From Paris to Tokyo: On the Suitability of ping to Measure Latency

Cristel Pelsser
Internet Initiative Japan
Tokyo, Japan
cristel@iij.ad.jp

Luca Cittadini
Roma Tre University
Rome, Italy
ratm@dia.uniroma3.it

Stefano Vissicchio
Universite catholique de Louvain
Louvain-la-Neuve, Belgium
stefano.vissicchio@uclouvain.be

Randy Bush
Internet Initiative Japan
Tokyo, Japan
randy@psg.com

ABSTRACT

Monitoring Internet performance and measuring user quality of experience are drawing increased attention from both research and industry. To match this interest, large-scale measurement infrastructures have been constructed. We believe that this effort must be combined with a critical review and calibrarion of the tools being used to measure performance.

In this paper, we analyze the suitability of `ping` for delay measurement. By performing several experiments on different source and destination pairs, we found cases in which ping gave very poor estimates of delay and jitter as they might be experienced by an application. In those cases, delay was heavily dependent on the flow identifier, even if only one IP path was used. For accurate delay measurement we propose to replace the ping tool with an adaptation of `paris-traceroute` which supports delay and jitter estimation, without being biased by per-flow network load balancing.

Categories and Subject Descriptors

C.2.3 [**Computer-Communication Networks**]: Network Operations—*Network monitoring*; C.4 [**Performance of Systems**]: Measurement techniques

General Terms

Measurement, Performance

Keywords

Ping; delay; jitter; load-balancing

IMC'13, October 23–25, 2013, Barcelona, Spain.

http://dx.doi.org/10.1145/2504730.2504765.

1. INTRODUCTION

With the Internet carrying more and more critical traffic, network performance becomes ever more important. Hence, network operators need to constantly measure their networks in order to detect and troubleshoot performance degradation which can be experienced by users and applications.

Growing interest in network performance measurement has translated into the deployment of a number of large-scale end-to-end measurement infrastructures such as the RIPE Atlas [13], RIPE TTM [14], BISmark [19] and M-Lab projects [7], and infrastructures such as the one of SamKnows probes [17]. Those infrastructures make the issues of how we measure Internet performance even more significant. For example, few basic measurement tools, like `ping`, `traceroute`, and `paris-traceroute`, can be used in performance measurements from Atlas probes.

We believe that it is imperative to calibrate these basic tools to ensure accurate characterization of Internet performance. We focus on ping, one of the most commonly used tools to measure delay and jitter.

By running a few manual experiments with ping, we discovered an unexpectedly high variance in the measurements even when they were conducted within a single ISP. Hence, we decided to take a more rigorous approach to understand whether such a surprising delay variability depended on some specific features of the network, on some bias of ping itself, or both.

In this paper, we assume that a set of fields, identifying the flow to which a packet belongs, is typically used by network devices to perform load-balancing. We rely on Augustin et al. [1] definition's of flow. We discovered that most of the delay variability that ping reported was due to ping sending probes belonging to different flows. This variance is likely due to diversity and redundancy of paths at different layers. In contrast, the delay variability was much less for probes belonging to the same flow. From an application perspective, this means that delay and jitter can vary from flow to flow, that is, the network may not perform as expected from ping results on specific flows. More importantly, applications that use several transport channels (e.g., to transport different audio, video, and data streams as in videoconferencing) should not assume that delay is consistent across channels.

The remainder of the paper is structured as follows. We cover background information and illustrate our measurement methodology in Section 2. We describe the results of our experiments in Section 3. We compare to related work in Section 4. Finally, we discuss the implications of our findings in Section 5.

2. MEASURING PER-FLOW DELAY

In this section, we describe some background, introduce the `tokyo-ping` tool that we developed and refined based on the work of paris-traceroute [1], and we detail our measurement methods.

2.1 Background

Tuples of source IP address, source port, destination IP address, and destination port are used to identify TCP flows. Load-balancing and redundancy mechanisms, such as *Equal Cost Multi-Path (ECMP)* and *Link Aggregation Group (LAG)*, commonly rely on hashes over these tuples to map an incoming packet to an outgoing physical interface. Routers performing layer-4 hashing often use bytes 12-19 of the IP header and bytes 1-4 of the IP payload [1]. In the following, we refer to a single combination of those twelve bytes as a *flow-id*. Note that the flow-id of an ICMP packet is composed of the type, code, and checksum.

Source port	**Destination port**
Length	Checksum

Figure 1: UDP header [15]. Fields in bold are part of the flow-id.

Type	**Code**	**Checksum**
Identifier		Sequence Number

Figure 2: ICMP echo message [16]. Echo request messages have type=8 and code=0. Echo reply messages have type=0 and code=0.

Type	**Code**	**Checksum**
unused (zero)		
IP Header + 64 bits of payload		

Figure 3: ICMP port unreachable message [16]. Type and code fields are both set to 3.

In our experiments, we used both UDP and ICMP probes. Figs. 1 and 2 show the structure of a UDP and an ICMP probe, respectively. When replying to an ICMP probe, the target host simply echoes the ICMP payload back in an echo reply message, which looks exactly the same as the probe except the type field is set to 8 instead of 0 (Fig. 2). When replying to a UDP probe, a target host generates an ICMP port unreachable message (Fig. 3) including the offending IP header and the first eight bytes of the offending IP payload, which map to the UDP header. Classic ping and traceroute emit probes that do not keep the flow-id constant, so their output is affected by the presence of load balancing. Paris-traceroute [1] is a traceroute-like tool which overcomes this limitation by keeping the fields that contribute to the flow-id (i.e., the fields in bold in Figs. 1 and 2) set to user-specified constants.

2.2 Adapting Paris-Traceroute

To isolate delay behavior of different flows, we used a modified version of paris-traceroute, which we called `tokyo-ping` [5]. The tokyo-ping tool reports delay as the Round-Trip Time (RTT) between a given source-destination pair using a user-specified flow-id. The main difference from paris-traceroute is that our tool keeps the flow-id of the return path constant when probing servers. Tokyo-ping supports both UDP and ICMP probes, and can be configured to emit probes with the same length as ping probes.

To measure RTT, tokyo-ping considers the flow-id of both the probes and the responses. For ICMP probes, responses are automatically guaranteed to keep a constant flow-id. In fact, a response to an ICMP probe contains the same payload as the probe, but has a different type value, hence a different checksum (Fig. 2). The return flow-id cannot be controlled by the probe source, making it impossible to explore the return paths. However, there is a one-to-one mapping between the flow-id of the probe and the flow-id of the response.

In general, the same does not hold for UDP probes. For UDP probes, the flow-id of the response depends on the payload of the ICMP error message (see Fig. 3), which is the IP header followed by the first eight bytes of the probe, i.e., the UDP header (Fig.1). Note that the UDP payload influences the UDP checksum, which in turn influences the ICMP checksum in the response. The original paris-traceroute only supports control of the return flow-id when targeting routers. We extend the technique in [1] to predict the ICMP message generated at a destination host (rather than an intermediate router) and then craft the UDP payload of the probe to keep the UDP checksum constant, yielding a constant return flow-id.

Further, using UDP probes with tokyo-ping, we were able to isolate the separate contributions of the forward and return paths to the RTT variability. This has been done by comparing the RTT on paths with the same forward path and on paths with the same return path (see Section 3.4).

Unfortunately, tokyo-ping is unable to control the return flow-id when targeting some operating systems (e.g., Linux) that include the full IP payload in ICMP error messages [2]. In this case, crafting the UDP payload makes little sense because the sum of the UDP data and UDP checksum is constant, so we cannot control the flow-id of the response. In such cases, we resorted to ICMP probes.

2.3 Measurement Methodology

We used tokyo-ping, experimenting with both ICMP and UDP probes, to measure different flows between source-destination pairs. Probes were sent with a TTL value large enough not to expire before reaching the target host.

During each run, we sent probes with 100ms spacing in order to reduce the likelihood of ICMP rate limiting at the destination. For each run, we sent 100 sequential classic pings followed by 100 probes for each different flow-id to be tested. We repeated this procedure 100 times. The interleaving of probes reduced the risk of having specific flow-ids biased by temporary network events during the experiment (e.g., routing changes). In such cases multiple or all flow-ids were likely to be affected by the event. Additionally, running the experiment 100 times improved statistical significance. Finally, we compared the distribution of RTTs returned by normal ping with the distribution of RTTs obtained for each

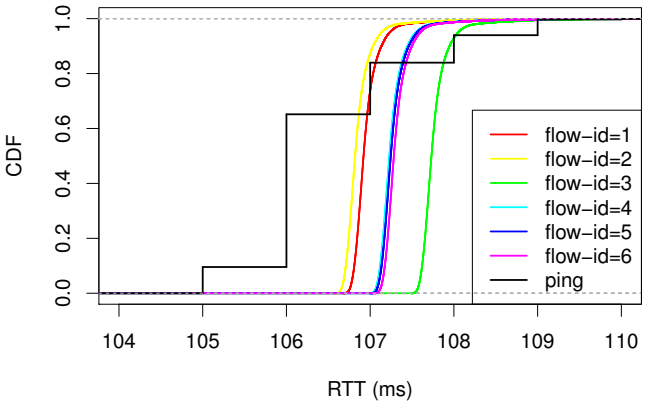

Figure 4: Per-flow and ping measured RTT from a source in Italy to a destination in the US.

distinct value of flow-id. Each experiment was repeated several times, on different days, and at different times of day, to avoid bias due to timing and traffic patterns.

Our study differs from [1] in the following aspects. First, we focused on RTT rather than on IP-level topology discovery. We made no attempt to discover paths to intermediate routers. Second, we targeted destination hosts instead of tuning the TTL to expire at the last router. This allowed us to be much more confident of the distribution of RTTs, as we were immune to the very common ICMP throttling by routers. Further, as ICMP TTL expired message generation is done in software, the load of the destination router strongly influences the RTT, adding to the variability of the distribution. This was not the case with our method. Third, we did not assume a one-to-one mapping of flow-id to IP path, in fact, the opposite. We probed different flow-ids even when the IP-level path was exactly the same. Our experiments showed that, in some cases, a single IP-level path could exhibit significantly different per-flow RTT distributions. Finally, we avoided the use of virtual machines as sources and destinations, to avoid virtualization overhead's effect on fine grained timing.

3. PING, RTTS, AND JITTER

In this section we report results of our methods when applied to measure different source-destination pairs. First, we performed measurement in controlled environments, where we had ground truth knowledge of the traversed paths. Then, we conducted larger-scale experiments. We found that differences in performance between different flow-ids can be significant. As a consequence, ping is in general a mediocre estimator for RTTs and heavily overestimates jitter.

3.1 From Italy to the US

Our first experiment ran tokyo-ping from a server in Italy to a destination in the US, using 6 different flow-id values. Results are depicted in Fig. 4 where each curve represents a Cumulative Distribution Function (CDF). The black staircase is the CDF of RTTs as measured by ping. Each data point (x, y) indicates that a y-fraction of ping probes observed an RTT of at most x milliseconds. The ping CDF is quantized because the Linux version of ping imposes a three digit precision on the reported RTTs. Each colored curve represents the CDF of RTTs measured for different flow-ids. Note that the RTT range between different flow-ids is small (4ms). The same does not always hold in the experiments we performed.

We stress that the RTT variability measured within each individual flow-id is quite low, especially compared to the variability measured by ping. In particular, ping reports jitter approximately five times greater than the largest per flow-id jitter. The apparent discrepancy of ping measuring lower RTTs than the minimum observed by setting the flow-ids is due to the fact that the six flow-id values did not cover the full range of treatment that the flows might experience. Repeating the experiment with 32 flow-ids resulted in the ping RTTs being in the same range as the RTTs measured setting the flow-ids.

Seeing this difference in performance among different flow-ids, we decided to use paris-traceroute to enumerate the IP-level paths between the two hosts. We found a surprisingly high number of IP paths, mainly due to Equal Cost Multi-Path (ECMP) configured over transoceanic links by an intermediate Tier-1 ISP.

3.2 Crossing the US using a single ISP

The above experiment showed that, in the presence of different IP paths, there may be differences in the RTT observed by packets of different flows. To ensure we completely understood what was happening, we decided to perform an even more controlled experiment. We sent ICMP probes from a server in Dallas to a server in Ashburn. The probes crossed a single IP path in a single ISP.

Fig. 5 shows the results of this experiment. In this case, the CDF for ping is not quantized because the source is a FreeBSD host running a version of ping that does not round RTT to three digits. In addition to the black staircase showing the CDF of RTT as measured by ping, we plot another staircase (shown in red) which is obtained by merging all of the RTT measures with different flow-ids and applying the same rounding. The red CDF is a very good approximation of the black one. This shows that the RTT distribution of ping is actually a sample from the collection of all distributions for different RTTs. The slight differences between the red and the black staircases are likely because our merged distribution assumes that each flow-id contributes equally to the sample, which might not be the case with ping. Moreover, 32 flow-ids may not be enough to capture the full variability observed by ping.

Looking at the per flow-id curves, we see that the difference between the RTTs of the flows with highest and lowest RTTs is even larger than in the first experiment, while we again observe low variability within each flow-id. Traditional ping can reliably estimate the upper and lower bounds of the distribution, but substantially overestimates jitter.

This result puzzled us. Thanks to the operator, we looked at the router configurations. Our probes crossed three lagged hops, one within the ingress Point-of-Presence (PoP), one across the long-haul core, and the last in the egress PoP. Each bundle was composed of slightly less than ten links. Could LAG cause flows to experience such different RTTs? Experiments between Dallas and Seattle with both ECMP and LAG showed similar behavior. We then ran two experiments with sources and destinations within the same PoP. In the first, the source and destination hosts were both connected to the same router. For the second, one LAG was used between routers in the PoP. Neither of these tests

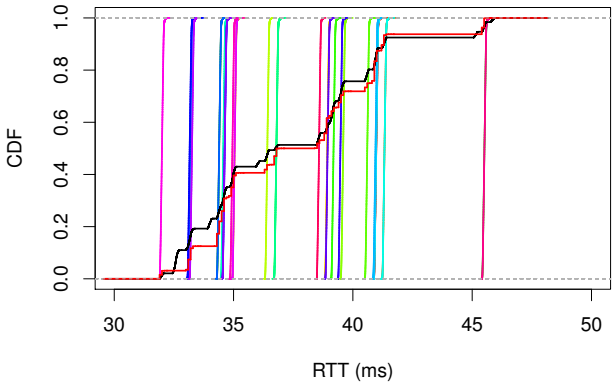

Figure 5: Experiment 2 - Performance of probes with and without fixed flow-id on an single IP path from Dallas to Ashburn.

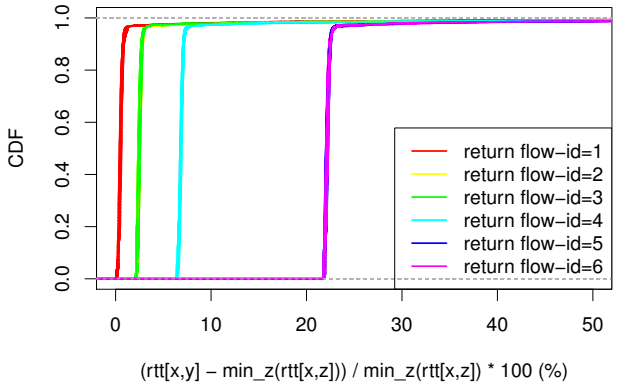

Figure 6: Contribution of return paths. Each color represents a different return flow-id. Data are normalized based on the minimum RTT experienced for a single forward flow-id.

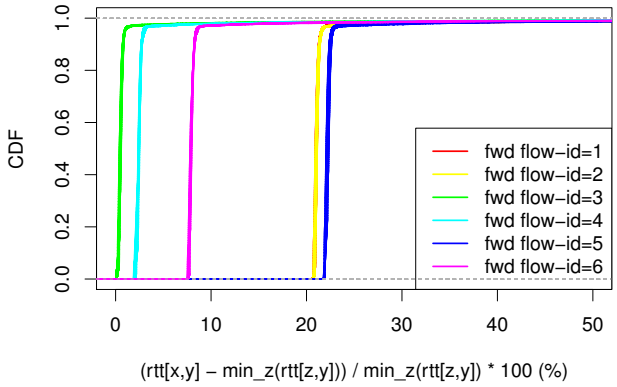

Figure 7: Contribution of forward paths. Each color represents a different forward flow-id. Data are normalized based on the minimum RTT experienced for a single return flow-id.

showed any RTT difference for different flow IDs. This made us suspect that something special happened on the long-haul inter-PoP LAG.

3.3 Pruning out some potential causes

We could see a few potential culprits for the per-flow performance behavior, namely (1) the use of ICMP probes, (2) the traffic load carried by the network, (3) synchronization effects due to the interleaving of probes, (4) MPLS settings, (5) the configuration of hashing functions and hash keys, (6) the diversity of links being bundled, or (7) the LAG vendor implementation. To test whether ICMP probes were the culprit, we sent UDP probes from Ashburn to Dallas. We set the header fields in such a way as to control both the forward and the return path flow-ids. We used 6 different values for the forward flow-id and 6 different values for the return flow-id, resulting in 36 different RTT distributions. In order to quantify the contribution of the forward and return flow-id separately, we normalized the data across forward flow-ids by taking the difference with respect to the minimum RTT value measured with that forward flow-id (and across return flow-id). Fig. 6 shows the CDFs of the RTT differences, where each color represents a different value of the return flow-id. Observe that the distribution of the RTT difference is consistent across different forward flow-ids, indicated by the fact that the distributions for the same return flow-id (i.e., same color) are not scattered around the plot. We perform a similar analysis to isolate the contribution of the forward flow-id, shown in Fig. 7. The ability to clearly isolate the contributions of forward and return flow-ids also indicates that the RTT differences are not measurement artifacts.

Cross-traffic cannot be the culprit for the RTT differences either. Running the Dallas - Ashburn experiment at different times of the day, different days of the week, and for different durations led to the same graphs, with the specific flow-ids always mapping to the corresponding RTT distribution. Discussion with the operator also confirmed that the cross-traffic was very low, as the traffic load never exceeded 50% of link capacity during our experiments. With respect to synchronization effects, we repeated the experiment using a different inter-probe spacing of 157 ms, i.e., a prime number reasonably far from multiples of 100 ms (used in all our experiments), and obtained identical results. We excluded MPLS settings and hashing specifics by checking router configurations in collaboration with network operators. This left us with LAG bundle physical link path diversity, LAG vendor implementation, or more obscure reasons as potential causes.

Despite our precautions to avoid VMs and routers as endpoints, and to cross networks for which we could get ground truth, we have not been able to pinpoint the exact causes of per-flow behavior. We know two major causes, LAG and ECMP. The ECMP issue is obvious, different path lengths, equipment, . . . LAG is more subtle, and can be any combination of varied hashing algorithms on line cards (for resilience, operators usually use multiple line cards on a single LAG), varied circuit paths, etc. And serious diagnosis of LAG variance is severely hampered by lack of vendor instrumentation, e.g. one can not look into the indvidual circuits' queues.

3.4 Collecting More Evidence

To ensure that our experiments did not just discover a few corner cases, we ran ping and tokyo-ping on a larger number of Internet-wide source-destination pairs. Our sources were FreeBSD servers in Dallas, Ashburn and Seattle, plus a Linux server in Rome. As destinations, we used a subset

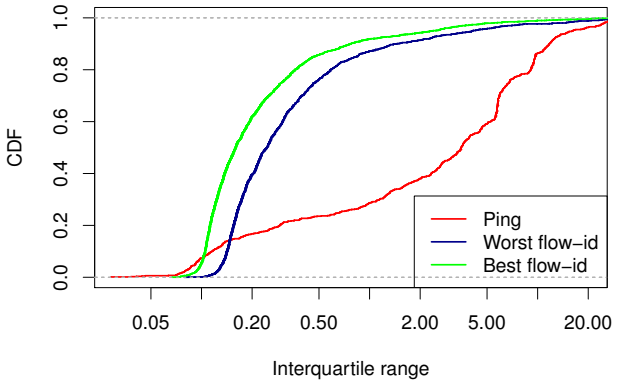

Figure 8: RTT variance with ping and tokyo-ping from a single source to 850 destinations

of the Google servers discovered by [3]. As our goal was to check the generality of the per-flow RTT behavior, we targeted distributed destinations selecting one IP per Autonomous System (AS) in this dataset. This resulted in 850 destinations. We used tokyo-ping to send ICMP probes with 16 different flow-ids. We sent ten probes for each flow-id and ten ping probes. We repeated this experiment 20 times.

Fig. 8 depicts the CDF of the inter-quantile ranges of RTT measurements that we observed using Ashburn as the source. The green curve (on the left) and the blue curve (in the middle) show the lowest and highest per-flow-id inter-quantile ranges, while the red curve shows the inter-quantile range for ping. Note that the variability of the distribution of RTTs reported by ping is systematically higher (with very few exceptions) than the variation of the most variable flow-id. In particular, for 40% of the destinations, the worst per-flow inter-quantile range is above 0.3 ms while for ping is above 5.2 ms. For 15 destinations, ping experienced lower RTT variability than the per-flow measurements (in the bottom-left). The lack of ground knowledge of router configurations and circuit paths in these larger-scale experiments prevented us from finding clear explanations.

We observed a similar behavior for all sources. We also performed experiments toward Alexa's top 100 sites and obtained very similar results.

Which portion of the RTT distribution exhibits per-flow behavior? When extending the range to incorporate 90% of the RTT distribution (the 95th-5th percentile range), instead of the 50% for the inter-quantile, the figure changes. The per-flow variance increases. For most destinations (90%) the worse per-flow variance is higher than the ping variance. This is likely a sign that while there are specific per-flow behaviors, other factors are also at play. Slightly reducing the portion of the RTT distribution we consider is enough to reflect per-flow behavior. From our source in Ashburn, 90% of the destinations have an RTT distribution with 90th-10th range lower with the flow id fixed than for ping. This shows that in general 80% of the flow observations are consistent.

4. RELATED WORK

Traceroute is renown to be error prone in load-balanced networks [1]. In [1], the authors propose a replacement for traceroute, called paris-traceroute, which takes into account path diversity at the IP layer. By measuring the minimum RTT across 100 measurements per destination and iterating over multiple flow identifiers, the authors conclude that, for most paths, there is no significant RTT difference among flow identifiers. This paper is close in spirit and complementary to [1], as we show that ping results are also biased by per-flow load-balancing techniques. We extend [1] in two ways. First, we show that IP is not the only layer to be considered in load-balanced networks (see Section 3). Path diversity may be present at lower layers given LAG or MPLS. Second, by relying on the RTT distribution instead of the minimum observation, we are able to show that ping is a bad estimator for any metric related to the RTT. Essentially, ping systematically overestimates RTT variability, which is significantly lower when measured for individual flow identifiers.

Previous efforts were devoted to better implementation of per-flow load balancing. For example, the MPLS entropy label [10] is intended to avoid routers going through deep packet inspection to perform per-flow balancing, which enables support of high traffic rates in core routers. [20] highlights the difficulties in hashing encapsulated traffic.

The effect of using different hash functions to balance traffic on multiple links is studied in [4], and best practices for optimal LAG/ECMP component link utilization while using hash-based techniques are described in [11]. In [21], the use of a given path in data centers, e.g., to avoid congestion, is achieved by modifying packet headers on the basis of traceroute-like results.

We show that predictability is a key feature for new load balancing methods, especially to ease diagnosis. From this perspective, we find interesting the load balancing technique proposed in [6]. Work on balancing traffic is of major importance as it is commonly used in ISP networks [1, 8] as well as in data centers [9, 12].

5. LESSONS LEARNED

We have shown that using ping for delay measurement may be biased if one ignores flow-based load balancing. This bias is intrinsic to ping's design, hence predictable *a priori*. In carefully crafted measurements, the dispersion reported by ping can be up to 40% of the RTT experienced by the flow with lowest latency. In other words, when we observe high variability in the delay measurements of ping, it is likely a measurement artifact of the tool itself.

This observation has several consequences.

1. **Ping is unfit to measure RTT distributions.** The distribution measured by ping is often a sample from a wider set of per-flow distributions. While this can identify upper and lower bounds for the delay with good approximation, it provides a mediocre estimate for delay. It overestimates jitter (or any other metric measuring the variability of the distribution). For this reason, it cannot reliably represent the performance experienced by applications. This should sound a warning both for researchers studying end-to-end Internet performance, and operators using ping for measurement or debugging purposes. We suggest that tokyo-ping, an adaptation of paris-traceroute, be deployed on large-scale measurement infrastructures.

2. **The importance of the flow identifier.** That a significant difference in latency may exist between flows for the same source and destination pair represents both a danger and an opportunity for applications. On the one hand, applications using multiple transport channels, e.g., the con-

trol, video, audio and data channels found in videoconferencing and streaming, cannot assume that network performance is consistent across channels. This implies that multi-channel applications are advised not to rely on a single control channel to accurately estimate delay and jitter of all opened TCP connections. If the application needs consistency across channels (e.g. to keep the lip sync in video streaming), SCTP [18] is a good alternative as it is able to multiplex streams while keeping a constant flow identifier. On the other hand, applications might experience better performance by carefully selecting the source and destination ports, e.g., during an initial negotiation phase. Moreover, our findings suggest that accurately monitoring per-channel performance from outside the application is harder than is commonly believed. Indeed, the performance that a monitoring tool (e.g., ping, IP-SLA, etc.) measures is not necessarily representative of the performance experienced by specific applications.

3. **Impact on common wisdom and prior work.** Our findings show how reality is often more complex than expected. Technologies and configurations at different layers of the protocol stack often interact in unexpected ways. Understanding these interactions and the behavior of different vendor implementations is difficult. This can frustrate or make inaccurate the modeling effort of research. Our experiments show that, at least in some cases, latency over a network path is not a well-defined concept, and we should more precisely define latency over a transport session. Generally speaking, we recommend researchers be cautious when drawing conclusions from experiments based solely on ping results.

Acknowledgements

We thank Olivier Bonaventure and Jean Lorchat for their insightful comments. We are very grateful to several network operators without whom this work would not have been possible. We are indebted to our shepherd Ethan Katz-Bassett for his guidance in producing the camera-ready. This work is partially supported by the European Commission's Seventh Framework Programme (FP7/2007-2013) Grant No. 317647 (Leone).

6. REFERENCES

[1] B. Augustin, T. Friedman, and R. Teixeira. Measuring load-balanced paths in the internet. In *ACM Internet Measurement Conference (IMC 2007)*, 2007.

[2] F. Baker, 1995. Internet Engineering Task Force (IETF) , RFC 1812.

[3] Matt Calder, Zi Hu Xun Fan, Ethan Katz-Bassett, John Heidemann, and Ramesh Govindan. Mapping the Expansion of Google's Serving Infrastructure (To Appear). In *Proceedings of the ACM Internet Measurement Conference (IMC '13)*, October 2013.

[4] Z. Cao, Z. Wang, and E. Zegura. Performance of hashing-based schemes for Internet load balancing. In *INFOCOM*, pages 332–341, 2000.

[5] Luca Cittadini. Tokyo-ping. `http://psg.com/tokyo-ping-v2.tar.gz`.

[6] Gregory Detal, Christoph Paasch, Simon van der Linden, Pascal MÃl'rindol, Gildas Avoine, and Olivier Bonaventure. Revisiting flow-based load balancing: Stateless path selection in data center networks. *Computer Networks*, 57(5):1204–1216, April 2013.

[7] C. Dovrolis, K. Gummadi, A. Kuzmanovic, and S. D. Meinrath. Measurement Lab: Overview and an Invitation to the Research Community. *SIGCOMM CCR 2010*, 2010.

[8] T. Flach, E. Katz-Bassett, and R. Govindan. Quantifying violations of destination-based forwarding on the Internet. In *ACM Internet Measurement Conference (IMC 2012)*, 2012.

[9] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta. VL2: a scalable and flexible data center network. In *ACM SIGCOMM*, 2009.

[10] K. Kompella, J. Drake, S. Amante, W. Henderickx, and L. Yong. The Use of Entropy Labels in MPLS Forwarding, November 2012. Internet Engineering Task Force (IETF), RFC 6790.

[11] R. Krishnan, S. Khanna, L. Yong, A. Ghanwani, Ning So, and B. Khasnabish. Mechanisms for optimal LAG/ECMP component link utilization in networks, April 2013. draft-krishnan-opsawg-large-flow-load-balancing-08.txt, work in progress.

[12] J. Mudigonda, P. Yalagandula, M. Al-Fares, and J. C. Mogul. Spain: Cots data-center ethernet for multipathing over arbitrary topologies. In *NSDI*, 2010.

[13] RIPE NCC. RIPE Atlas. `https://atlas.ripe.net`.

[14] RIPE NCC. RIPE Test Traffic Measurement Service. `https://atlas.ripe.net`.

[15] J. Postel. User Datagram Protocol, 1980. Internet Engineering Task Force (IETF) , RFC 768.

[16] J. Postel. Internet Control Message Protocol, 1981. Internet Engineering Task Force (IETF) , RFC 792.

[17] SamKnows. Samknows. `http://www.samknows.com`.

[18] R. Stewart. Stream Control Transmission Protocol, 2007. Internet Engineering Task Force (IETF) , RFC 4960.

[19] Georgia Tech et al. Project BISmark. `http://projectbismark.net`.

[20] Jeff Wheeler and Job Snijders. Understanding MPLS hashing. NANOG 57, February 2013.

[21] K. Xi, Y. Liu, and J. Chao. Enabling flow-based routing control in data center networks using probe and ECMP. In *INFOCOM Workshop on cloud computing (2011)*, page 614âĂŞ619, 2001.