Minimally-Intrusive Frequent Round Trip Time Measurements Using Synthetic Packet-Pairs

Sebastian Zander, Grenville Armitage
Centre for Advanced Internet Architectures (CAIA)
Swinburne University of Technology
Melbourne Australia
{szander, garmitage}@swin.edu.au

***Abstract*—Accurate and frequent round trip time (RTT) measurements are important in testbeds and operational networks. Active measurement techniques inject probe packets that may modify the behaviour of the observed network and may produce misleading RTT estimates if the network handles probe packets differently to regular packets. Previous passive measurement techniques address these issues, but require precise time synchronisation or are limited to certain traffic types. We introduce Synthetic Packet-Pairs (SPP), a novel passive technique for RTT measurement. SPP provides frequently updated RTT measurements using *any* network traffic *already present* in the network *without* the need for time synchronisation. SPP accurately measures the RTT experienced by any application's traffic, even applications that do not exhibit symmetric client-server packet exchanges. We experimentally demonstrate the advantages of SPP.**

***Index Terms*—Passive measurement, round trip time, RTT.**

I. Introduction

The measurement of a path's round trip time (RTT) is often crucial when evaluating and improving the performance of network protocols or network equipment. Also, for service providers it is becoming increasingly important to monitor and manage the RTT experienced by highly interactive applications, such as voice over IP, latency-sensitive first person shooter (FPS) games [1] or mission-critical business applications. Existing RTT measurement techniques differ in whether they are active (injecting additional traffic into the network) or passive (using traffic already in the network), their achievable sample rate (RTT measurements per time unit), whether they rely on synchronised clocks and their ability to determine the RTT experienced by flows from different applications.

High rate sampling of a path's RTT is important for detailed observation of RTT versus time, such as watching bottleneck queues filling and draining or the rise and fall of contention-induced transmission delays. Active techniques find this problematic, as the additional probe traffic (proportional to the desired sample rate) can interfere with the characteristics of the path being probed and skew the measured RTTs. Active techniques may also produce misleading RTTs across network paths that differently prioritise the forwarding of IP packets according to their higher-layer protocol and/or port numbers. It is hard to measure the RTT an application flow sees when the probe packets experience different forwarding delays [2].

Passive techniques can potentially sample at close to the existing traffic rate with no new load on the path and measure the actual RTTs of an application flow's traffic. However, passive one way delay (OWD) measurements require observation points with precisely synchronised clocks, and previously proposed passive RTT measurement techniques require the enhancement of routers along a path, manipulation of packet contents in flight, or rely on symmetric request-response patterns from the observed traffic.

We introduce a novel passive RTT measurement technique, called "synthetic packet-pairs" (SPP)[1], that:

- Calculates RTT of a network path using *existing* traffic.
- Does *not* require clock synchronisation.
- Is protocol-agnostic and works with asymmetric traffic flows (flows without request-response behaviour).
- Provides accurate RTT samples at a rate proportional to the slowest traffic rate in either direction.
- Does not require changes on network routers or modifications of packet contents.

SPP is beneficial in scenarios where the path under test is sensitive to the additional load of active probing, frequent sampling of the path is desired (at sample rates approaching the packet rates of existing traffic), tightly synchronised measurement point clocks are unavailable (for cost or deployment reasons), or probe traffic is treated differently from the application traffic under study. SPP is also advantageous where detailed RTT estimates are desired for delay or jitter estimation, particularly for interactive applications that do not have symmetric request-response packet pairs.

SPP is non-intrusive where RTT is measured in real-time and a separate communication path exists between the measurement points, or where RTTs can be computed offline (e.g. testbed measurements). Otherwise, it is minimally intrusive, since the packet timestamp data that needs to be transmitted uses relatively little bandwidth. We have released an open-source implementation [4] and a more detailed report [5].

Section II outlines the challenges and requirements for RTT measurement schemes. Section III describes our proposed SPP algorithm. Section IV experimentally demonstrates the advantages of SPP. Section V concludes the paper.

[1] SPP is unrelated to Keshav's packet-pair available bandwidth probing [3].

978-1-4799-0537-9/13/$31.00 ©2013 IEEE

II. Background and Motivation

Here we briefly review existing RTT measurement techniques – see [5] for a more detailed discussion.

A. Active RTT measurement

Active techniques (such as `ping`) inject extra packets (probes) across a network path and use their transit times to measure the path's delay at the instant each probe was sent.

Unfortunately, routers or switches along the path may handle active probe packets differently to regular traffic, resulting in unrepresentative RTTs. For example, when routers handle ICMP packets in their slow path `ping` can overestimate a path's RTT [2]. Bottleneck routers may also prioritise specific application flows during times of congestion. Active probe packets that do not match the rules will generate RTTs that do not reflect the RTT experienced by prioritised traffic.

Active measurement schemes add network load proportional to the desired RTT sample rate, which can noticeably alter the network's behaviour and performance. This is particularly problematic when high-frequency measurements are desired, for example to capture the impact of one or more TCP flows congesting a bottleneck queue, or observe the impact of TCP cross-traffic on time-sensitive flows through shared bottlenecks. It is also problematic over link technologies, such as 802.11 Wireless LANs, where modest loads in packets per second cause noticeable degradation [6].

B. Passive RTT measurement

By measuring the delays experienced by existing traffic, passive measurement techniques avoid adding load and can identify the RTTs experienced by application flows subject to different forwarding priorities. The sample rate depends on the rate at which traffic traverses the path. Passive schemes cannot sample idle paths, but in this case test traffic can be generated.

Some techniques measure one-way delay (OWD) by noting the time it takes an IP packet to transit between two measurement points having precisely synchronised clocks (e.g. [7], [8]) and RTT can be calculated by adding OWD in each direction. However, OWD techniques either require precise time synchronisation (which is often complex and potentially costly to deploy), require modifications to existing routers [9] or manipulate the content of existing packets [10].

Other techniques estimate RTT at a single measurement point from the time it takes for an application request packet in one direction to be answered by a matching response packet in the return direction, e.g. [11]–[13]. They require symmetric client-server traffic with enough information in each packet to match request-response pairs. Also, these techniques fail when traffic in each direction has no predictable timing or semantic relationship, for example many client-server FPS games emit packets in each direction asynchronously [1].

III. SPP Measurement Technique

SPP fills a gap in the existing set of RTT measurement techniques, and was initially documented in a technical report [14]. No new traffic is injected, no modifications are required to existing infrastructure, and traffic in each direction may be asymmetric and from unrelated application flows. Here we mostly describe the technique, our prototype implementation is discussed in [4], [5].

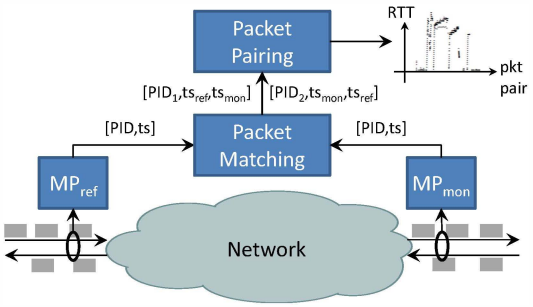


Figure 1: SPP overview: packet capturing, packet matching and packet-pair identification

Figure 1 illustrates the three stages of SPP – capturing packets at two measurement points (MPs), matching packets seen at both MPs, and identifying packet-pairs from which to calculate RTTs.

A. Measurement Points and Packet IDs

We will describe SPP in terms of two MPs, $\mathrm{MP_{ref}}$ (reference) and $\mathrm{MP_{mon}}$ (monitor), located so that the network traffic of interest traverses both points. $\mathrm{MP_{ref}}$ passively records the passing of packets heading towards $\mathrm{MP_{mon}}$, for example by monitoring traffic using in-line network taps or mirrored ports on a switch. $\mathrm{MP_{mon}}$ performs the same action for packets heading towards $\mathrm{MP_{ref}}$.

For every recorded packet both $\mathrm{MP_{ref}}$ and $\mathrm{MP_{mon}}$ log a timestamp (ts) representing when the packet was seen and a short 'packet ID' (PID). The PID is calculated from a suitable hash function (e.g. CRC32 or other hash functions in [8], [15]) over key bytes within the packet. To uniquely identify the same packet passing $\mathrm{MP_{ref}}$ and $\mathrm{MP_{mon}}$ the PID is based on portions of a packet that are invariant during transit between $\mathrm{MP_{ref}}$ and $\mathrm{MP_{mon}}$ but vary between different packets (cf. [7], [8]).[2] Each MP accumulates a list of [PID,ts] pairs based on the captured packets. These two lists are then combined to create packet-matched lists used for packet-pair identification and RTT calculation.

B. Packet Matching

Figure 1 shows the [PID,ts] lists being combined at a third location, but they can also be combined at $\mathrm{MP_{ref}}$ or $\mathrm{MP_{mon}}$. If the link used to transmit [PID,ts] pairs is a physical out-of-band link or a logically isolated channel, the [PID,ts] pairs may be combined in real-time without affecting the observed network. If no separate channel is available, [PID,ts] pairs may

[2]Our prototype uses the CRC32 hash function. It supports selecting which IP and TCP/UDP header fields are used as hash input, and specifying any network address translation (NAT) along the path. This allows handling situations where $\mathrm{MP_{ref}}$ and $\mathrm{MP_{mon}}$ are placed at either side of a middlebox performing NAT, NAT with port translation or acting as a firewall (which may manipulate TCP header fields, such as the TCP sequence number).

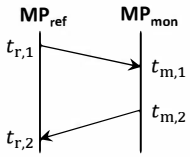


Figure 2: Packet pair and corresponding timestamps

be stored at each MP for later transfer across the measured network (e.g. during non-measurement or off-peak periods). In these cases SPP is non-intrusive. If a channel along the same infrastructure being measured is used during the measurement, SPP is minimally intrusive in the sense that transmitting [PID,ts] lists uses relatively little bandwidth.[3]

Two input streams I_{mon} and I_{ref} represent [PID,ts] pairs from packets captured at MP_{mon} and MP_{ref} respectively. A queue Q_{ref} is used to buffer [PID,ts] pairs from I_{ref} not yet detected in I_{mon}. The algorithm processes packets from I_{mon} in the order of their arrival at MP_{mon}.

For each packet P_{cur} captured at MP_{mon} it searches for a packet with the same PID captured at MP_{ref}. The algorithm first checks if a matching packet is found in Q_{ref}. If not, new packets are read from I_{ref} into Q_{ref} until a packet matches, the maximum queue length is reached or a packet's timestamp differs by more than a pre-defined T_{delta} from the timestamp of P_{cur}.[4] Before the next packet of I_{mon} is processed all packets in Q_{ref} whose timestamps differ by more than T_{delta} from the timestamp of P_{cur} are considered lost and removed from Q_{ref}.

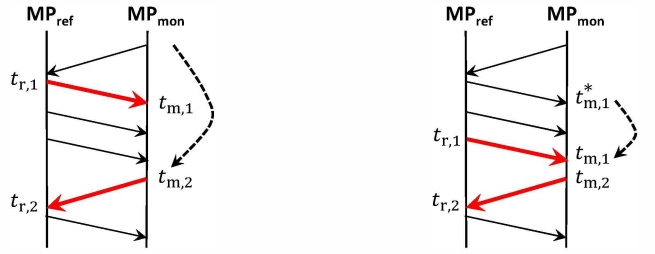
The result of the packet matching is a list of $[\text{PID}_1,\text{ts}_{\text{ref}},\text{ts}_{\text{mon}}]$ tuples, representing the time potential first (1) packets of pairs were seen at MP_{ref} and MP_{mon} respectively. The same algorithm is also run with the directions reversed to construct a list of $[\text{PID}_2,\text{ts}_{\text{mon}},\text{ts}_{\text{ref}}]$ tuples, representing potential second (2) packets of pairs that were seen flowing from MP_{mon} to MP_{ref}.

C. Packet Pair Identification and RTT Computation

To explain our packet-pairing algorithm we first define a short-hand notation for the timestamp $t_{i,j}$, where i indicates MP_{ref} (r) or MP_{mon} (m) and j indicates whether it is the first (1) or second (2) packet of a packet pair. For example, in Figure 2 $t_{\text{r},1}$ is the timestamp of the first packet at MP_{ref} and $t_{m,1}$ is the timestamp of the first packet at MP_{mon}.

1) Packet pairs: SPP makes the key assumption that each packet is used in at most one pair. Otherwise, different RTT estimates would be dependent. Furthermore, SPP ensures that the two packets of a pair are as close together as possible. Packet pairs can overlap in time, since otherwise the measurement frequency would be limited by the RTT of the path.

SPP starts with the first packet from the list of packets going from MP_{ref} to MP_{mon}. It then searches in the second packet list (packets going from MP_{mon} to MP_{ref}) for the first packet where the condition $t_{\text{m},2} > t_{\text{m},1}$ is true. A packet pair has now been identified (Figure 3a), but this pair is not necessarily the closest pair. To find the closest pair, $t^*_{\text{m},1}$ is set to $t_{\text{m},1}$ and the algorithm traverses further through the first list in search for any packets where $t_{\text{m},1} > t^*_{\text{m},1}$ and $t_{\text{m},1} < t_{\text{m},2}$. As long as such packets are found the algorithm advances in the first list. This ensures there are no other packets between the two packets of a pair (Figure 3b).



(a) Finding a packet pair (b) Finding the closest pair

Figure 3: SPP packet pairing approach

2) Computing RTT: Once a packet-pair has been identified the RTT computation is straightforward:

$$\text{RTT} = (t_{\text{r},2} - t_{\text{r},1}) - (t_{\text{m},2} - t_{\text{m},1}) \ . \qquad (1)$$

The packet pairing algorithm then continues with the next packet in the first list (packets from MP_{ref} to MP_{mon}). Note that the two directions could be reversed and the RTTs could be computed in the other direction if desired.

Equation 1 is based on time differences of the same clocks, so there is no need for the clocks at MP_{ref} and MP_{mon} to be synchronised over long time frames. Short-term clock *skew* during the intervals $(t_{\text{r},2} - t_{\text{r},1})$ at MP_{ref} and $(t_{\text{m},2} - t_{\text{m},1})$ at MP_{mon} may introduce error, but typically this error is very small [5].

A key benefit of SPP is that the packets making up each pair in Equation 1 need not be generated by the same application or hosts. That allows SPP to estimate RTT from any packet flows regardless of whether or not there is symmetric request-response behaviour. SPP can even create RTT measurements from two unrelated unidirectional flows in different directions.

IV. Experimental Evaluation

The detailed report [5] describes several scenarios where SPP proves to be valuable. Here we focus on the scenario where traffic prioritisation prevents active probing from measuring application-specific RTTs, and also show the benefits of high passive sample rates for observing rapid RTT transitions.

Traffic prioritisation is increasingly used to minimise the latency for specific traffic through bottleneck routers. The actual classification of flows may occur using IP addresses, TCP/UDP port numbers, direct payload inspection or the traffic's statistical properties (such as packet length distributions [16]). Active probes will be unable to measure the prioritised traffic's RTT, since the active probe packets are unlikely

[3] Our prototype transmits 248 [PID,ts] pairs in one 1500 byte packet [4].

[4] SPP does not require time synchronisation, but we use T_{delta} to drastically improve the packet-matching performance. For each packet observed at MP_{mon} only a T_{delta} time window of packets from MP_{ref} needs to be searched. In the prototype T_{delta} is configurable [4].

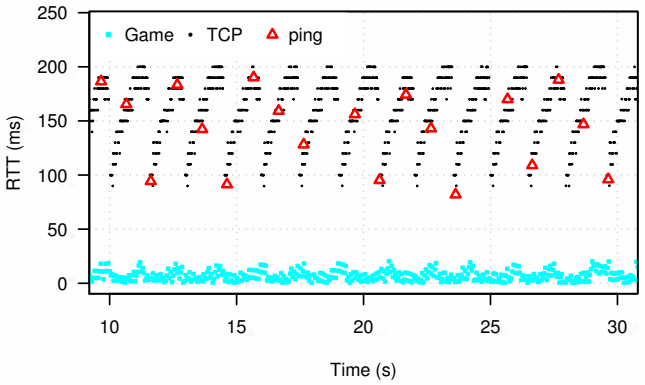


Figure 4: RTT of prioritised FPS online game traffic and TCP cross traffic measured by SPP, and RTT measured by ping (not classified as interactive traffic)

to meet the classifiers rules for header fields (addresses and ports) let alone statistical properties. On the other hand, SPP can measure the prioritised traffic's RTT directly.

As an example, we classified traffic through an OpenWRT-based home router into interactive and non-interactive classes based on packet length statistics [17] and prioritised the interactive traffic on an emulated 1 Mbps ADSL upstream link (using Dummynet). We simultaneously generated FPS game traffic, TCP cross-traffic with Iperf (to congest the router), and actively measured RTT with ping. Figure 4 shows a 20-second time window of game and TCP flow RTTs (measured with SPP) and ping's RTT measurements. Ping sees the cyclical high RTT caused by the TCP cross-traffic, but misses the low RTT experienced by the prioritised game traffic.

Figure 4 also demonstrates that ping misses the detailed RTT fluctuations induced by TCP. SPP's ability to provide frequent RTT measurements is especially useful if one is interested in time series measurements, such as investigating TCP congestion control behaviour. Detailed insights can also be obtained using high probe rate active measurements, but to the potential detriment of the path under observation. In our scenario active probing at a rate of the TCP RTTs measured by SPP would require almost 60 pings/second.

V. Conclusions

Evaluating the performance of network protocols or network equipment often requires accurate RTT measurements. Active measurement techniques are useful but limited – they add traffic in proportion to their probe rate, and probe packets may experience different RTTs to regular traffic. Previous passive measurement techniques require precisely synchronised clocks at different measurement points (MPs), router modifications or traffic with symmetric request-response behaviour.

We proposed a novel passive technique called 'synthetic packet-pairs' (SPP) that measures the RTT of a network path using whatever symmetric or asymmetric two-way traffic exists between two unsynchronised MPs. We have publicly released an open-source implementation [4], which can combine traffic observations from MPs in near real-time or offline. We showed that SPP provides accurate RTT measurements at a rate proportional to the two-way traffic being observed. This enables tracking RTT fluctuations over link technologies that are sensitive to excess traffic loads, measuring latency on paths where active probe packets are treated differently to regular traffic, and seeing short-term RTT transients that are invisible to low-rate active probing.

Acknowledgements

We thank Thuy Nguyen, Lutz Mark and Brandon Tyo for contributing to the initial SPP development, Amiel Heyde for implementing the public SPP prototype, David Hayes, Atwin Calchand and Chris Holman for subsequent bug fixes, and Nigel Williams for helping with the OpenWRT experiments.

References

[1] G. Armitage, M. Claypool, P. Branch, *Networking and Online Games – Understanding and Engineering Multiplayer Internet Games*. John Wiley & Sons, April 2006.

[2] K. Auerbach, "Limitations of ICMP Echo for network measurement." InterWorking Labs, April 2004. http://iwl.com/white-papers/75-limitations-of-icmp-echo-for-network-measurement.

[3] S. Keshav, "Packet pair flow control," 1995. http://blizzard.cs.uwaterloo.ca/keshav/home/Papers/data/95/pp.pdf.

[4] A. Heyde, "SPP Implementation." http://caia.swin.edu.au/tools/spp/.

[5] S. Zander, G. Armitage, "Minimally-Intrusive Frequent Round Trip Time Measurements Using Synthetic Packet-Pairs – Extended Report," Tech. Rep. 130730A, Centre for Advanced Internet Architectures, Swinburne University of Technology, 2013.

[6] T. Nguyen and G. Armitage, "Quantitative Assessment of IP Service Quality in 802.11b Networks and DOCSIS networks," in *Australian Telecommunications Networks & Applications Conference (ATNAC)*, pp. 121–128, December 2004.

[7] I. D. Graham, S. F. Donnelly, S. Martin, J. Martens, J. G. Cleary, "Nonintrusive and Accurate Measurement of Unidirectional Delay and Delay Variation on the Internet," in *Internet Summit (INET)*, July 1998.

[8] T. Zseby, S. Zander, G. Carle, "Evaluation of Building Blocks for Passive One-way-delay Measurement," in *Passive and Active Measurement Workshop*, April 2001.

[9] R. R. Kompella, K. Levchenko, A. C. Snoeren, G. Varghese, "Every Microsecond Counts: Tracking Fine-grain Latencies with a Lossy Difference Aggregator," in *ACM SIGCOMM Conference on Data Communication*, pp. 255–266, 2009.

[10] M. Cola, G. De Lucia, D. Mazza, M. Patrignani, M. Rimondini, "Covert Channel for One-Way Delay Measurements," in *International Conference on Computer Communications and Networks*, August 2009.

[11] J. Jiang, C. Dovrolis, "Passive Estimation of TCP Round-trip Times," *ACM Computer Communication Review (CCR)*, vol. 32, pp. 75–88, 2002.

[12] S. Jaiswal, G. Iannaccone, C. Diot, J. Kurose, D. Towsley, "Inferring TCP Connection Characteristics Through Passive Measurements," in *IEEE INFOCOM*, March 2004.

[13] J. But, U. Keller, D. Kennedy, G. Armitage, "Passive TCP Stream Estimation of RTT and Jitter Parameters," in *IEEE 30th Conference on Local Computer Networks (LCN)*, November 2005.

[14] S. Zander, G. Armitage, T. Nguyen, L. Mark, B. Tyo, "Minimally Intrusive Round Trip Time Measurements Using Synthetic Packet-pairs," Tech. Rep. 060707A, Centre for Advanced Internet Architectures, Swinburne University of Technology, 2006.

[15] C. Henke, C. Schmoll, T. Zseby, "Empirical Evaluation of Hash Functions for PacketID Generation in Sampled Multipoint Measurements," in *Passive and Active Measurement Conference (PAM)*, 2009.

[16] T. Nguyen, G. Armitage, "A Survey of Techniques for Internet Traffic Classification using Machine Learning," *IEEE Communications Surveys & Tutorials*, vol. 10, no. 4, pp. 56–76, 2008.

[17] S. Zander, G. Armitage, "DIstributed Firewall and Flow-shaper Using Statistical Evidence (DIFFUSE)." http://caia.swin.edu.au/urp/diffuse.