Telemetron: Measuring Network Capacity Between Off-Path Remote Hosts

Devashish Gosain[1], Aishwarya Jaiswal[2], H. B. Acharya[3], and Sambuddho Chakravarty[2]

[1]Max Planck Institute for Informatics, Germany
Email: dgosain@mpi-inf.mpg.de
[2]Indraprastha Institute of Information Technology Delhi (IIITD), New Delhi, India
Email: {aishwarya14007, sambuddho}@iiitd.ac.in
[3] Rochester Institute of Technology, NY, USA
Email: acharya@mail.rit.edu

2021 IEEE 46th Conference on Local Computer Networks (LCN) | 978-1-6654-1886-7/21/$31.00 ©2021 IEEE | DOI: 10.1109/LCN52139.2021.9524946

***Abstract*—This paper presents *Telemetron*, the first active bandwidth measurement tool that can estimate the path capacity between two remote hosts, from an off-path Measuring Machine (MM). It is possible to induce traffic flow between off-path remote hosts—sending request packets to one host, with a spoofed source IP, will cause the first host to send reply packets to the other. The challenge for MM is to measure the rate at which these packets arrive at the second machine. Our key observation is that if the second machine has a global IP-ID counter, the arrival of packets can be monitored remotely, using probes from MM. By observing the rate of increment in the global IP-ID counter, MM estimates the path capacity between remote hosts. Telemetron shows high accuracy; on average, the path capacity reported is 92.5% of the theoretical limit.**

I. Introduction

Network bandwidth is a key parameter of network performance, affecting the entire range of Internet applications and protocols. Traffic engineering, load balancing, and capacity planning all depend on measuring and estimating network bandwidth. As a result, there is a considerable body of research literature [1], [2] devoted to various kinds of bandwidth, such as capacity (maximum total rate of data transmission over a path), available bandwidth (unused capacity over a path, available to new flows), etc.

In this paper, we focus on *capacity*, *i.e.,* the maximum rate at which data can be transmitted over a network path. This is the metric commonly used by ISPs to plan their operations — *e.g.*, Malloy *et al.* [3] and Fichou *et al.* [4] develop systems for the optimal allocation of capacity to multiple users. Network operators also use capacity estimation to identify bottleneck links [5], select best paths, perform traffic engineering, *etc.*

Prior researchers propose several techniques to estimate capacity between two Internet hosts: variable packet size, packet pair and packet train dispersion methods [2]. Such techniques vary the number of packets, their size, *etc.*, and measure the bottleneck capacity. However, all such methods suffer from a common weakness: they require the user to have direct access to the source, the destination, or both. *In other words, no existing tool can find the path capacity between two remote hosts.* Yet, there is a real need for off-path capacity estimation. For instance, in the recently proposed Time Series Latency Probes (TSLP) technique to find under-provisioned inter-AS links [6], the authors cannot find link capacities, and are forced to assume that all links between two networks have the same capacity. Knowledge of the actual link capacities would greatly increase the accuracy of such predictions.

This paper introduces and demonstrates *Telemetron* (Fig. 1), a novel tool that measures path capacity between two *remote* machines (end hosts or routers) using only measurements from an off-path measuring machine (MM). *We require no control over either the source or the destination of the path measured.* Telemetron measures the capacity of the path, through a novel use of the IP-ID counter on the destination machine.

To measure the amount of traffic transmitted between the two remote machines (named source and destination), Telemetron sends spoofed packets to the source. These spoofed packets elicit responses from source, directed to destination. The responses, in turn, trigger replies from destination. The IP-ID counter at destination is incremented each time it sends a reply packet. If the destination machine uses a *global* IP-ID counter [7], the MM is able to remotely measure this increment in IP-ID. From the rate of change in IP-ID, MM learns the rate of arrival of packets at destination. As it knows the size of the packets, MM is able to estimate the path capacity.

Telemetron can be used to measure the capacity of *any* network path ending in a machine whose TCP/IP stack uses a global IP-ID counter[1]—*whether this machine is an end host or a router*. This makes it more useful than existing techniques, which require the experimenter to have control over one or both end-points of the path being measured (and which are thus mostly used for end hosts). Thus, in future, Telemetron could be used to augment large-scale Internet maps [9], [10] with link data for under-served networks.

[1]A recent conservative scan of the entire Internet by Salutari *et al.* [7], revealed $\approx 480,000$ machines with global IP-ID counters and Li *et al.* [8] identified $222,782$ global IP-ID hosts within the US, indicating that such points are not hard to find.

II. Background and Relevant Work

Several existing tools measure the path capacity between two hosts: *Pathchar* [11], *Nettimer* [1], *PathRate* [12] *etc.* In all such approaches, the path source transmits a series of packets (*packet train*) to the path destination, varying their quantity and lengths. The end-to-end delay of the received packet train is known as dispersion. The amount of data transmitted through the packet train, divided by dispersion, is used to estimate the path capacity. Prasad *et al.* [2] provide a comprehensive review of all such techniques. However, all existing techniques require at least one of the end points to be in control of the experimenter.

Our novel tool, Telemetron, requires control of neither the source nor the destination (of the path we measure). It relies on an off-path MM sending the spoofed packet train to manipulate the global IP-ID counter of the destination.

The IP-ID is a 16-bit field in the IPv4 header, originally used to assist in the reassembly of fragmented packets at the receiver. Several TCP/IP implementations make use of a global IP-ID counter, i.e., the IP-ID field is incremented by one on every outgoing packet. Telemetron's use of global IP-ID counters for measuring path capacity, was inspired by previous authors who use IP-ID for network measurements.

- The first such use, Antirez' *Idle Scan* [13], was a port scanning technique for the tool `nmap`. It used the IP-ID field to covertly identify open TCP ports on a remote host.
- Bellovin [14] used IP-ID to find the number of hosts behind a NATed IP. Keys *et al.* [15] developed Midar, which uses IP-ID to resolve router aliases in network maps.
- Ensafi *et al.* [16] made use of IP-ID variations to remotely detect censorship in a target country. In 2021, Li *et al.* [8] used these IP-ID variations to confirm the IP blocklist used by the remote hosts.

Despite its dependence on global IP-ID, we assert that Telemetron is far more generally useful than existing methods, which need the user to have privileges on path source and/or destination machines. Telemetron can remotely compute path capacity *from any machine to any global IP-ID machine*, and a considerable fraction of machines on the Internet have such global counters. A recent large-scale scan [7], checking one random IP from each /24 prefix, identified over $480,000$ machines with global IP-ID.

III. Approach

A. Pre-requisites for Telemetron

The specific requirements of different entities in the Telemetron system (as shown in Fig. 1) are as follows.

Measurement Machine ***(MM):*** (1) IP-Spoofing: MM needs IP spoofing, so as to impersonate as R_e to R_f.

Reflector ***(R_f):*** (1) Port constraint: At least one TCP port must be open (P_f) so that R_f replies with SYN/ACK packet for a new SYN packet.

Receptor ***(R_e):*** (1) Port constraint: At least one TCP port must be closed (P_e) so that R_e generates RST packets in response to SYN/ACK packets[2]. (2) It must have a global IP-ID counter.

We also assume that the firewalls of the remote machines *i.e.,* path source (reflector R_f) and path destination (receptor R_e) are not set to filter our probe packets.

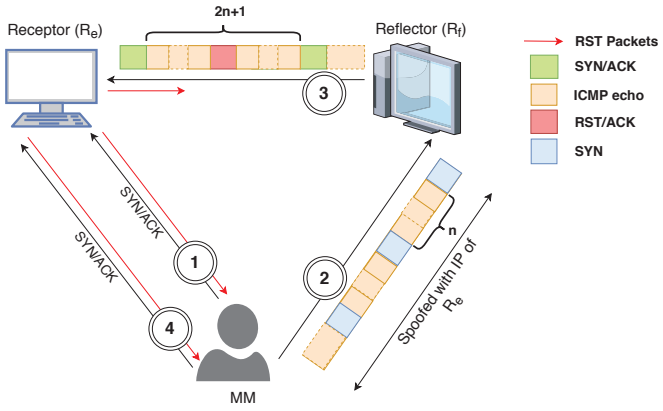

Figure 1. Telemetron: Estimating capacity between reflector and receptor from MM.

B. Working of Telemetron:

Step-by-step, Telemetron works as follows. (Fig. 1):

1) MM sends a SYN/ACK packet to a closed port (P_e) on R_e. R_e replies with a RST packet to MM. The reply contains, the IP-ID value of R_e.
2) MM sends a packet train (Fig. 2) to R_f. All these packets have spoofed source IP address, and appear to be from R_e. The packet train consists of *chunks*, each comprising of one standard 40 byte SYN packet ($src.port = P_e, dest.port = P_f$) and n ICMP echo request packets, each of 1400 bytes. The SYN packets are all identical, except that we increment the sequence number of each successive SYN packet by one[3].

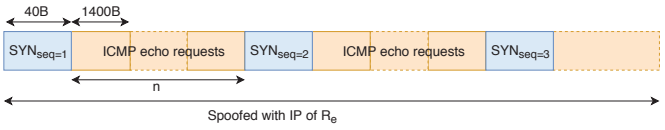

Figure 2. Packet train which MM sends to R_f.

3) On reception of packet train
   a) R_f, responds with a new packet train (Fig. 3). Note that these packets are addressed to R_e, as R_f believes it is replying to packets from R_e.
   For the first SYN packet seen by R_f, it generates a SYN/ACK. For the n ICMP echo requests, it generates n ICMP echo replies. Next, R_f receives a fresh SYN packet that again appears to be arriving from R_e.

[2]Almost all machines generally have port 113 (Ident port) closed.

[3]If SYN packets share a sequence number, R_f would drop them as duplicates.

However, as per the TCP standard (RFC 793[4]) this fresh SYN elicits RST/ACK. And once again, for the succeeding n ICMP echo requests, it generates n ICMP echo replies.

Thus, the R_f generates SYN/ACKs for odd numbered SYNs, RST/ACKs for even numbered SYNs, and ICMP replies for all Echo requests.

b) R_e receives the packet train from R_f.

From R_e's perspective, these packets are all unsolicited and were received on a closed port. So for every SYN/ACK packet from R_f, R_e generates a RST packet. It also ignores the rest of the packets.

The generation of RST packets causes R_e to increment its IP-ID. The IP-ID increment is equal to the number of SYN/ACK packets it receives. Between one SYN/ACK and the next, there will be $2n$ ICMP echo reply messages (of 1400 bytes each) and one RST/ACK packet (Fig. 3); all these packets are dropped by the kernel of R_e.

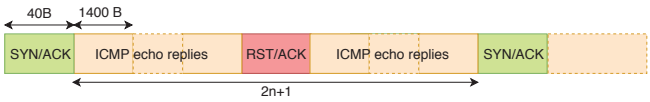

Figure 3. Packet train which R_f sends to R_e.

4) After sending the packet train (Fig. 2) to R_f, MM probes R_e with SYN/ACK packets. We denote the time difference between the successive probes to be δ time units.

R_e replies with RST packets that contain its global IP-ID value at that instant. The difference in IP-ID values, obtained from the responses of two successive probes, is proportionate to the volume of traffic received at R_e in the time between these probes.

If in δ time, MM observes the IP-ID to be incremented by I, it *estimates* the capacity between R_f and R_e as:

$$C(R_f, R_e) = \frac{\texttt{(ICMP bytes) + (TCP bytes)}}{\texttt{time}} = \frac{(I * 2n * 1400) + (40 * (2I - 1))}{\delta}$$

We explain this with an example.

In our experiments, for a particular (R_f, R_e) pair, we set C(MM, R_f) to 100 Mbps and C(R_f,R_e) to 10 Mbps.

MM sent spoofed packet train (Fig. 2) with high data rate (1 Gbps) to R_f. The packet train reached R_f at the rate of $\approx$ 100 Mbps (as C(MM, R_f) is 100 Mbps, and limits the flow rate). In response, R_f generated a new packet train to R_e (ref. Fig. 3). This (response) packet train reached R_e at a rate of $\approx$ 10 Mbps (as C(R_f,R_e) is 10 Mbps).

At the same time, MM also sent probe packets (SYN/ACK) to R_e, at δ time intervals. MM observed the IP-ID was incremented by I over time δ, and estimated the bandwidth as per the above equation.

Keeping $n = 10$, $\delta = 250$ ms, we actually observed $I = 11$. Thus Telemetron reported $C(R_f, R_e) = 9.86$ Mbps.

Packet Train in Telemetron: We observed that trains with small sized packets (40 byte TCP SYN with no payload) were insufficient to fully utilize the link capacity. Prasad *et al.* [2] also presented a similar observation—large packet sizes ($>$ 500 bytes) are required to achieve the maximum data rate. Even when we crafted MTU size SYN packets from MM and sent them to R_f, the response from R_f still consisted of 40 byte SYN/ACK packets, and could not fully utilize the path capacity between R_f and R_e. *A packet train with only SYN packets was able to cause IP-ID increment and reflection, but failed to fully utilize the link capacity.*

The ICMP echo requests, each of 1400 bytes, resulted in responses (1400 byte ICMP echo replies generated from R_f) that fully utilized the link bandwidth. However, *ICMP replies by themselves would fail to increment the IP-ID counter (as they would simply be dropped by R_e).*

Thus, we used a combination of SYN and ICMP packets to build the probe train (ref. Sec. III). ICMP Echo replies with 1400 byte payload were used to exhaust bandwidth, and SYN/ACKs (from R_f) trigger RSTs from R_e, thus incrementing its global IP-ID counter.

Implementation of Telemetron: The code for Telemetron's packet train was written in C and used `Raw Sockets`, to achieve maximum possible data rate of 1 Gbps (our MM had 1 Gb interface cards). Our packet train consists of 10-20 ICMP packets placed between two SYN packets. The number of ICMP packets sent, also has an impact on capacity estimation. If there are too many ICMP packets in the train, MM observes very few IP-ID perturbations in duration δ. (A train composed mostly of ICMP packet has few TCP SYN/ACK packets, and it is the SYN/ACK packets that elicit RSTs and increment the IP-ID counter.) On the other hand, with too few ICMP packets to pad the train, one may observe the adverse impact of packet reorderings. *E.g.*, the receptor might receive a burst of SYN/ACK packets, which arrive before the separating ICMP packets; this causes incorrect and fluctuating measurements.

IV. Experimental Setup and Results

We now explain the experiments that we conducted to establish the efficacy of Telemetron. Fig. 4 represents our setup for testing Telemetron in a controlled lab environment.

We used Telemetron to measure the capacity of a network path including a known bottleneck (between Nodes 1 and 2), so the ground truth was the (known) capacity of that link. We observed the effects of varying (1) Capacity of the bottleneck link. (2) Amount of cross-traffic (from Nodes 3 and 5, to Node 1). Table I presents our results.

The first set of results (Rows 1 – 3) show the performance of Telemetron in the absence of cross-traffic. In all tests (varying the bottleneck at node 2 from 10 to 1000 Mbps), Telemetron returned an estimate close to the actual capacity (column 3).

[4]If a SYN arrives when a TCP connection is already established, or is underway, and its seq. no. is within the receive window, the receiver sends a RST and terminates the connection.

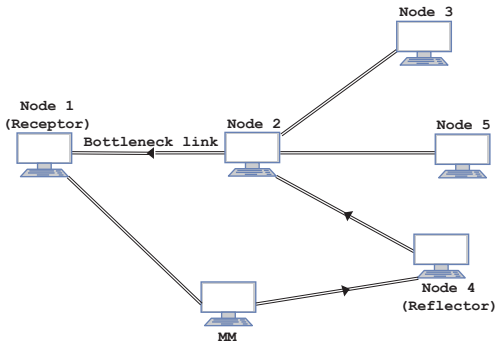

Figure 4. Laboratory setup: Six desktop machines equipped with 1 Gbps interface cards, on an Ethernet network. Node 1, with global IP-ID counter, was designated R_e. Node 4 was R_f. Node 2 acted as intermediate hop between R_f and R_e. Nodes 3 and 5 were used to generate background traffic.

The second set of results show the effect of *inflexible* background traffic (i.e., existing UDP/ICMP traffic from node 3 and node 5 to node 1). Column 2 shows the cross-traffic injected in the network. In these tests, MM sent the probe train to node 4 at 100 Mbps.

We conclude that in the absence of cross-traffic, Telemetron reports capacity accurately (≈ 100 Mbps). In the presence of non-adaptive cross-traffic (non-TCP flows), which competes with the Telemetron packet train, *the capacity is proportionally divided among the flows*. Telemetron, of course, can only report the capacity experienced by its flow, the packet train.

Actual Capacity (Mbps)	Inflexible Cross-Traffic (Mbps)	Capacity measured (Mbps)
1000	0	946
100	0	96
10	0	9.4
100	100	49
	75	55.6
	50	67.2
	30	76.16
	10	96.32

Table I
CAPACITY ESTIMATION IN THE LAB ENVIRONMENT.

For instance, when total cross-traffic (from node 3 and node 5 to node 1) totaled 100 Mbps, and the probe packet train was also transmitted at 100 Mbps, Telemetron recorded a capacity of 49 Mbps (ref. Table I).

We then experimentally observed how *TCP* cross-traffic impacts the behavior of Telemetron. We set node 3 to be the server; node 1, as client, downloaded a large (1 GByte) file from the server using the utility `wget`. The initial download rate (TCP flow) was 88 Mbps, which is slightly lower than the capacity estimated by Telemetron (ICMP flow). Then, Telemetron was executed from MM and capacity between node 4 and 1, was estimated to be 95.3 Mbps, (which is close to 96 Mbps measured without TCP cross-traffic). TCP throughput, on the other hand, was sharply reduced (to roughly 327 KBps) due to the shared link between node 2 and node 1. We thus conclude that capacity measurement with Telemetron *is not impacted by TCP traffic*, most likely because TCP is very sensitive to packet loss (and adjusts to congestion by "backing off"). Over a range of experiments, varying flows of both TCP and non-TCP cross-traffic, and varying the capacity of the bottleneck (10, 100, and 1000 Mbps), we consistently observed the expected capacity values.

V. CONCLUSION

The existing tools for bandwidth measurement require the observer to control at least one end of the path, whose bandwidth is to be measured. This paper introduces a new tool, *Telemetron*, which can measure network path capacity between *any* two hosts from a single off-path vantage point.

Telemetron is reliable and accurate in practice. In in-lab tests, it reports capacity of about 93% of the theoretical limits, which compares favorably to existing tools. We intend to test this tool on Internet and expect that this tool may become useful for large-scale measurement of capacity (a direction we intend to explore in future).

REFERENCES

[1] K. Lai and M. Baker, "Nettimer: A Tool for Measuring Bottleneck Link Bandwidth." in *USITS*, vol. 1, 2001, pp. 11–11.
[2] R. Prasad, C. Dovrolis, M. Murray, and K. Claffy, "Bandwidth estimation: metrics, measurement techniques, and tools," *IEEE Network*, 2003.
[3] P. J. Malloy, D. Znamova, A. J. Cohen, A. Dunn, J. W. Strohm, A. H. Ali, and R. M. Elsner, "Network capacity planning," 2012, US Patent 8,296,424.
[4] A. Fichou, C. Galand, and J.-F. Le Pennec, "Network capacity planning based on buffers occupancy monitoring," 2004, US Patent 6,690,646.
[5] S. Sundaresan, X. Deng, Y. Feng, D. Lee, and A. Dhamdhere, "Challenges in inferring internet congestion using throughput measurements," in *Proceedings of the 2017 Internet Measurement Conference*. ACM, 2017, pp. 43–56.
[6] A. Dhamdhere, D. D. Clark, A. Gamero-Garrido, M. Luckie, R. K. Mok, G. Akiwate, K. Gogia, V. Bajpai, A. C. Snoeren, and K. Claffy, "Inferring persistent interdomain congestion," in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*. ACM, 2018, pp. 1–15.
[7] F. Salutari, D. Cicalese, and D. J. Rossi, "A closer look at IP-ID behavior in the wild," in *International Conference on Passive and Active Network Measurement*. Springer, 2018, pp. 243–254.
[8] V. G. Li, G. Akiwate, K. Levchenko, G. M. Voelker, and S. Savage, "Clairvoyance: Inferring blocklist use on the internet," in *Passive and Active Measurement: 22nd International Conference, PAM 2021, Virtual Event, March 29–April 1, 2021, Proceedings*. Springer Nature, p. 57.
[9] R. Durairajan, S. Ghosh, X. Tang, P. Barford, and B. Eriksson, "Internet atlas: a geographic database of the internet," in *Proceedings of the 5th ACM workshop on HotPlanet*. ACM, 2013, pp. 15–20.
[10] R. Durairajan, P. Barford, J. Sommers, and W. Willinger, "Intertubes: A study of the us long-haul fiber-optic infrastructure," in *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 4. ACM, 2015, pp. 565–578.
[11] V. Jacobson, "Pathchar: A tool to infer characteristics of Internet paths," 1997.
[12] C. Dovrolis, P. Ramanathan, and D. Moore, "Packet-dispersion techniques and a capacity-estimation methodology," *IEEE/ACM Transactions On Networking*, vol. 12, no. 6, pp. 963–977, 2004.
[13] Antirez, "Nmap idle scan," https://nmap.org/book/idlescan.html.
[14] S. M. Bellovin, "A technique for counting NATted hosts," in *Proceedings of the 2nd SIGCOMM Workshop on Internet Measurement*. ACM, 2002.
[15] K. Keys, Y. Hyun, M. Luckie, and K. Claffy, "Internet-scale IPv4 alias resolution with MIDAR," *IEEE Transactions on Networking*, 2013.
[16] R. Ensafi, P. Winter, A. Mueen, and J. R. Crandall, "Analyzing the Great Firewall of China over space and time," *Proceedings on privacy enhancing technologies*, vol. 2015, no. 1, pp. 61–76, 2015.