A novel measurement method that does not rely on fluid model assumption is proposed for the available bandwidth measurement in `PerfTrace`, and it has the following features:

- **Robust.** As for the measurement traffic, we send a packet sequence lasting for a fixed period ($\sim$ 20ms) instead of a fixed number of packets, which is more noise-immune.
- **Retrospective.** Our method estimates the bandwidth based on the historical data instead of a wide-range binary search like `Pathload`, resulting in much fewer iterations.
- **Agile.** When estimating the available bandwidth, we propose **DualProbe** algorithm (§III-C2), which removes the need for convergence of our probe rate.

1) Methodology: If the rate of probe traffic arriving at link L is $R_{in,L}$, and the available bandwidth of the link is AB_L, we call L as **critical link** when $R_{in,L} > AB_L$. A critical link will impact the probe traffic and thereby affect the measured result of available bandwidth. It's possible to have multiple critical links along one path. For simplicity, we first discuss the single-critical-link case and then extend it to the multiple-critical-link scenarios.

TABLE II: Variables and their meanings in §III-C1.

Variable	Meaning
C_{tight}	Capacity of tight link [12]
AB_{max}	Maximum available bandwidth in the historical data of the link
AB_{cur}	Current available bandwidth of the link
R_{cross}	Rate of cross traffic on the link
R_{snd}	Rate of the sent probe traffic
R_{rcv}	Rate of the received probe traffic

(1.1) Single Critical Link

Because the tight link (TABLE II) is necessarily a critical link, in the single-critical-link case the critical link is the tight link. Our method expands the idea of **PRM**, i.e., infer whether the rate of probe traffic is greater than the available bandwidth by observing whether the congestion occurs. In detail, there are two cases depending on whether the rate of probe traffic is greater than the available bandwidth:

Case 1: $R_{snd} \leq C_{tight} - R_{cross}$. No congestion occurs, so we have: $R_{rcv} \geq R_{snd}$.

Case 2: $R_{snd} > C_{tight} - R_{cross}$. Congestion occurs, leading to a decline in the receiving traffic rate. Thus we can observe: $R_{rcv} < R_{snd}$.

We focus on Case 2. When probe traffic and cross traffic have the same priority, they compete for bandwidth fairly. Therefore, the receiving rate of probe packets (sending rate is R_{snd1}) is:

$$R_{rcv1} = \frac{R_{snd1}}{R_{snd1} + R_{cross}} \cdot C_{tight} \tag{1}$$

As $AB_{cur} = C_{tight} - R_{cross}$, therefore we have:

$$AB_{cur} = C_{tight} - \frac{R_{snd1} \cdot (C_{tight} - R_{rcv1})}{R_{rcv1}} \tag{2}$$

There are two unknowns in the equation above. To find AB_{cur}, we need to get another similar equation by a repeated measurement with a different probe speed R_{snd2} ($> AB_{cur}$). Then we can find AB_{cur} with the two simultaneous equations:

$$AB_{cur} = \frac{R_{snd1} R_{rcv2}(R_{rcv1} + R_{snd2}) - R_{rcv1} R_{snd2}(R_{snd1} + R_{rcv2})}{R_{rcv2} R_{snd1} - R_{rcv1} R_{snd2}} \tag{3}$$

(1.2) Double Critical Links

Assume that there are two critical links $link_1$ and $link_2$ with capacities of C_1, C_2 and cross traffic of R_{cross1}, R_{cross2} respectively, then, the available bandwidth of them are:

$$\begin{cases} AB_1 = C_1 - R_{cross1} \\ AB_2 = C_2 - R_{cross2} \end{cases} \tag{4}$$

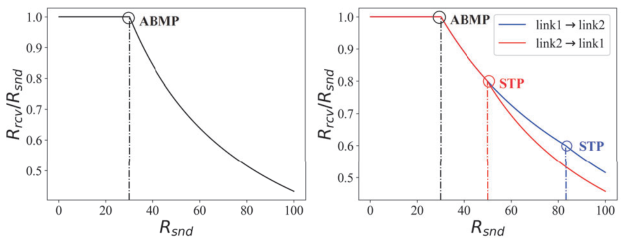
When **(a)** $\boldsymbol{AB_1 = AB_2 (= AB)}$, we assume that the probe traffic with a sending rate $R_{snd} > AB$ goes through $link_1$ and $link_2$ sequentially. The traffic drops to R_1 on $link_1$, and drops to R_2 on $link_2$, then the final receiving rate is R_2.

$$\begin{cases} R_1 = \frac{R_{snd}}{R_{snd} + C_1 - AB} \cdot C_1 \\ R_2 = \frac{R_1}{R_1 + C_2 - AB} \cdot C_2 \end{cases} \tag{5}$$

With Equation (5) and $R_{rcv} = R_2$, we have:

$$R_{rcv} = \frac{R_{snd}}{R_{snd} + \frac{C_1 \cdot C_2}{C_1 + C2 - AB}} \cdot \frac{C_1 \cdot C_2}{C_1 + C2 - AB} \tag{6}$$

Based on the equation above, we can know that a double-critical-link path with two link capacities of C_1 and C_2 and the same available bandwidth AB acts in the same way as a single-critical-link path with a link capacity of C' and available bandwidth of AB, where $C' = \frac{C_1 \cdot C_2}{C_1 + C2 - AB}$. Therefore, Equation (3) is still applying.



(a) $AB_1 = AB_2 = 30$ (b) $AB_1 = 30,\ AB_2 = 50$

Fig. 2: Double critical links with capacities of 80 ($link_1$) and 100 ($link_2$) respectively.

When **(b)** $\boldsymbol{AB_1 \neq AB_2}$, suppose $AB_1 < AB_2$. If the probe traffic passes $link_2$ first, the probe traffic will only be changed by $link_1$ as long as $AB_1 < R_{snd} \leq AB_2$. That is to say, it's equivalent to the single-critical-link case.

If the probe traffic passes $link_1$ first, when probe rate $R_{snd} > AB_1$, the probe rate drops to R_1 at $link_1$. When $R_1 \leq AB_2$, $link_2$ won't affect the probe rate. Therefore, it's equivalent to the case with only one critical link $link_1$ when R_{snd} satisfying:

$$R_{snd} \leq \frac{C_1 - AB_1}{C_1 - AB_2} \cdot AB_2 \tag{7}$$

We use an example to illustrate the conclusion we found above. Suppose we have $link_1$ and $link_2$ with capacity of 80 and 100. We assume $AB_1 = AB_2 = 30$ for the case (a) and $AB_1 = 30,\ AB_2 = 50$ for the case (b). We focus on the relationship between R_{rcv}/R_{snd} and R_{snd}. When $R_{snd} < 30$, R_{rcv}/R_{snd} shall be 1; when $R_{snd} > 30$, we have $R_{rcv} < R_{snd}$, i.e., $R_{rcv}/R_{snd} < 1$. We aim to find the point from which R_{rcv}/R_{snd} starts to decline from 1, i.e., the available bandwidth measurement point (**ABMP**). In the case of double critical links with different available bandwidth, besides the **ABMP**, another turning point exists when $R_{snd} = AB_2$ (passes $link_2$ first) or $R_{snd} = \frac{C_1 - AB_1}{C_1 - AB_2} \cdot AB_2$ (passes $link_1$ first), which we call the second turning point (**STP**).

We know from Figure 2 that double-critical-link paths can be converted into single-critical-link cases as long as we make the probe rate in the range from **ABMP** to **STP**.

(1.3) Multiple Critical Links

The conclusions introduced before on the double-critical-link cases can be extended to the multiple-critical-link cases. The cases with multiple critical links having the same available bandwidth are equivalent to single-critical-link cases. As for the cases with multiple critical links having different available bandwidths, we can find the **STP**, and the measurement is also equivalent to the single-critical-link case if we control our probe rate between the **ABMP** and the **STP**.

2) Measurement Procedure: As for the single-critical-link cases, we perform the measurement twice with different probe rates, which are higher than the available bandwidth. Then, we can get the available bandwidth using Equation (3). Regarding the multiple-critical-link cases, we shall control our probe rate to be lower than the **STP**. Additionally, a low probe rate will result in less intrusiveness. To get the probe rate slightly higher than the available bandwidth, we design a five-phase measurement program as shown in Figure 3. We describe each of the five phases below.

Initialization. This phase aims to get the initial probe rate for available bandwidth measurements, as shown in the Algorithm 1. The users can specify the initial probe rate if they have prior knowledge about bandwidth. Otherwise, `PerfTrace` queries the local database for recent measurement records and then determines the initial probe rate based on the historical data. If neither of the above, we measure the asymptotic dispersion rate (**ADR** [21]) firstly.

Algorithm 1 Startup Probe Rate Selection

```
Input: The initial probe rate specified by users: $R_0$;
    Historical available bandwidth measurements stored in database: $S = \{AB_{t-1}, AB_{t-2}, AB_{t-3}, \cdots\}$;
    Window size of historical values used for reference: $N$ (default 10);
    Enlarge factor: $factor$.   ▷ $factor$ is a number slightly greater than 1 (default 1.2).
Output: The startup probe rate: $R_{snd0}$;
1: if $R_0$ is not Null then
2:     $R_{snd0} \leftarrow R_0$   ▷ priority use of specified rate
3: else if $S \neq \emptyset$ then
4:     $AB_{max} \leftarrow \max_{1<i<N} AB_{t-i}$
5:     $R_{snd0} \leftarrow \max(AB_{max}, AB_{t-1} \times factor)$   ▷ we want the initial probe rate to be a value slightly greater than the available bandwidth.
6: else
7:     $R_{snd0} \leftarrow \infty$   ▷ $\infty$ means we do the ADR test.
8: return $R_{snd0}$
```

Fast Decreasing. This phase prevents the probe rate from being too high, leading to congestion and packet loss on the paths. Fast Decreasing reduces the R_{snd} according to the previous measurement R_{rcv} to lower the probe rate than the available bandwidth.

Slow Growing. We can finally find a probing rate slightly higher than the available bandwidth by continuously multiplying the probe rate with a factor marginally greater than 1. This process is similar to the TCP slow start. Unlike the Fast Decreasing phase, which is based on the R_{rcv}, Slow Growing is based on the R_{snd} in the measurement.

Dual Probes. We send our probes twice with different probe rates. For the first probe, we use the probe rate we got by the first three phases mentioned above, which is a probing rate slightly higher than the available bandwidth. For the second probe, we use the receiving rate of the first probe as the probe rate. These two probes result in two equations with two unknowns. We can find the available bandwidth by solving the equations (Equation (3)).

Result Storage. We store the results in our local database. In the subsequent measurements, we can use the previous results to determine the initial probe rate.

The five-phase measurements seem complicated, and the second and third phases require self-loops. Luckily, as long as we choose the appropriate initial probe rate, we can skip the loops (through the green path in Figure 3). Our practical experiments show that most bandwidth measurements only require three- or four-time iterations.

IV. Evaluation

Our evaluation aims to show that in comparison to other existing tools, `PerfTrace`: (i) measures more metrics with lower time and traffic cost and (ii) measures available bandwidth more accurately.

A. Basic Mode

As for the metrics, including latency, jitter, and loss rate, we use the Basic Mode of `PerfTrace` to perform our evaluation. Though the AB Mode of `PerfTrace` can also measure them, it will result in a higher overhead for the available bandwidth measurements. We only use the AB Mode when evaluating the ability to measure available bandwidth.

We deploy vantage points on two Internet nodes in North America and Asia, namely $Node_1$ and $Node_2$. $Node_1$ is a cloud server with an egress bandwidth of 1Gbps, while $Node_2$ is a physical host accessed through a wired network with an egress bandwidth of 100Mbps. The RTT between them is about 250ms. We use `tcpdump` [22] to collect the probe data and count the total traffic cost.

Three well-known tools are used as competitors: `ping`, `OWping` and `Sting`. For fairness, we use unified measurement parameters: 100 probe packets with intervals of 10ms. It takes 1s to send out all the probe packets. Considering the time cost of establishing TCP connections (except `ping`), waiting for retransmission, and calculating results, the whole process of the measurements takes $1\text{s} + \text{RTT} \approx 1.25\text{s}$ at least. As shown in Table III, `PerfTrace` get more metrics in one measurement when compared with other tools. Without `PerfTrace`, the measurement of all the metrics in Table III requires one `ping` and two `OWping` (forward and backward) performed, resulting in $\sim 9.4\times$ time cost and $\sim 3.6\times$ traffic cost than `PerfTrace`.

B. AB Mode

We evaluate the ability of `PerfTrace`, `Pathload`, `Pathchirp`, `Assolo`, `Abing`, and `Spruce` to measure available bandwidth on our testbed.

Testbed Setup. We build a testbed with three commercial switches and several hosts in a dumbbell topology to evaluate the accuracy of several tools, as illustrated in Figure 4. The ports of the switches are of type Copper, speed 100 Mbps, and mode Full Duplex. Several hosts connected to switch ports and accessed using VLAN constitute a local area network. These hosts can send data to each other.

We measure the available bandwidth of $Path_{AB}$ between $Node_A$ and $Node_B$. At the same time, we inject random cross traffic with `iPerf3` on $Link_1$ and $Link_2$, the mandatory links from $Node_A$ to $Node_B$. The true available bandwidth $AB =$

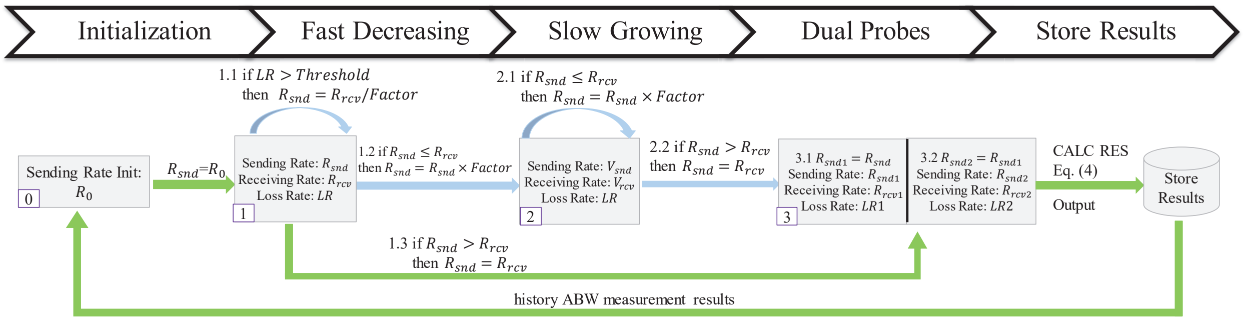

Initialization
Fast Decreasing
Slow Growing
Dual Probes
Store Results
1.1 if $LR > Threshold$ then $R_{snd} = R_{rcv}/Factor$
2.1 if $R_{snd} \le R_{rcv}$ then $R_{snd} = R_{snd} \times Factor$
Sending Rate Init: R_0
0
R_{snd}=R_0
Sending Rate: R_{snd} Receiving Rate: R_{rcv} Loss Rate: LR
1
1.2 if $R_{snd} \le R_{rcv}$ then $R_{snd} = R_{snd} \times Factor$
Sending Rate: V_{snd} Receiving Rate: V_{rcv} Loss Rate: LR
2
2.2 if $R_{snd} > R_{rcv}$ then $R_{snd} = R_{rcv}$
3.1 $R_{snd1} = R_{snd}$ Sending Rate: R_{snd1} Receiving Rate: R_{rcv1} Loss Rate: $LR1$
3
3.2 $R_{snd2} = R_{snd1}$ Sending Rate: R_{snd2} Receiving Rate: R_{rcv2} Loss Rate: $LR2$
CALC RES Eq. (4)
Output
Store Results
1.3 if $R_{snd} > R_{rcv}$ then $R_{snd} = R_{rcv}$
history ABW measurement results

Fig. 3: The procedure of available bandwidth measurement.

TABLE III: Metrics measured by various tools and the duration and overhead of the measurement. All metrics can be divided into one-way (forward, F and backward, B) and two-way (round-trip, RT). Sting's measurements are often unsuccessful, so there is no exact duration and overhead.

Tools	**Latency**			**Jitter**			**Loss Rate**			**Duration/s**	**Overhead/KB**	
	F	B	RT	F	B	RT	F	B	RT		F	B
ping	✗	✗	✓	✗	✗	✓	✗	✗	✓	1.43	7.2	7.2
OWping	✓	✗	✗	✓	✗	✗	✓	✗	✗	7.78	3.2	3.4
Sting	✗	✗	✗	✗	✗	✗	✓	✓	✓	*	>2.1	>2.0
PerfTrace	✓	✓	✓	✓	✓	✓	✓	✓	✓	**1.81**	**3.3**	**4.2**
ping+OWping×2	✓	✓	✓	✓	✓	✓	✓	✓	✓	16.99	13.6	14.0

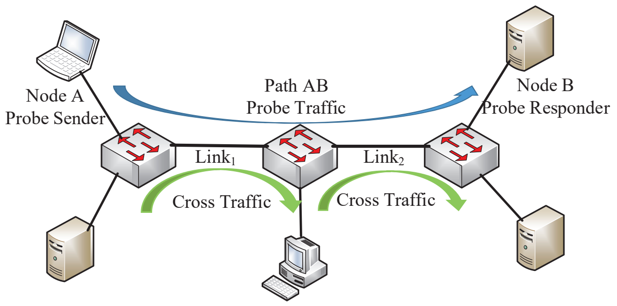


Fig. 4: Testbed with dumbbell topology.

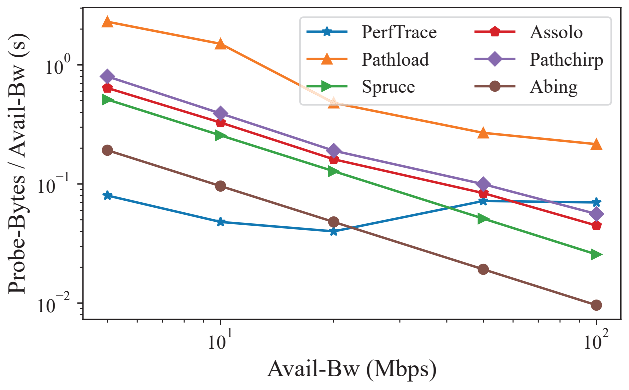


Fig. 5: Data overhead of several tools at different available bandwidth conditions.

$\min_{\forall i \in \{1,2\}}(C_i - Rcross_i)$. We can monitor the cumulative amount of data forwarded by the switch ports in real-time, so it is not difficult to obtain the cross traffic rate and the true available bandwidth of all links (C_{link}=100Mbps) through the SNMP protocol. Using the above available bandwidth obtained via SNMP as Ground Truth, we can evaluate the measurement accuracy of each tool.

Accuracy. We use several tools to monitor the available bandwidth in $Path_{AB}$ separately, and the results are shown in Figure 6. Of the several tools, `PerfTrace` and `Spruce` perform the best, very close to the Ground True. `Spruce` indirectly obtains the available bandwidth by measuring the cross traffic rate in the path, so the tight link capacity C_{tight} (100Mbps in our testbed) needs to be provided when measuring with `Spruce`. While `PerfTrace` requires no additional information when measuring. Other than the above two tools, the other tools overestimate the available bandwidth.

The mean absolute error and mean relative error of the tools mentioned above are listed in Table IV. `PerfTrace` has the highest accuracy with a mean absolute error of 3.01Mbps and a mean relative error of 5.22% over the entire measurement period. `Spruce` is in second place, with a mean relative error of 8.17% over the measurement. The mean relative errors of the remaining tools range from 21.84% to 37.24%.

TABLE IV: Errors of several available bandwidth measurement tools in the testbed

Tools	**Mean Absolute Error**	**Mean Relative Error**
PerfTrace	3.01Mbps	5.22%
Spruce	4.20Mbps	8.17%
Assolo	9.56Mbps	21.84%
Pathload	13.66Mbps	31.44%
Pathchirp	18.18Mbps	37.24%

Overhead. We calculate the average ratio of the probing data overhead to the available bandwidth (relative overhead) of the

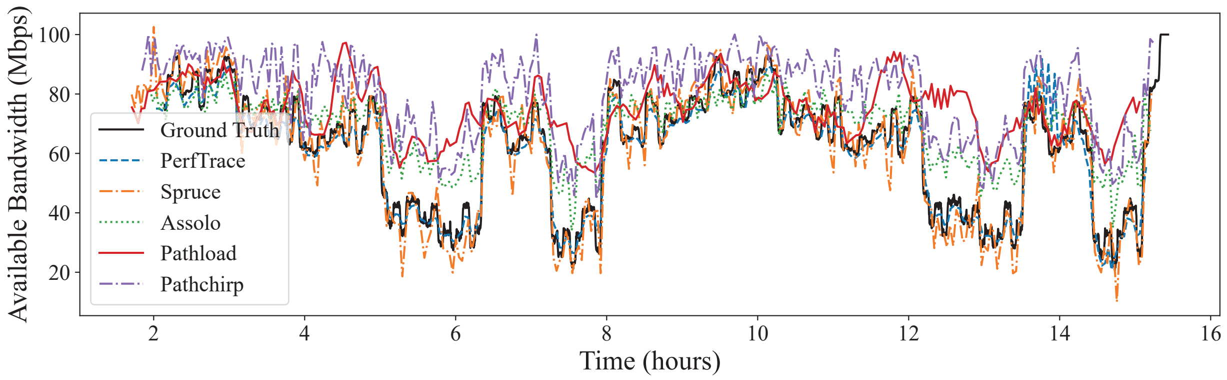


Fig. 6: Results of several available bandwidth measurement tools in the testbed.

monitoring path for each tool. A larger ratio indicates greater intrusiveness. Figure 5 implements the relationship between the average relative overhead of each tool and the available bandwidth. We find that for low bandwidth paths (less than 20Mbps), the overhead of `PerfTrace` is less than other tools. As the available bandwidth increases, the relative overhead of the other tools decreases, and the relative overhead of `PerfTrace` remains stable (within 0.1 seconds). It is worth noting that regardless of the monitoring path's bandwidth, the resources consumed by `PerfTrace`'s probing overhead are consistently only about 1/600 of the total resources for a measurement frequency once per minute.

V. Conclusion

This paper presents `PerfTrace`, a new end-to-end network performance monitoring tool to make network monitoring multi-metric, fast, lightweight, and accurate. On one hand, `PerfTrace` provides a high integration of different functions of existing measurement tools, supporting the measurement of essential metrics such as latency, jitter, packet loss, and available bandwidth. On the other hand, an innovative available bandwidth measurement algorithm is proposed to make `PerfTrace` more efficient and accurate. `PerfTrace` is built with a generic framework and flexible expandability, making it easy to update. In the future, we will continue to apply and optimize `PerfTrace` in real networks. Furthermore, we will continue contributing new algorithms and functions to `PerfTrace` to make it more versatile and powerful.

References

[1] F. Y. Yan, J. Ma, G. D. Hill, D. Raghavan, R. S. Wahby, P. Levis, and K. Winstein, "Pantheon: the training ground for internet congestion-control research," in *2018 Annual Technical Conference ({ATC} 18)*, 2018, pp. 731–743.

[2] C. Feng, L. Wang, K. Wu, and J. Wang, "Bound-based network tomography with additive metrics," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 316–324.

[3] Y. Eianv, "Amazon found every 100ms of latency cost them 1% in sales," https://www.gigaspaces.com/blog/amazon-found-every-100ms-of-latency-cost-them-1-in-sales, accessed June 9, 2022.

[4] A. Li, X. Yang, S. Kandula, and M. Zhang, "Cloudcmp: comparing public cloud providers," in *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, 2010, pp. 1–14.

[5] L. Kleinrock and W. E. Naylor, "On measured behavior of the arpa network," in *Proceedings of the May 6-10, 1974, national computer conference and exposition*, 1974, pp. 767–780.

[6] M. Hasib and J. A. Schormans, "Limitations of passive & active measurement methods in packet networks," in *London Communications Symposium (LCS), London, UK*, vol. 38, 2003.

[7] K. Ervasti, "A survey on network measurement: Concepts, techniques, and tools," 2016.

[8] C. Tan, Z. Jin, C. Guo, T. Zhang, H. Wu, K. Deng, D. Bi, and D. Xiang, "{NetBouncer}: Active device and link failure localization in data center networks," in *16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19)*, 2019, pp. 599–614.

[9] T. Pan, E. Song, Z. Bian, X. Lin, X. Peng, J. Zhang, T. Huang, B. Liu, and Y. Liu, "Int-path: Towards optimal path planning for in-band network-wide telemetry," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 487–495.

[10] F. Gont, G. Gont, and C. Pignataro, "Recommendations for filtering icmp messages," *draft-ietf-opsecicmp-filtering-03*, 2012.

[11] X. Yang, X. Wang, Z. Li, Y. Liu, F. Qian, L. Gong, R. Miao, and T. Xu, "Fast and light bandwidth testing for internet users." in *NSDI*, 2021, pp. 1011–1026.

[12] J. Strauss, D. Katabi, and F. Kaashoek, "A measurement study of available bandwidth estimation tools," in *Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, 2003, pp. 39–44.

[13] P. Ha and L. Xu, "Available bandwidth estimation in public clouds," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2018, pp. 238–243.

[14] J. Navratil and R. L. Cottrell, "Abwe: A practical approach to available bandwidth estimation," in *Proceedings of the 4th Passive and Active Measurement Workshop PAM 2003*. Citeseer, 2003.

[15] V. J. Ribeiro, R. H. Riedi, R. G. Baraniuk, J. Navratil, and L. Cottrell, "pathchirp: Efficient available bandwidth estimation for network paths," in *Passive and active measurement workshop*, 2003.

[16] E. Goldoni, G. Rossi, and A. Torelli, "Assolo, a new method for available bandwidth estimation," in *2009 Fourth International Conference on Internet Monitoring and Protection*. IEEE, 2009, pp. 130–136.

[17] M. Jain and C. Dovrolis, "Pathload: A measurement tool for end-to-end available bandwidth," in *In Proceedings of Passive and Active Measurements (PAM) Workshop*. Citeseer, 2002.

[18] R. Prasad, M. Jain, and C. Dovrolis, "Effects of interrupt coalescence on network measurements," in *International Workshop on Passive and Active Network Measurement*. Springer, 2004, pp. 247–256.

[19] P. Ha, "Bandwidth estimation in cloud networks," Ph.D. dissertation, The University of Nebraska-Lincoln, 2021.

[20] T. P. Team, "owping - client application to request one-way latency tests," https://github.com/perfsonar/owamp, accessed June 9, 2022.

[21] C. Dovrolis, P. Ramanathan, and D. Moore, "What do packet dispersion techniques measure?" in *Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No. 01CH37213)*, vol. 2. IEEE, 2001, pp. 905–914.

[22] D. A. Joseph, V. Paxson, and S. Kim, "tcpdump tutorial," *University of California, EE122 Fall*, 2006.