Report

Towards Secure End-to-End Network Measurements

Author(s):
Karame, Ghassan O.; Capkun, Srdjan

Publication Date:
2009

Permanent Link:

https://doi.org/10.3929/ethz-a-006814991 →

Towards Secure End-to-End Network Measurements

Ghassan O. Karame and Srdjan Čapkun

Abstract

End-to-end network measurement tools are gaining increasing importance in many Internet services. This is the case since the performance and Quality-of-Service (QoS) of many online applications and services largely depend on their network characteristics. These measurement tools were designed, however, without prior security consideration which, given the current adversarial Internet, renders their extracted network estimates questionable. In this article, we highlight the major security vulnerabilities of existing end-to-end measurement tools and we sketch possible research directions to counter these threats. We further argue that finding "full-fledged" solutions for these challenges might require a serious re-design of the current Internet.

I. Introduction

Network measurements are becoming crucial for the operation and security of the Internet, and of several services including application-level multicast trees, content distribution and peer-to-peer (P2P) systems [1]. Numerous tools for estimating network performance have been proposed (e.g., *bandwidth measurement:* Sprobe [4], *latency:* traceroute, ping, *link quality:* mtr, etc.). Most of these tools push the measurement function to the end-hosts, thus conforming to the well-known end-to-end principle [3] that suggests the move of functions closer to the applications and to the end-hosts.

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:

– **No prior security considerations:** Current network measurement tools were developed without prior security considerations, which makes them vulnerable to *external* and *internal* attacks ranging from IP spoofing to delay and rushing attacks [7]. Since the measurements are performed end-to-end, the end-hosts might not be able to distinguish these attacks from "authentic" measurements. These security vulnerabilities might also affect the operation of the applications that make use of these measurement tools (e.g., Internet coordinate systems [2], multicast trees), thus increasing the gain of the attacker.

– **Implicit trust assumptions:** Current network measurement tools were designed to work in a trusted environment and implicitly assume that both end-hosts are honest and behave "correctly". However, in many situations, the interests of end-hosts might differ considerably and thus, they might have strong incentives to cheat and increase their advantage in the network [1], [6] (e.g., free-riding). In fact, if the endpoints misbehave and do not obey the measurement protocol, the estimated end-to-end metric will not reflect the authentic state of the network. To the best of our knowledge, the division of trust between end-hosts has not been yet investigated in the context of network measurements.

This article examines the security vulnerabilities of existing end-to-end network measurement tools, presents a list of design goals that *secure* measurement tools should satisfy to cope with adversarial behavior in the Internet and highlights possible research solutions in this area. Namely, we argue that

G. Karame and S. Čapkun are affiliated with the Department of Computer Science, ETH Zurich, 8092 Switzerland.
E-mail: {krameg,capkuns}@inf.ethz.ch

finding "clean-slate" solutions to secure network measurements might require re-designing the architecture of the current Internet. Our findings therefore motivate the need for a *secure* next-generation Internet that considers secure network measurements as "first-class network citizens" [9]. In this respect, we analyze the prospects of relying on a trusted network infrastructure to securely evaluate the network performance.

This article is structured as follows. Section II presents a brief outline of existing measurements tools. Section III compiles a list of security threats encountered in current tools. Section IV suggests a list of desirable properties that secure network measurement tools should satisfy. In Section V, we sketch possible avenues to alleviate attacks against existing tools. In Section VI, we extract relevant lessons for the future and we conclude in Section VII.

II. End-to-End Measurement Tools

Till recently, the end-to-end principle [3] has justified the rationale of moving functions closer to the end-hosts that use the function and has shaped the way in which the current Internet is designed. 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 due to the absence of viable alternatives. Unfortunately, the design of current tools is revealing the limits of foresight; indeed, while the original end-to-end principle assumed a setting where both end-points are honest and behave correctly, today, users might have strong incentives to cheat [1] in order to increase their benefit in the network. Furthermore, given the spread of malware, end-hosts are highly likely to be compromised when compared to intermediate network components.

A. Passive Network Measurements

Passive network measurement tools rely on monitoring existing traffic between end-hosts to extract their estimates. These tools are scalable since they do not generate extra traffic for measurement purposes. Most current tools use packet capture libraries to monitor traffic and to extract various metrics of interest. Several tools for passive measurements exist, such as Nettimer (for bandwidth estimation), Viznet (for throughput tests), sting (for latency tests), etc..

However, passive measurement tools are finding less applicability nowadays due to the fact that existing traffic is not always suitable to produce an indicative estimate [4].

B. Active Network Measurements

A large number of tools *actively* generate probe traffic to extract their estimates about the network state. Most of these tools are single-ended and do not require cooperation from the measured host. Several tools for active network measurements have been proposed; the Cooperative Association for Internet Data Analysis (CAIDA) hosts a comprehensive list of these tools along with their measurement functions. Examples include Pathchar, and Sprobe [4] measuring the *bottleneck bandwidth*, Cprobe and Spruce measuring the *available bandwidth*. Several other tools aim at measuring *the roundtrip times (RTT), latency, delay and the throughput* of an end-to-end path, such as traceroute, mtr, ping, etc.. A detailed description of the operation of these tools is out of the scope of this article.

System Model: While they might be different in purpose and technique, most *active* end-to-end measurement tools share a similar system model consisting of a *verifier* and a *prover* connected by a *network* (Figure 1). 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

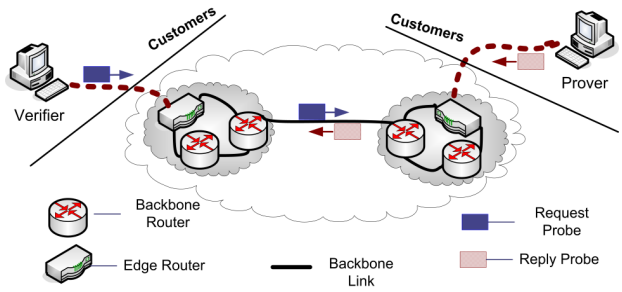

Fig. 1. *Active* network measurements: probes are exchanged between the verifier and the prover. The former then analyzes the probes arrival/departure times.

back its reply probe packets to 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 in question. For instance, Sprobe [4] makes use of TCP SYN and RST packets to force the remote-host to send packet-pairs replies. The verifier then estimates the bottleneck bandwidth by computing the time dispersion between the reply packet-pairs.

In this article, we are not concerned with the detailed operation of current measurement tools; instead, we abstract away their distinct features and we focus on highlighting the vulnerabilities of the general end-to-end measurement model in adversarial settings. Given that the basic differences between these tools reside in the way they extract their estimates from the departure/arrival times of the exchanged probe packets, the model described above applies to the multitude of active end-to-end measurement tools and is not constrained by the peculiarities of a particular estimation technique. Due to lack of space, we focus on the security vulnerabilities of active measurement techniques and where appropriate we will point to the similarities and/or differences to passive measurement techniques.

III. Security Threats against End-to-End Network Measurements

In this section, we outline the major security threats against active end-to-end network measurement tools.

A. Attacker Model

Untrusted provers constitute the core of our *internal* attacker model; untrusted provers are 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.

Note that these attacks can also be performed by external attackers that compromise routers on the path between the prover and the verifier to alter the measurements.

B. Exemplary Threats

In what follows, we briefly describe common security threats against existing end-to-end measurement tools.

a) Delay Attacks: 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. The amount of delay Δ that needs to be introduced depends on the probe size and on the estimation techniques in use. Note that this attack is not particular to end-hosts nor to active measurements; a *compromised router* can equally trick the verifier into accepting fake claims by introducing appropriate delays to the probes' traversal times.

To perform delay attacks, untrusted provers can "manipulate" their networking interface and introduce appropriate delays that match their desired claims (Section III-C). Alternatively, provers can make use of available software, such as traffic shapers (e.g., NetLimiter, HTB, etc.) to throttle their outgoing traffic according to a target rate matching their network performance claims. Delay attacks can be very hard to detect given a small unnoticeable delay. In an experiment we conducted, we were able to claim a 10 Gbps download bandwidth using the Pathchar bandwidth estimation tool, over a 100 Mbps physical download connection, by introducing a maximum delay of 120 μs.

b) Rushing Attacks: An untrusted prover can also predict the *Identifier* and *Sequence Number*[1] fields in the request probe packets and send *specially crafted* reply probes ahead of time. In this way, an attacker can claim a smaller RTT which translates into a different measurement estimate.

c) Impersonation Attacks: By relying on ICMP, TCP or UDP-based probe packets, current measurement tools do not provide any form of source or destination authentication. An external attacker can *spoof* the IP of the prover and issue probe replies, as if it was the actual prover; the resulting metric measured by the verifier would thus be that of the attacker. This attack is further aggravated if the attacker re-routes probe packets to multiple heterogeneous workstations at its disposal to claim a performance value of her choice. Impersonation attacks can be equally mounted against passive network measurement tools; here, an attacker can generate traffic on behalf of a host and thus alter the network measurements.

C. Example: Delay Attacks on Bandwidth Estimation

To demonstrate the feasibility of delay attacks, we conducted sample experiments in which untrusted provers fake their bandwidth claims and we simulated the impact of these attacks on a typical tree-based distribution application.

Since the performance of resource distribution largely depends on the organization of peers in the distribution tree (*slow* peers are the leaves of the tree and *fast* peers are located close to the multicast root), we assume that a central high-speed verifier measures the bandwidths of network hosts using the Sprobe [4] bandwidth estimation tool to optimally place them in the tree. Sprobe is a popular tool based on the packet-pair technique [5], where the verifier sends *two* back-to-back *large* TCP SYN request probe packets of equal size to the provers. The provers then issue back their TCP RST reply probe-pairs; the verifier estimates the provers' bandwidths as follows: $B = \frac{S}{T}$, 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.

Untrusted provers can claim a *higher* bandwidth by introducing a delay $\Delta = S \cdot (\frac{1}{B_{claimed}} - \frac{1}{B_{auth}})$ before 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. By claiming higher bandwidths, provers are likely to be placed close to the content multicast root, and are therefore likely to acquire the resource faster. Alternatively, provers might claim *lower* bandwidths, to reduce their upload contribution in the network. Provers can claim lower bandwidths by delaying their reply to the second request probe issued

[1]Generally, the *Sequence Number* field in the probe requests is incremental and therefore can be easily predicted.

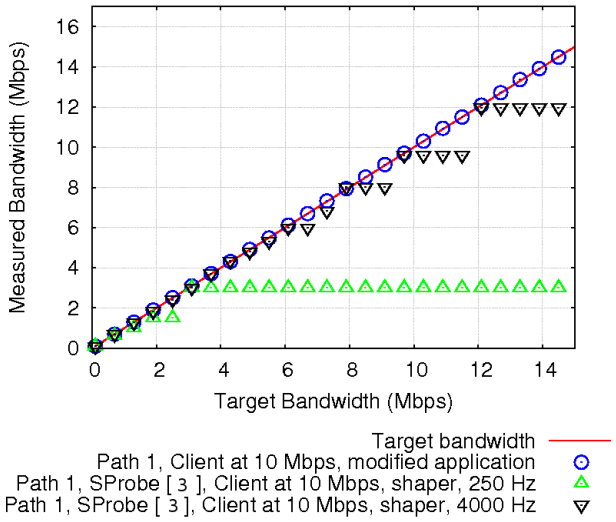

Fig. 2. Delay Attacks on Sprobe [4]. Here, the client is connected by a 10 Mbps link to the verifier.

by the verifier. These attacks can be equally performed by a *rogue* router located on the path between the end-hosts.

To introduce the aforementioned delays, we relied on *(i)* an open source Linux traffic shaper (the HTB shaper) and *(ii)* on a application that modifies the networking interface of end-hosts. Our application replaces the kernel's TCP/IP stack by a raw socket and uses an iptable rule to drop all replies issued by the kernel; the prover then sends back the TCP SYN reply probes with the desired delay. We conducted our measurements on 10 Mbps symmetric connections where the verifier is located in Switzerland and the prover is located in Germany; we denote this path by *Path1*. Our findings are depicted in Figure 2. 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. Our results show that, by relying on a bandwidth shaper, or on a modified networking interface, the prover can claim a bandwidth of its choice irrespective of its physical bandwidth capability [7].

Implications of Attacks: In a prototype simulation we have conducted, we studied the impact of malicious peers on the resource distribution times in traditional binary trees where one host wishes to distribute a 5 MB file comprised of 20 fragments of size 256 KB each to 1000 leechers. The bandwidth distribution of the studied nodes was derived from the findings in [1]. In our simulation, selfish hosts inflate or deflate their reported bandwidths by a factor ranging from 2 to 10 times. Our results show that selfish peers can considerably increase resource download times by intentionally misreporting their upload bandwidth. Namely, the average download times over *all* peers doubles when 40 % of the leechers claim to have a lower upload bandwidth. This effect is even more detrimental when those peers claim a higher upload bandwidth than they actually have; the average download time of all the fragments over all peers almost quadruples.

IV. Design Goals for Secure Network Measurement Tools

In this section, we outline desirable properties that a *secure* end-to-end network measurement tool should satisfy to be widely deployed in the current Internet.

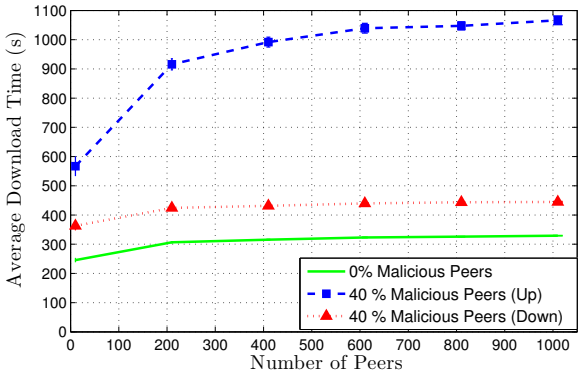

Fig. 3. Impact of attacks on the peers' average download times in a typical multicast binary trees. In this example, one host wishes to distribute a 5 MB file comprised of 20 fragments of size 256 KB each to 1000 leechers. Each data point in the plot is averaged over 1000 runs.

A. Active and Cooperative Measurements

Some previous work [4] argues that network measurement tools should be designed to work in uncooperative environments to scale to a large number of Internet hosts without the need for dedicated software on the prover's side. Although this is indeed a desirable property, we find support for uncooperative environments rather unrealistic. In fact, with the increasing proliferation of "de-facto" security applications, such as home firewalls, uncooperative probing techniques find less applicability in the near future as they are likely to be considered hostile by end-hosts. Some routers already *filter* ICMP packets due to their potential malevolent use [5]. The support for cooperative measurement environments seems therefore inevitable in the near-future. Furthermore, end-to-end security would impair the use of measurement tools in passive and uncooperative environments as it requires end-host cooperation in achieving essential security primitives (e.g., source authentication). Unless functionality is pushed to the Internet backbone (Section VI), cooperative environments present themselves as vital "playgrounds" for secure end-to-end network measurements in the current Internet.

B. Resilience to External and Internal Attackers

Current end-to-end network measurement techniques push trust towards end-hosts; an untrusted prover can abuse this trust in numerous ways and claim a performance capability that increases its advantage. Secure end-to-end measurement tools should *detect* those attackers that are not correctly executing the measurement phases. Furthermore, secure end-to-end measurement tools should be resilient to *external attacks* on the measurement process.

C. Efficiency and Accuracy

Secure end-to-end measurement tools should preserve the accuracy of the measurements and cope with inherent network properties such as cross-traffic and traffic dynamism. We further argue that the whole process should be as efficient as possible. This includes the overhead in authenticating the exchanged probes, and detecting possible host misbehavior.

V. Possible Countermeasures

In what follows, we sketch potential solutions to mitigate attacks against end-to-end network measurement tools.

A. Countering Impersonation and Rushing Attacks

Network measurement tools can use standard authentication protocols (e.g., lightweight symmetric cryptography) to counter impersonation attacks. By authenticating the exchanged probes (using pre-shared keys for example), an external attacker will not be able to impersonate the prover by issuing probe packets on its behalf since it cannot forge the prover's authentication on the verifier's request probes. Previous work suggested that the overhead incurred by lightweight authentication does not affect the accuracy of the measurements.

To counter rushing attacks, the verifier can also use pseudo-random functions to generate its request probes such that they cannot be predicted by the provers and require that the reply probes issued by the provers are correlated in content to its request probes.

B. Countering Delay Attacks

Although few schemes were proposed in the scope of securing RTT measurements [8], they assume a context where the prover needs to attest that it is within the proximity of the verifier; the prover, therefore, does not have incentives to mount delay-based attacks in this setting. This is not the case in current Internet applications. In what follows, we highlight "best-effort" avenues to address the challenges posed by delay attacks.

Analyzing "Reference" Data: In theory, delay attacks can be alleviated if the verifier knows an estimate of the RTT to the prover. The verifier can acquire RTT references through offline measurements or from online servers that perform RTT measurements around the globe. However, given the variability of RTTs in the Internet and the impact of small delays on the measurement process, this technique is unlikely to prevent untrusted provers from claiming a fake measurement value. Nevertheless, we argue that it *limits* the range of fake claims that the provers are able to make.

Outlier Detection: One viable method to filter inauthentic measurements is to acquire relevant information about the prover's host properties, such as its operating system, type of networking interface, location, etc. The literature includes several proposals for remote device fingerprinting and path fingerprinting [10]. Using such information, the verifier might detect fake claims by untrusted provers. For example, if the verifier knows that the prover is equipped with an 11 Mbps wireless card, then it is unlikely that it has a 50 Mbps download bandwidth.

Similarly, the verifier can deduce the prover's location and its type of Internet connection from its IP address; if the IP of the prover corresponds to an IP in the same state where the verifier is located, then it is unprobable that its RTT to the verifier is 300 ms. In addition, by resolving a prover's IP address into its Domain Name Server (DNS), the verifier can detect discrepancies in the measurements; for instance, if the verifier measures a 5 Mbps download bandwidth while the prover's DNS name is "user.dialup.com", then it is highly probable that there was an attack on the measurements. Recent results show that 34% of the IPs leak information about their hosts' bandwidths and/or locations [7].

Furthermore, statistical outlier detection [6] can also be used to prevent untrusted hosts from tampering with the measurement process. For instance, a dedicated server can keep *history* of the recorded measurements. Using statistical outlier detection methods, fake claims can be detected and malicious hosts can be identified.

Detecting Traffic and Bandwidth Shapers: Traffic shapers (e.g., NetLimiter) present themselves as effortless and powerful routines to conduct delay attacks on end-to-end network measurement tools. Traffic shapers limit the amount of data a host transmits and/or accepts by delaying incoming and/or

	Resilience to Inflation Attacks	Resilience to Deflation Attacks
Round Trip Time (RTT)	✓	×
Upload Bottleneck Bandwidth	✓	⋆
Download Bottleneck Bandwidth	⋆	⋆
Upload Bottleneck Bandwidth	✓	⋆
Download Available Bandwidth	✓	⋆
Upload Available Bandwidth	✓	⋆

TABLE I
RESILIENCE OF SOME END-TO-END MEASUREMENT TOOLS AGAINST DELAY-BASED ATTACKS. ⋆ REFERS TO A PROPERTY THAT IS PARTIALLY ACHIEVED.

outgoing packets to match a specified rate input by users. However, the distribution of traffic originating from these shapers is gaussian and exhibits special properties (as shown in Figure 2, the rate at which a small burst of packets can be shaped is dependent on the kernel's system timer frequency). This enables a remote verifier to detect their deployment at the prover's side with high probability [7].

Tamper-Resistant Network Interface: As far as we aware, it is very hard, if not impossible, for the verifier to remotely detect tampering with the networking interface of the provers. One solution that addresses this problem is to embed networking tamper-resistant software or hardware on the provers' side to prevent them from tampering with their network interface [7]. Unfortunately, tamper-proof software/hardware come at high implementation costs nowadays.

VI. TOWARDS TRUSTED NETWORK INFRASTRUCTURE

In the previous paragraphs, we highlighted exemplary attacks on network measurement tools and we suggested "best-effort" techniques to mitigate these attacks. Our findings are summarized in Table I. However, these techniques fail to fully eliminate the multitude of potential threats against existing end-to-end measurement tools. Namely, mitigating delay attacks on measurements remains, to a large extent, an open research problem. We believe that the easy realization of these attacks renders the estimates of current tools highly questionable. Surprisingly, while the evolution of the Internet stimulated a serious re-interpretation of the end-to-end principle, the design of end-to-end measurement tools remains unaddressed and lacks basic security primitives.

Given the current trends in designing a "clean-slate" future Internet, our findings motivate the need for a *secure* next-generation Internet. We argue that any next-generation Internet design must provide robust functionality to support secure network measurements [9]. Since network measurements are gaining paramount importance in monitoring the performance of the Internet, *secure* infrastructural support for network measurements becomes rather a necessity. We can identify a number of reasons why it might be beneficial to embed parts of the measurement functions "inside the network": *1)* by performing measurements at an intermediate point, end-users can avoid the cost and overhead of generating unwelcome traffic across the network, *2)* by pushing functionality from end-hosts to *dedicated* and *trusted* network components, several security threats can be eliminated. For instance, edge routers could securely timestamp incoming and outgoing probes and identify whether queuing has occurred [7]. This would facilitate the detection of delay attacks. Performance "awareness" is another desirable design property for next-generation Internet. Dedicated network components could in the future construct and store bandwidth and latency maps of Internet hosts by monitoring incoming and outgoing traffic.

However, even if these challenges are overcome through an infrastructure that supports network measurements, we still expect other hazards to arise with respect to the design of future measurement components. Indeed, since the Internet traffic pattern is becoming increasingly dynamic, the underlying measurement infrastructure should be flexible to accommodate for such dynamism without the need of frequent re-design [9]. Furthermore, future network components should embed additional functionality, e.g., to measure, store, timestamp and sign packets. This can only come at the expense of increased storage, size and cost.

Delicate access control to the infrastructure is another crucial design property that needs to be considered in order to address the privacy and integrity of the measurements and the hosts. That is, the use of dedicated measurement components requires the presence of trusted authorities that control and maintain these components; any compromise of the measurement components might result in severe performance deterioration of the entire network. Overcoming these limitations requires a very careful design of the measurement functions and architecture of any next-generation Internet.

VII. Conclusion

In this article, we highlighted the major security vulnerabilities in existing end-to-end measurement tools and we sketched possible avenues to counter these threats. Another aim of this article is to increase the awareness in the community of the design issues for these tools. Due to the lack of foresight in their design, it is very hard, if not impossible, to fully counter all security challenges against existing tools. More specifically, delay attacks pose serious challenges to the consistency of network measurements as they constitute simple, yet powerful, means for attackers to abuse the operation of existing tools and maximize their profit in the network. Full-fledged countermeasures might require the design of a "clean-slate" future Internet that provides trusted infrastructure for secure network measurements.

References

[1] S. Saroiu, P. Gummadi, S. Gribble, "A Measurement Study of Peer-to-Peer File Sharing Systems", In *Proceedings of MMCN* 2002.

[2] M. A. Kaafar, K. Salamatian, L. Mathy, T. Turletti, C. Barakat and W. Dabbous, "Securing Internet Coordinate Embedding Systems", In *Proceedings of ACM SIGCOMM*, 2007.

[3] J. H. Saltzer, D. P. Reed and D. D. Clark, "End-to-End Arguments in System Design", In *ACM Transactions on Computer Systems*, 1984.

[4] S. Sariou, P. Gummadi and S. Gribble, "SProbe: A Fast Technique for Measuring Bottleneck Bandwidth in Uncooperative Environments", In *Proceedings of IEEE INFOCOM*, 2002.

[5] K. Lai, M. Baker, "Measuring Link Bandwidths using a Deterministic Model of Packet Delays", In *Proceedings of ACM SIGCOMM*, 2000.

[6] 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.

[7] G. Karame, D. Gubler and S. Capkun, "On the Security of Bottleneck Bandwidth Estimation Techniques", In *Proceedings of SecureComm*, 2009.

[8] S. Brands and D. Chaum, "Distance-Bounding Protocols", In *Proceedings of EUROCRYPT*, 1994.

[9] P. Barford, "Measurement as a First Class Network Citizen", *White Paper*, 2005.

[10] R. Sinha, C. Papadopoulos and J. Heidemann, "Fingerprinting Internet Paths using Packet Pair Dispersion", 2006.