Towards Trustworthy Network Measurements

Ghassan O. Karame

NEC Laboratories Europe
69115 Heidelberg, Germany
ghassan.karame@neclab.eu

Abstract. End-to-end network measurement tools are gaining increasing importance in many Internet services. These tools were designed, however, without prior security consideration which renders their extracted network estimates questionable, given the current adversarial Internet. In this paper, we highlight the major security vulnerabilities of existing end-to-end measurement tools and we sketch possible avenues to counter these threats by leveraging functionality from the OpenFlow protocol. More specifically, we show that the security of bottleneck bandwidth estimation and RTT latency measurements in network coordinate systems can be strengthened when the network deploys a number of OpenFlow-operated switches.

Keywords: Software Defined Networks, OpenFlow protocol, Security, Network Measurements.

1 Introduction

The ability to measure the network performance is an intrinsic component in the design of the current Internet. Network measurements are crucial for the operation and security of the Internet, and of several services including content distribution and peer-to-peer (P2P) systems [23]. Numerous tools for estimating network performance have been proposed (e.g., *bandwidth measurement:* Sprobe [24], *latency:* ping [21], *link quality:* mtr [3], etc.). However, the increasing dependence of current applications and services on network measurement tools is showing the limits of foresight in the design of these tools:

– **End-to-end measurements:** Current measurement tools push the measurement function to the end-hosts, and do not require functionality from intermediate network elements (e.g., switches). By doing so, they implicitly assume that end-hosts are honest and behave "correctly". However, if hosts misbehave and do not obey the measurement protocol (e.g., free-riding [23, 27]), the estimated end-to-end metric will not reflect the genuine state of the network.

– **No prior security considerations:** Current network measurement tools were developed without prior security considerations, which makes them vulnerable to a number of security threats. Since the measurements are performed end-to-end, the end-hosts might not be able to distinguish these attacks from "authentic" measurements [15, 19].

Till recently, the end-to-end principle [22] has provided a justifiable rationale for moving functions closer to the end-hosts and has shaped the way the current Internet is designed. The true leverage of the end-to-end argument was implicitly a global architecture comprising a "naive" network and "smart" applications that do not require functionality from the switching elements deployed within the network. Given this, the design of network measurements tools equally adopted the end-to-end principle, owing to the unavailability of infrastructural support for measurements and to the absence of viable alternatives. This renders the task of securing network measurements rather challenging in the current Internet.

Nowadays, the emergence of Software Defined Networks (SDNs) suggests a slight departure from the end-to-end principle. These networks separate the "control plane" and the "data plane", and thus achieve a large degree of "network virtualization". OpenFlow [5] is one such protocol that enables the construction of SDNs in practice. OpenFlow is a data link layer communication protocol that enables an *OpenFlow controller* to configure paths, in *software*, through a number of *OpenFlow-operated switches*. Here, the controller issues (routing) rules to the switches using a secure control channel; the switches can then dynamically implement the requested rules on the data plane.

In this paper, we argue that the OpenFlow protocol can strengthen the security of applications that rely on end-to-end network measurements. To that end, we start by highlighting the security vulnerabilities of existing network measurement tools. We then sketch possible avenues that leverage OpenFlow to enhance the security of bottleneck bandwidth estimation and of RTT measurements in network coordinate systems. To the best of our knowledge, the security provisions of OpenFlow-enabled networks (and the resulting division of trust between their hosts) have not been yet analyzed in the context of network measurements.

The remainder of the paper is organized as follows. Section 2 compiles a list of security threats encountered in existing measurement tools. In Section 3, we sketch possible avenues to alleviate attacks against network measurements by leveraging functionality from the OpenFlow protocol. Section 4 overviews related work, and we conclude the paper in Section 5.

2 Threats against Network Measurements

In this section, we start by outlining the major security threats against existing end-to-end network measurement tools.

2.1 Threat Model

While they might be different in purpose and technique, most *active* end-to-end measurement tools share a similar model consisting of a *verifier* and a *prover* connected by a *network*. The verifier wants to *measure* and *verify* the end-to-end performance of the path to the prover. The verifier actively generates probe packets destined to the prover, who appropriately echoes back its reply probe packets to the verifier (the prover *cooperates* with the verifier, otherwise the prover will be denied service by the verifier). The verifier then estimates the performance of the end-to-end path to the prover by extracting and analyzing the probe packets' arrival times depending on the measurement technique.

While an external attacker can *spoof* the IP [11] of the prover and issue back replies on its behalf, *untrusted provers* constitute the core of our *internal* attacker model. Untrusted provers denote those hosts involved in the measurement process, but they are not trusted by the verifier to correctly execute the measurement steps. Untrusted provers can *intentionally* manipulate the sending time of their reply probes and claim a measurement value of their choice.

2.2 Delay Attacks on Network Measurements

Most end-to-end measurement tools rely on ICMP or TCP/UDP implementations at end-hosts or at routers to exchange probe packets along a path.

While rushing attacks (where the adversary predicts the reply packets and sends them ahead of time) and impersonation attacks can be countered by relying on lightweight cryptographic primitives, delay attacks pose a serious challenge to existing network measurements tools. An untrusted prover can intentionally *delay* its reply probes to convince the verifier of a performance value of its choice. Delay attacks can result in both inflated (higher) or deflated (lower) measurement estimates [18,19]. The amount of delay that needs to be introduced depends on the probe size and on the estimation techniques in use.

To perform delay attacks, untrusted provers can "manipulate" their networking interface and introduce appropriate delays that match their desired claims. Alternatively, provers can make use of available software, such as traffic shapers (e.g., NetLimiter [4], HTB [2], etc.) to throttle their outgoing traffic according to a target rate matching their network performance claims.

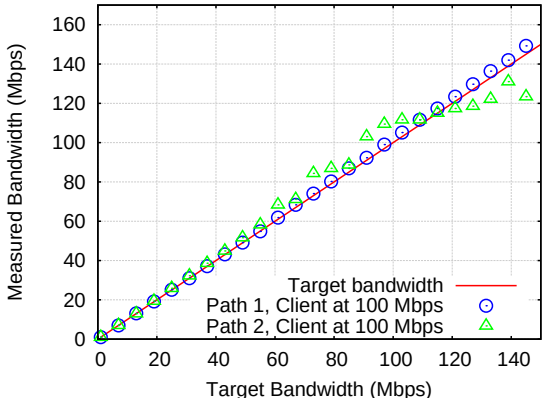


Fig. 1. Delay Attacks on Sprobe [24]. Here, we conducted our measurements on 100 Mbps symmetric physical connections deployed on two paths: **Path1** where both the verifier and the prover are located in the same state, and **Path2** where the verifier is located in Europe and the prover is located in the US.

Example—Delay Attacks on Bandwidth Estimation: The packet-pair technique is a widely adopted technique for measuring the bottleneck bandwidth (the minimum capacity) of an Internet path. To measure the download bandwidth in the packet-pair technique, the verifier sends *two* back-to-back *large* probe packets of equal size to the prover. These packets are likely to queue at the bottleneck link; their dispersion is then inversely proportional to the bottleneck bandwidth of the path [24]. The prover then issues back small reply probe-pairs; the verifier estimates the prover's download bandwidth: $B = \frac{S}{T}$, where B is the bandwidth to the prover, S is the size of the request probes, and T is the time dispersion between the reply probe-pair[1] [24]. Similarly, to measure the *upload* bandwidth, the verifier issues small request packet probes; the prover in turn replies with large reply packet probes. The upload bottleneck bandwidth is inversely proportional to the dispersion between the reply packet pairs.

Untrusted provers can claim a higher (or lower) bandwidth by introducing a delay $\Delta = S \cdot |\frac{1}{B_{claimed}} - \frac{1}{B_{auth}}|$ before (after) responding to the first request packet. Here, $B_{claimed}$ denotes the fake claimed bandwidth of the prover and B_{auth} is the genuine bandwidth of the prover [18, 19]. We implemented this attack on a popular bandwidth estimation tool based on the packet-pair technique, Sprobe [24]. Our findings are depicted in

[1] Since the reply packets are small in size, their dispersion should reflect the initial dispersion of the large request probes sent by the verifier.

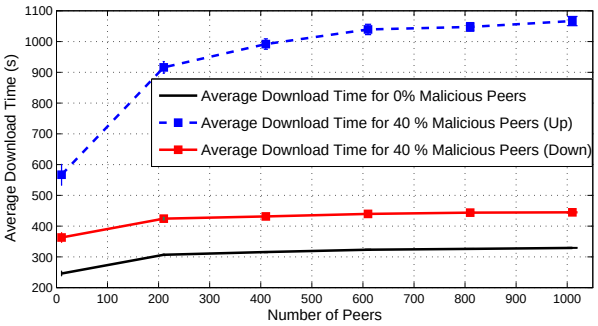


Fig. 2. Impact of bandwidth inflation/deflation attacks on a content distribution network based on multicast binary trees. Each data point is averaged over 1000 runs; where appropriate, we present the corresponding 95% confidence intervals.

Figure 1. In the figure, *target bandwidth* refers to the bandwidth that an untrusted prover claims and *measured bandwidth* denotes the bottleneck bandwidth estimate extracted by the verifier. Indeed, the prover can, by appropriately delaying its reply probes, claim a bandwidth of its choice irrespective of its physical bandwidth capability.

Implications of Attacks: We now investigate the impact of the aforementioned attack in a content distribution network (CDN) based on multicast binary trees.

We implemented a C-based simulator that simulates a content distribution network, in which a central verifier measures the bandwidths of peers prior to organizing them in a binary multicast tree. Here, *fast* peers should be located close to the multicast root in order to boost the performance of the network [25]. In our simulations, the nodes' bandwidth were chosen based on the bandwidth distribution in current P2P networks as reported in [23]. Figure 3 shows the detrimental impact of fake bandwidth claims on resource distribution; the average download times over *all* peers in the network *doubles* when 40 % of the peers claim a lower bandwidth than their own. This effect is more detrimental when those peers claim a higher bandwidth than they actually have; the average download time over all peers almost *quadruples*.

3 Trustworthy Network Measurements using OpenFlow

In what follows, we analyze the provisions of OpenFlow in strengthening the security of network measurements and their applications. We focus on *bottleneck bandwidth* estimation and on *network coordinate* measurements. Here, we consider the system model outlined in Section 2.1 and

we assume a setting where the path between the verifier and the prover traverses a network domain $\mathcal{D}$ that is governed by an OpenFlow controller C. We further assume that C *cooperates* with the verifier in order to ensure the security of the conducted measurements (e.g., cooperation between the CDN and the network operator).

3.1 Bottleneck Bandwidth Estimation

We start by outlining a scheme that leverages the OpenFlow protocol and enables the secure estimation of the upload[2] bottleneck bandwidth of the prover using the packet-pair technique (cf. Section 2.2). Here, we assume that the verifier's download bandwidth is much larger than that of the prover, otherwise, it cannot measure the upload bottleneck bandwidth of the prover.

Our solution unfolds as follows. The verifier crafts request packets that contain pseudo-randomly generated payloads and requires that the reply packet issued by the prover echo the contents of the request packets. This prevents the prover from issuing the reply packets before receiving the corresponding request packets. The verifier also requests that all measurement packets issued by the prover embed within their headers a pre-defined flag (e.g., in the ToS field). This serves to announce to the OpenFlow-operated switches on the path to the verifier that these correspond to bandwidth measurement packets. Finally, the verifier informs the controller C about the IP address of the prover.

C then propagates a rule to the outermost OpenFlow-operated switch that connects to the *prover*, requesting to *queue* all the packets that it receives from the IP addresses of the prover, whose headers contain a measurement flag. This removes any additional delay that the prover may have inserted within the transmission of its packet-pairs. This process is shown in Figure 3(a). In the OpenFlow protocol, this request is defined through a FLOW_MOD message type. For instance, a FLOW_MOD message that requests from a switch to queue packets that originate from IP address "x.y.z.w" and that have the ToS field set to "11111111" looks like:

Match set: All wildcards but (NW_DST that has value "x.y.z.w" AND NW_TOS that has value "11111111")
Action set: ENQUEUE (queue: 1, port: a)

[2] A similar analysis equally applies for download bandwidth estimation.

Here, queue 1 must be appropriately defined on the switch in question; the scheduler releases packets from this queue only when it contains at least two packets. In addition, the controller C requests that the switch forwards the packets (that are filtered in the above match set), along with their received timestamps, to C (e.g., in the control plane), who in turn sends them to the verifier. Let $disp$ denote the dispersion between the packet-pair received by the verifier, and let $\delta = (t_2 - t_1)$, where t_1 and t_2 denote the respective reception time of the packet-pair at the switch, as reported by C. Here, two cases emerge:

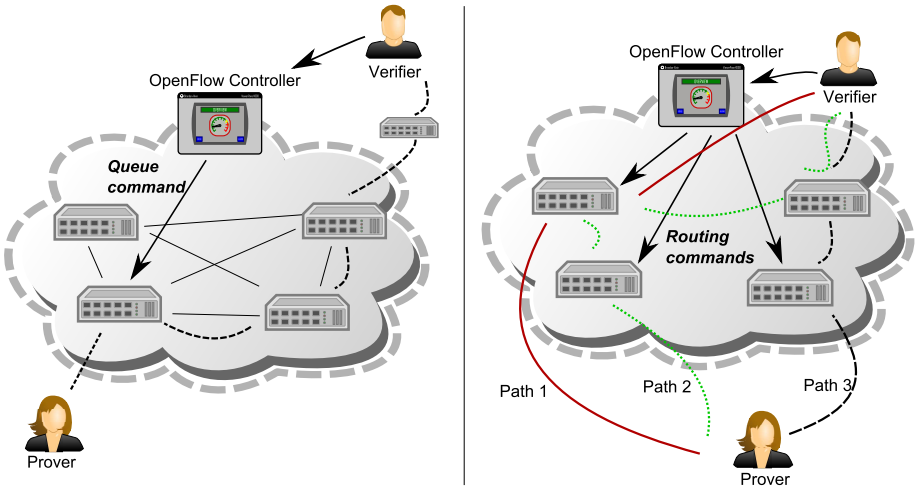
- $\delta \leq disp$. Here, the bottleneck link is located between the verifier and the OpenFlow switch. Since the latter queued the packet-pair, the verifier is certain that the bandwidth estimate is correct.
- $disp < \delta$. In this case, the bottleneck link is located on the prover's side of the OpenFlow switch and the verifier is certain that the bottleneck bandwidth B of the prover ranges between: $\frac{S}{\delta} \leq B \leq \frac{S}{disp}$, where S is the size of the reply packets of the prover.

The closer is the prover from an OpenFlow-operated switch in $\mathcal{D}$, the more accurate is the estimate acquired by the verifier. That is, the smaller is the number of hops that separate the prover from the outermost OpenFlow-operated switch, the higher is the probability that the bottleneck link is located after the switch, on the path to the verifier. This conforms with recent studies that show that bottleneck links typically co-exist within inter-domains links [12].

3.2 Network Coordinate Measurements

Network coordinate systems provide hosts with the means to easily learn their coordinates (RTT latencies) relative to other hosts in the network. In these systems, hosts compute their "coordinates" in the network by measuring their latency to other nodes [9]. In [15, 16], Kaafar *at al.* analyzed the impact of delay attacks on network coordinate systems and propose the reliance on surveyor nodes to counter these threats. In what follows, we show that the reliance on OpenFlow-enabled switches can also alleviate delay attacks on these systems. Here, we assume that the prover wants to claim a coordinate position of its choice relative to the verifier (e.g., to be placed favorably in a CDN).

The verifier measures the coordinate position of the prover as follows. The verifier crafts its echo request packets, by ensuring that their content cannot be predicted (to prevent rushing attacks) and by inserting



(a) Bottleneck bandwidth estimation (b) Network coordinate measurements

Fig. 3. Leveraging OpenFlow to strengthen the security of network measurements. The shaded area corresponds to a domain $\mathcal{D}$ that is located on the measured path.

a pre-defined flag in their headers (e.g., in the ToS field). The verifier also requires that the prover's reply packets *(i)* are correlated in content to the request packets (e.g., echoing the request packets) and *(ii)* embed the flag in their headers. Given the IP address of both the prover and the verifier, the controller C then dynamically configures a random path (across the OpenFlow-operated switches) that the packets (whose headers contain a flag) exchanged among the verifier and the prover traverse (Figure 3(b)). For example, in the OpenFlow protocol version 1.0, this is defined through the following message that is propagated to each switch in $\mathcal{D}$:

Match set: All wildcards but (NW_DST that has value "x.y.z.w" AND NW_TOS that has value "11111110") **Action set:** OUTPUT (port: a)

Here, by defining the output port of the flow, C manages the routing of the filtered packets. C also informs the verifier about the chosen path; the verifier in turn measures the RTT to the prover of the packets traversing the configured path. This process is repeated for a number of independent runs, in which different paths are configured by the controller. Note that these paths need to be chosen such that the same outermost OpenFlow-operated switch does not repeat across different runs. Since the prover

cannot predict the path that the packets will follow to the verifier, it is easy to see that the prover cannot insert the accurate amount of delays [16] in order to claim a latency/coordinate of its choice relative to the various OpenFlow-operated switches (and therefore to the verifier, since the latter knows its RTT latency relative to the switches[3]).

Clearly, the bigger is the domain $\mathcal{D}$, the larger is the number of OpenFlow-operated switches, and the harder it is for the prover to predict the path (and the corresponding required delay) to the verifier. We point out that, here, the prover does not have to be located in the proximity of $\mathcal{D}$.[4]

4 Related Work

Several tools for *active* network measurements have been proposed and evaluated empirically over a number of Internet paths [1]. Examples include Pathchar [13], and Sprobe [24] measuring the *bottleneck bandwidth*, ping [21] for measuring *network delay*, etc.. On the other hand, *passive* network measurement tools rely on monitoring existing traffic between end-hosts to extract their estimates. Several tools for passive measurements exist, such as Nettimer [20], Viznet [6], etc.. However, these tools are finding less applicability nowadays since existing traffic is not suitable for them to produce an indicative estimate [24].

In [27], Walters *et al.* propose to mitigate attacks against measurements in overlay networks by combining anomaly detection and reputation-based techniques [10, 17]. In [18, 19], Karame *et al.* investigate the vulnerabilities of bandwidth estimation techniques in adversarial settings. In [15, 16], Kaafar *et al.* analyze the security of RTT measurements in Internet coordinate systems.

The literature features a number of proposals that leverage SDNs and the OpenFlow protocol [14, 26]. However, these contributions focus on monitoring the network status, and do not address the security of network measurements.

5 Concluding Remarks

Given the current trends in designing a "clean-slate" future Internet, this paper motivates the need for a *secure* next-generation Internet. Given

[3] These act as hidden landmarks [8] to securely position the prover.

[4] In the case where the prover is located is the close proximity of an OpenFlow switch in $\mathcal{D}$, then the verifier can directly estimate the position of the prover since it knows the relative coordinate of the OpenFlow-operated switch.

the importance of network monitoring, *secure* infrastructural support for network measurements becomes rather a necessity [7]. In this respect, we showed that OpenFlow-operated networks can strengthen the security of applications that rely on end-to-end network measurements.

However, even if most switches in the network provide support for network measurements, we still expect other hazards to arise with respect to the security of the overall network. Indeed, the reliance on dedicated measurement components requires the presence of trusted authorities that control and maintain these components. Here, access control to the infrastructure is a crucial design property; any compromise might result in severe performance deterioration throughout the entire network. Overcoming these limitations requires a careful design of the measurement functions in the future Internet.

Acknowledgements

The author would like to thank Srdjan Capkun for the helpful discussions and valuable comments. The author would also like to thank Roberto Bifulco for the various discussions on the OpenFlow protocol.

References

1. "CAIDA, tools: taxonomies", http://www.caida.org/tools/taxonomy/performance.xml.
2. "HTB Traffic Shaper", http://luxik.cdi.cz/ devik/qos/htb/.
3. "mtr", http://www.bitwizard.nl/mtr/.
4. "NetLimiter", http://www.netlimiter.com/.
5. "OpenFlow–Enabling Innovation in your Network", Available from http://www.openflow.org/.
6. "Viznet", http://dast.nlanr.net/Projects/Viznet/
7. P. Barford, "Measurement as a First Class Network Citizen", http://pages.cs.wisc.edu/p̃b/sngi_whitepaper.pdf.
8. S. Capkun, K. B. Rasmussen, M. Cagalj and M. Srivastava, "Secure Location Verification With Hidden and Mobile Base Stations", In *IEEE Transactions on Mobile Computing (TMC)*, 2008.
9. F. Dabek, R. Cox, F. Kaashoek, and R. Morris, "Vivaldi: A Decentralized Network Coordinate System", In *Proceedings of SIGCOMM*, 2004.
10. T. Dimitriou, G. Karame and I. Christou, "SuperTrust A Secure and Efficient Framework for Handling Trust in Super Peer Networks", In *Proceedings of ACM PODC*, 2007.
11. B. Harris and R. Hunt, "TCP/IP security threats and attack methods", In *Computer Communications*, 1999.
12. N. Hu, L. Li, Z. m. Mao, P. Steenkiste, and J. Wang, "A Measurement Study of Internet Bottlenecks" In *Proceedings of INFOCOM*, 2005.
13. V. Jocobson, "Pathchar", http://www.caida.org/tools/utilities/others/pathchar.

14. L. Jose, M. Yu, and J. Rexford, "Online measurement of large traffic aggregates on commodity switches", In *Proceedings of Hot-ICE*, 2011.
15. M.A. Kaafar, L. Mathy, C. Barakat. K. Salamatian, T. Turletti, and W. Dabbous, "Securing Internet Coordinate Embedding Systems", In *Proceedings of ACM SIGCOMM*, 2007.
16. M.A. Kaafar, L. Mathy, T. Turletti, and W. Dabbous, "Virtual Networks under Attack: Disrupting Internet Coordinate Systems", In *Proceedings of CoNext*, 2006.
17. G. Karame, I. Christou, and T. Dimitriou, "A Secure Hybrid Reputation Management System for Super-Peer Networks", In *Proceedings of IEEE CCNC*, 2008.
18. G. Karame, D. Gubler, and S. Capkun, "On the Security of Bottleneck Bandwidth Estimation Techniques", In *Proceedings of SecureComm*, 2009.
19. G. Karame, B. Danev, C. Bannwart, and S. Capkun, "On the Security of End-to-End Measurements based on Packet-Pair Dispersions", In *IEEE Transactions on Information Forensics & Security (TIFS)*, 2013.
20. K. Lai and M. Baker, "Nettimer: A Tool for Measuring Bottleneck Link Bandwidth", In *Proceedings of USITS*, 2001.
21. M. Muuss, "ping", ftp://ftp.arl.mil/pub/ping.shar.
22. J. H. Saltzer, D. P. Reed and D. D. Clark, "End-to-End Arguments in System Design", In *ACM Transactions on Computer Systems*, 1984.
23. S. Saroiu, P. Gummadi, S. Gribble, "A Measurement Study of Peer-to-Peer File Sharing Systems", In *Proceedings of MMCN* 2002.
24. S. Sariou, P. Gummadi and S. Gribble, "SProbe: A Fast Technique for Measuring Bottleneck Bandwidth in Uncooperative Environments", In *INFOCOM*, 2002.
25. M. Schiely, L. Renfer and P. Felber, "Self-Organization in Cooperative Content Distribution Networks", In *Proceedings of NCA*, 2005.
26. A. Tootoonchian, M. Ghobadi, and Y. Ganjali, "OpenTM: traffic matrix estimator for OpenFlow networks", In *Proceedings of PAM*, 2010.
27. A. Walters, D. Zage and C. Nita-Rotaru, "A Framework for Mitigating Attacks Against Measurement-Based Adaptation Mechanisms in Unstructured Multicast Overlay Networks", In *ACM/IEEE Transactions on Networking*, 2007.