On the Security of Bottleneck Bandwidth Estimation Techniques

Ghassan Karame, David Gubler, and Srdjan Čapkun

Department of Computer Science
ETH Zürich, Switzerland
karameg@inf.ethz.ch, dgubler@student.ethz.ch, capkuns@inf.ethz.ch

Abstract. Several wide-area services are increasingly relying on bottleneck bandwidth estimation tools to enhance their network performance. Selfish hosts have, therefore, considerable incentives to fake their bandwidths in order to increase their benefit in the network. In this paper, we address this problem and we investigate the vulnerabilities of current bottleneck bandwidth estimation techniques in adversarial settings. We show that finding "full-fledged" solutions for the multitude of attacks on the end-to-end bandwidth estimation process might not be feasible in the absence of trusted network components; we discuss solutions that make use of such trusted components. Nevertheless, we discuss other possible solutions that alleviate these threats without requiring trusted infrastructure support and we evaluate the effectiveness of our proposals on PlanetLab nodes.

Keywords: Security, Bandwidth Estimation, Bandwidth Shapers.

1 Introduction

Bottleneck bandwidth measurements are gaining increasing importance in many wide-area Internet systems and services including multicast trees [1], content distribution and peer-to-peer (P2P) systems [3]. Bottleneck bandwidth refers to the maximum throughput that a path can provide to a flow, when there is no other competing traffic load. Recently, bottleneck bandwidth estimation has attracted significant interest in the literature. This is mainly due to the fact that the performance and Quality-of-Service of most Internet services are based on their bandwidth capacities.

Several tools for bottleneck bandwidth estimation (e.g., Nettimer [4], Pathchar [5], pchar [6], bprobe [8], pathrate [9], Sprobe [10], etc.) have been proposed and evaluated both by simulations and empirically over a number of Internet paths. These techniques can be mainly classified in two categories [11]: the one-packet and the packet-pair technique. Both techniques are well understood and can provide accurate estimates under certain conditions. In both techniques, probe packets are exchanged between the *verifier* (or the sender) and the *prover* (or the receiver) to extract estimates of the network bandwidth characteristics.

To measure bandwidth in a scalable way, current bandwidth measurement tools push the estimation functionality to the end-hosts. This renders them vulnerable to a wide range of security threats as *trust* is pushed to end-hosts that are more likely to be compromised than core/edge routers. Due to the increasing reliance on bandwidth estimation in current Internet services, untrusted hosts have considerable incentives to abuse this trust and fake their bottleneck bandwidth claims in order to increase their advantage from these services (e.g., free-riding in P2P networks [3]). Indeed, current measurement techniques are often at odds with "security" when deployed in adversarial settings. A malicious host can abuse the operation of these techniques in numerous ways to claim an inflated and/or deflated bandwidth: an untrusted host can make use of bandwidth shapers or can delay its probe packets to claim any bandwidth of its choice. By inflating its bandwidth claims, an untrusted host is likely to be delegated high priority in the network. For example, the untrusted host can be chosen as a super-peer in a P2P network [12] or a recommended server in content distribution networks based on the highest-capacity path. Similarly, untrusted provers might claim lower bandwidths to reduce their contribution in the network.

While some proposals (e.g., [10], [42], etc.) recommend the deployment of bottleneck bandwidth estimation tools across Internet hosts, we argue that the easy and accurate realization of attacks against current bottleneck bandwidth estimation techniques raises serious concerns about the suitability of their deployment. A thorough evaluation of these techniques in adversarial settings should therefore precede any prospective large-scale deployment.

Previous work [10], [11], [13], [14], [16], [17], [18] focused on evaluating the performance of bandwidth estimation techniques and did not address their security vulnerabilities. In this paper, we address this problem and we analyze the major security threats against current bottleneck bandwidth estimation techniques. We also investigate the impact of available software – such as traffic shapers – on the bandwidth estimation process. We demonstrate the effect, feasibility and the accuracy of these attacks on PlanetLab nodes [43]. Another important aim of this work is to extract relevant lessons about the security prospects of existing bottleneck estimation techniques and to hint application designers on the choice of a bandwidth estimation technique that better satisfies their desired level of assurance in the measurements. To the best of our knowledge, this is the first work that investigates the security vulnerabilities of bandwidth measurements in adversarial settings.

Our findings suggest that "full-fledged" solutions against the multitude of attacks on the bandwidth estimation process might not be feasible without requiring functionality from trusted network components; namely, since measurements are conducted end-to-end, fully mitigating delay-attacks against bandwidth estimation emerges as a challenging research problem. Remote attestation by trusted network components represents one of the few viable options to prevent such attacks. In this work, we discuss the viability and the effectiveness of this proposal in securing bandwidth measurements. We further propose and analyze several other solutions and heuristics that do not require any infrastructural support and we demonstrate that these schemes counter a large subset of attacks on current bandwidth estimation techniques.

The rest of the paper is structured as follows: Section 2 briefly overviews current bandwidth estimation techniques. Section 3 compiles the list of security threats against bottleneck bandwidth estimation techniques. In Section 4, we briefly discuss a solution to thwart these attacks based on remote attestation by trusted network components. In Section 5, we propose a set of techniques that do not require infrastructural support and we evaluate their effectiveness on PlanetLab nodes. In Section 6, we discuss possible insights in the design space of secure bottleneck bandwidth measurements. We conclude the paper in Section 7.

2 Bottleneck Bandwidth Estimation

The bottleneck bandwidth B_{min} of a path is the maximum rate that the path can provide to a flow from the source to the sink. B_{min} is determined by the *minimum* link capacity in the path. In what follows, we outline the operation of the two major bottleneck bandwidth estimation techniques: the one-packet technique and the packet-pair technique.

The One-Packet Technique. The one-packet technique relies on the assumption that a packet's traversal time across a path can be computed as the sum of its transmission and propagation delays, as follows:

$$t_l^j = t_0^j + \sum_{i=0}^{l-1}\left(\frac{S_j}{B_i} + d_i\right), \tag{1}$$

where t_l^j is the traversal time of packet j through l links, t_0^j is the sending time of packet j, S_j is the packet size, B_i is the bandwidth of link i and d_i is the latency of link i.

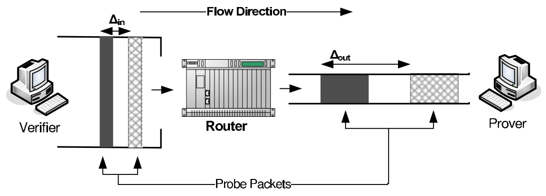


Fig. 1. Packet-Pair technique: The temporal spacing between the packets after the bottleneck link is inversely proportional to the bandwidth. The narrow part of the pipe represents the bottleneck link.

Assuming that the transmission delay is linear with respect to the packet size, it is highly likely that if the verifier transmits a large number of packets of *variable size*, at least one will have negligible queuing delay, and therefore the minimum round-trip time (RTT) values of these packets will form a line whose slope is the inverse of the link bandwidth to the prover [11]. This technique produces an estimate of the bandwidth at each hop in the path; the bottleneck bandwidth is then computed as the minimum value of the estimated link bandwidths. Note that the one-packet technique can only measure the download bandwidth (i.e., from the verifier to the prover). Examples of tools using the one-packet technique are Pathchar [5] and Clink [7].

The Packet-Pair Technique: Here, the verifier sends *two* back-to-back *large* packets of equal size to the prover. Once the prover receives these packets, it issues back its reply packet-pairs; the verifier then estimates the prover's *download* bandwidth by measuring the time dispersion between the reply packet-pairs [10]. Similarly, to estimate the prover's *upload* bandwidth, the prover sends two *large* packets adjacently in time to the verifier. The intuition behind the packet-pair technique is that when two large packets of the same size are sent back-to-back, it is highly likely that their queuing occurs at the bottleneck link of capacity B_i. Once the bottleneck link is traversed, the temporal spacing Δ_{out} between the two packets remains constant (Figure 1) and is inversely proportional to the bottleneck bandwidth [11]. Assuming FIFO queuing, the dispersion Δ_{max} after the packet-pair traverse H hops is as follows:

$$\Delta_{max} = \max_{i=0 \ldots H} \left(\frac{S}{B_i}\right) = \frac{S}{\min_{i=0 \ldots H}(B_i)} = \frac{S}{B}, \tag{2}$$

where B_i is the bandwidth of link i, S is the packet size and B is the bottleneck bandwidth of the path.

Several implementations of the packet-pair technique exist such as Nettimer [4], Pathrate [9] and Sprobe [10].

2.1 The Need for Secure Measurements

Bottleneck bandwidth measurements have the potential to solve considerable problems in applications and areas such as network management, end-to-end admission control, routing and traffic engineering [1], P2P networks, content distribution architectures [2], etc.. Selfish hosts might, therefore, have considerable incentives to fake their bandwidth claims and increase their profit from these applications; by claiming higher bandwidths, selfish hosts are likely to be assigned higher priority in the network. Alternatively, hosts might claim lower bandwidths to limit their contribution in the network. This renders "secure" bandwidth measurement a crucial task nowadays.

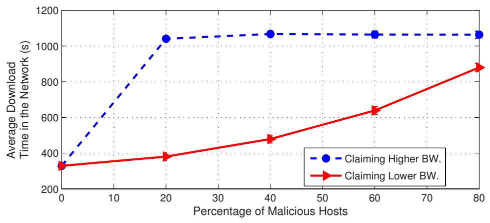


Fig. 2. Effect of malicious hosts on the average download time in a multicast binary tree application. Here, the num. of hosts is 1000 and the resource size is 3 MB. Each data point is averaged over 100 runs.

For instance, in multicast distribution architectures, the download performance of hosts is highly affected by the organization of the nodes in the tree;

one slow peer located near the root of the tree can significantly impact the resource distribution time in the network [2]. In a prototype simulation that we have conducted[1], we investigate the effect of selfish hosts faking their bandwidths in an exemplary multicast binary-tree architecture. We assume a realistic bandwidth distribution amongst the nodes derived from the findings in [3]. As shown in Figure 2, selfish hosts can considerably affect the average resource download times in the entire network by claiming incorrect bandwidths. This effect is even more detrimental when hosts claim higher bandwidths than they actually have; the average download time over all peers in the network almost *quadruples* when only 20% of peers over-report their bandwidths.

3 Bandwidth Manipulation Attacks

In this section, we investigate delay-based attacks along with the major security threats against current bottleneck bandwidth estimation techniques.

3.1 System and Attacker Model

Our system consists of a *verifier* and a *prover*, connected by a *network*. The verifier *measures* and *verifies* the bottleneck bandwidth of the path to an untrusted prover. Here, we assume that the verifier actively probes the prover by issuing probe packets. The prover echoes its reply probe packets to the verifier. The latter estimates the bandwidth of the prover by extracting packet arrival times according to either the one-packet or the packet-pair technique. We focus on *bottleneck* bandwidth measurement and we assume that the application making use of the bandwidth measurement requires that the prover *cooperates* with the verifier during this process (otherwise it would be difficult to securely estimate its bandwidth). We limit our analysis to those applications that require an accurate estimate of the bottleneck bandwidth to the prover for their correct operation. For instance, while bandwidth manipulation attacks can be tolerated in BitTorrent [15], such attacks might affect the performance of the entire network in routing services, content distribution networks, multicast architectures, etc..

We further assume that the verifier uses a high-speed connection; therefore, its bandwidth will not affect the bottleneck bandwidth of the path to the prover. We do acknowledge that current bandwidth estimation tools can result in rather large estimation errors, however we assume that *enough* probe packets are exchanged to abstract away the effects of noisy measurements.

Untrusted provers constitute the core of our *internal* attacker model; by an untrusted prover, we refer to a host that is involved in bandwidth measurements, however it is not trusted by the verifier to correctly execute the measurement protocols. We assume that untrusted provers need to inflate/deflate their bandwidth claims by a considerable amount ($> 200\%$) to increase their profit in the network.

[1] Simulation details are omitted due to lack of space.

An external attacker Eve can equally compromise routers on the path between the verifier and the prover. By compromising routers, Eve can delay the exchanged probes to alter the bandwidth estimated by the verifier. Eve can also re-route probe packets through another bottleneck link to influence the conducted measurements.

3.2 Attacks on Current Techniques

Bandwidth measurement tools were developed without prior security considerations as they rely on ICMP/TCP implementations at end-hosts and do not guarantee any form of source nor destination authentication. An external attacker can *spoof* the IP [26] of the prover and issue back ICMP replies on its behalf; the measured bandwidth would be that of the attacker. The adversary could also re-route the probes to hosts at its disposal ([19], [20]) to claim a bandwidth of her choice (*sybil attack* [27]). In what follows, we analyze the detrimental impact of ***delay attacks*** on bottleneck bandwidth measurements.

Delay & Rushing Attacks on the One-Packet Technique: An untrusted prover can intentionally *delay* its reply packets to convince the verifier of a bandwidth claim of its choice (Figure 3). Given a set $\mathbb{S}$ of the variable-sized packets used in the one-packet technique, the prover can claim *lower* bandwidths $B_{claimed}$ than its genuine bandwidth B_{auth} by introducing a delay Δ_j to all packets $j \in \mathbb{S}$ of size $S_j > S_i$, where i is the *smallest* packet in $\mathbb{S}$, as follows:

$$B_{claimed} = \frac{S_j - S_i}{RTT_j - RTT_i + \Delta_j} \tag{3}$$

$$\Delta_j = (S_j - S_i) \cdot \left(\frac{1}{B_{claimed}} - \frac{1}{B_{auth}}\right), \tag{4}$$

Here, RTT_j denotes the *smallest* round trip time of probe j from the verifier to the prover. Note that the prover can equally claim a *higher* bandwidth by delaying probes $j \in \mathbb{S}$ of size $S_j < S_k$, where k denotes the *largest* packet in $\mathbb{S}$, by $\Delta_j = (S_k - S_j) \cdot (\frac{1}{B_{auth}} - \frac{1}{B_{claimed}})$.

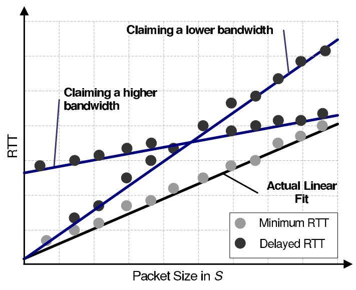


Fig. 3. Delay Attacks on the One-Packet technique

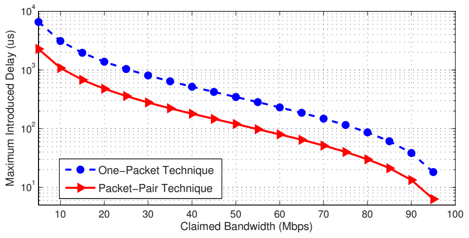


Fig. 4. Maximum delay required to fake bandwidth claims in the one-packet and the packet-pair technique. $B_{auth} = 100$ Mbps, the probe size ranges from 58 bytes to 1500 bytes. The path contains 5 link-layer hops.

In Equation 4, we assume that there are no intermediate hops on the path between the verifier and the prover. In practice, the untrusted prover has to further take into account the delays caused by the intermediate hops. This could be achieved by repeatedly applying Equation 4 for all link-layer hops in the desired link as follows:

$$\Delta_j = \sum_{i=1}^{H} \left((S_j - S_i) \cdot (\frac{1}{B_{claimed}} - \frac{1}{B_{auth}}) \right), \tag{5}$$

where H is the total number of link-layer hops in the measured path. Delay attacks can be very hard to detect given the unnoticeable delay that they introduce (Figure 4). Note that this attack is not only restricted to untrusted provers; a *rogue router* (compromised by Eve) can equally trick the verifier into accepting a fake bandwidth claim by introducing appropriate delays to the packet traversal time.

An untrusted prover can also predict the *Identifier* and *Sequence Number*[2] fields in the ICMP echo request packets and “rush” its reply by sending *specially crafted* ICMP echo replies ahead of time. In this way, an attacker can claim a smaller RTT which translates to a different bottleneck bandwidth measurement. A combination of these rushing and delay attacks could even reduce the maximum delay Δ_j that needs to be introduced to fake bandwidth claims.

Packet-Attraction and Repulsion attacks on the Packet-Pair Technique. In current implementations of the packet-pair technique [10], the verifier sends large back-to-back TCP SYN packets and awaits the corresponding TCP RST packets from the prover. Assuming that the prover immediately replies to the probe requests, this time dispersion will also be reflected in the difference of TCP SYN packet arrival times. By intentionally delaying the second reply probe, an untrusted prover increases the time dispersion between the packet-pairs and consequently the verifier would assume the existence of a smaller bottleneck link on the path to the prover. The required delay Δ is computed as follows:

$$B_{claimed} = \frac{S}{\Delta_{dispersion} + \Delta} \tag{6}$$

$$\Delta = S \cdot (\frac{1}{B_{claimed}} - \frac{1}{B_{auth}}) \tag{7}$$

where $\Delta_{dispersion}$ is the genuine dispersion between the packet-pairs, Δ denotes the additional delay between the packet-pairs, S is the size of the probes, $B_{claimed}$ is the fake claimed bandwidth of the prover and B_{auth} is the genuine bandwidth of the prover. As shown in Figure 4, Δ is considerably small – even for the largest probe size of 1500 bytes – compared to the delay required in the one-packet technique. This suggests that delay attacks are indeed more challenging to detect in the packet-pair technique when compared to the one-packet technique (Section 5.2).

Similarly, an untrusted prover or a rogue router can claim a smaller time dispersion between packet-pairs and consequently a higher download bottleneck

[2] Generally, the *Sequence Number* field in the ICMP echo request is incremental and therefore can be easily predicted.

bandwidth. The prover can delay its reply till both TCP SYN packets are received before sending its packet-pair replies with a time dispersion of its choice. Since RST packets are typically small in size, they will not queue at the bottleneck link. In this way, the prover can successfully claim a higher bandwidth than its genuine physical one.

At first glance, one might consider that these attacks can only be mounted by sophisticated attackers. However, this intuition is *not* correct. While a sophisticated user is able to manipulate his interface to temporarily delay all reply probes, less powerful provers can cause the same effect by using bandwidth shapers as shown in the following section.

3.3 Demonstration of Delay Attacks

In what follows, we demonstrate the feasibility of delay attacks on the one-packet and packet-pair techniques. Our findings are depicted in Figures 5 and 6. In our plots, *target bandwidth* refers to the bottleneck bandwidth claimed by an untrusted prover and *measured bandwidth* denotes the bottleneck bandwidth estimate extracted by the verifier. We rely on 10 and 100 Mbps symmetric physical connections deployed on three paths: **Path1** where both the verifier and the prover hosts (running Ubuntu *v.* 7.04 with 1 GB of RAM) are both located in Switzerland, **Path2**[3] where the verifier and the prover (host running Debian with 2 GB of RAM) are located in Switzerland and Germany, respectively, and **Path3** where the verifier is located in Switzerland and the prover is located in Illinois, USA. The prover runs RedHat Linux with 320 MB of RAM. Each data point in our plots is averaged over 1000 measurements.

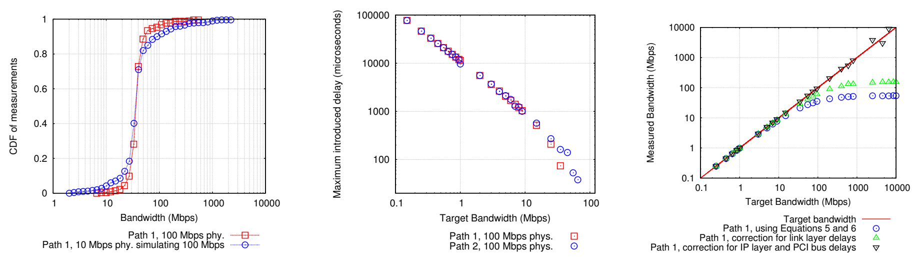
One-Packet Technique. We created a prototype tool based on *Pathchar* [5] that delays the prover's reply packets (Equation 5). We used probe sizes ranging from 58 bytes to 1514 bytes (Ethernet headers included). Our application replaces the kernel's TCP/IP stack by a raw socket and uses an iptable rule to drop all replies issued by the kernel; it then sends back the reply probes with the desired delay.

As shown in Figures 5(a) and 5(c), an untrusted prover can claim any bandwidth of its choice in the one-packet technique by appropriately introducing small – almost unnoticeable – delays before issuing its replies (Figure 5(b)). Given the impact of small delays, the accuracy of the bandwidth claims can be further increased by accounting for the prover's PCI bus delays (Figure 5(c)).

Packet-Pair Technique: We demonstrate delay attacks on Sprobe [10] using an application that modifies the prover's networking interface and an open-source traffic shaper.

The cumulative distribution functions (CDF) of the conducted measurements (Figure 6(a)) suggests that these attacks – whether originating from a modified

[3] We were not able to conduct one-packet experiments on Path2 due to the fact that intermediate routers were blocking the ICMP probes.



(a) Claiming 100 Mbps bandwidth on a 10 Mbps connection.

(b) Maximum introduced delay in claiming a lower bandwidth over a 100 Mbps downlink connection.

(c) Impact of link-layer and PCI bus delays on measurements over a 100 Mbps downlink connection.

Fig. 5. Delay attacks on the One-Packet technique

application or from bandwidth shapers – are almost *statistically* indistinguishable at the verifier's side from *authentic* bandwidth measurements, which renders them very hard to detect.

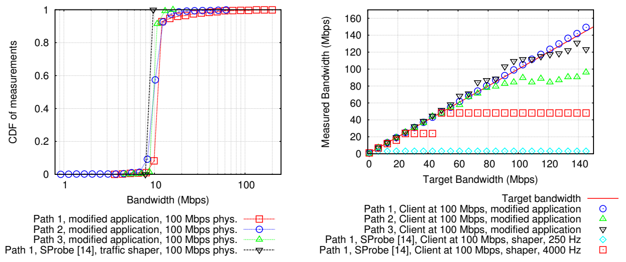
Our analysis in Section 3.2 is further validated in Figure 6(b). Indeed, the prover can claim a bandwidth of its choice irrespective of its actual physical download bottleneck[4]. These attacks can be equally achieved by bandwidth shapers (Figure 6(b)). We further investigate the effect of bandwidth shapers on bandwidth estimation in Section 5.2.

4 Trusted Infrastructure Support for Bandwidth Measurement

To the best of our knowledge, it is hard, if not impossible, for the verifier to *fully* ensure that the remote provers did not intentionally introduce delays before issuing their replies. Although some schemes were proposed in the scope of securing link quality measurement [40] and RTT measurements [41], they assume that the prover does not have incentives to mount delay-based attacks; this is not the case in bandwidth estimation scenarios. An intuitive solution to thwart this problem is to use tamper-resistant hardware [39] to prevent hosts from tampering with their network interface. However, this hardware comes at a high cost.

Remote attestation by trusted network components emerges as one of the few workable alternatives to fully securing bottleneck bandwidth measurements. In what follows, we briefly outline a scheme that makes use of trusted edge-routers and we show that our solution effectively mitigates delay attacks against bandwidth estimation. In Section 5, we discuss several other alternatives to partially alleviate these attacks without requiring infrastructural support.

[4] Note that Path2 featured considerable cross-traffic during the measurements, which explains the estimate errors in the plots.



(a) Claiming 10 Mbps on a 100 Mbps uplink connection. (b) Packet-delay attacks on a 100 Mbps downlink connection.

Fig. 6. Delay attacks on the Packet-Pair technique

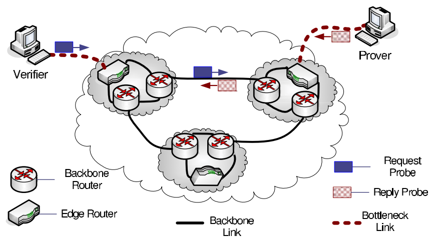


Fig. 7. Bottleneck bandwidth measurements using trusted edge routers

As shown in Figure 7, we assume in our analysis that the bottleneck links reside between the outer-most edge-routers and the end-hosts. Sample experiments on PlanetLab [43] nodes confirm that this is a reasonable assumption. We further assume that edge-routers are trusted by all entities and can timestamp, generate and authenticate packets.

Our scheme for securing bottleneck bandwidth measurements unfolds as follows: when the verifier wishes to measure the bottleneck bandwidth of the path to a prover, it sends along that path a request packet containing the IP address of the prover and the type of bandwidth measurement of interest (upload and/or download). Upon reception of the latter packet, the edge-router connected to the prover measures the capacity of the bottleneck link it shares with the prover and sends its measurement results to the verifier. The verifier can validate the authenticity of the measurement results since they come enclosed with the signature of the edge-router. The edge-router estimates the bottleneck bandwidth of the link it shares with the prover as follows:

- **Upload Bandwidth Measurement.** Similar to the packet-pair technique, the prover sends two large back-to-back packets to the edge-router. Since the latter is located on the other side of the bottleneck link, it can verify that no additional delay Δ (Equation 7) was introduced between the packet-pairs (the edge-router measures the time delay between the last bit of the first packet and the first bit of the second packet is negligible). By doing so, the edge-router is certain that both packets queued at the bottleneck link. It then measures the time dispersion between the packets to estimate the bottleneck link of the path to the prover according to the packet-pair technique.

- **Download Bandwidth Measurement.** To measure the downlink bottleneck of the prover, the edge-router can estimate the time it needs to upload a packet-pair on the path to the prover. Since the bottleneck link is shared by both the prover and the edge-router and assuming a high transmission rate, the latter's upload throughput corresponds to the download capacity[5] of the bottleneck link.

5 "Best-Effort" Solutions for Current Bandwidth Estimation Techniques

In Section 4, we showed that by relying on trusted network components, security threats against bottleneck bandwidth measurements can be fully mitigated. Given the current architecture of the Internet, we do acknowledge, nevertheless, that relying on trusted infrastructure might constitute a rather "bulky" proposal nowadays. In this section, we discuss and evaluate several other "best-effort" countermeasures that do not require trusted infrastructure support.

5.1 Mitigating Spoofing and Rushing Attacks

Bottleneck bandwidth measurement tools can make use of lightweight authentication protocols to counter impersonation attacks. Furthermore, the verifier can use pseudo-random functions to generate its request probes such that they cannot be predicted by the provers and require that the reply probes are correlated in content to its request probes. Alternatively, the verifier can make use of distance bounding protocols [30] or can require that the prover authenticates the received pseudo-random probes using a shared key. Thus, the probability that the prover correctly rushes its replies before receiving the request probes can be made satisfactorily negligible ($O(2^{-k})$ for k-bit probes).

Note that the time required to authenticate each request probe is negligible compared to the probes' propagation times. For example, the time required to encrypt a 1500 bytes message with a 256 bit key using the AES implementation in the Crypto++ library on an Intel Core 2 1.83 GHz processor running Windows XP is 19 μs [21]. We implemented a variant of the Sprobe tool [10] in which the prover is required to encrypt (using AES) the request probes and we have conducted sample bandwidth measurements on Paths 1 and 2 using this application. Our findings in Figure 8 show that the accuracy of the measurements is preserved, which makes AES-based authentication suitable for integration within current bandwidth estimation techniques.

5.2 Alleviating Delay Attacks

In what follows, we discuss some techniques to alleviate delay attacks on bandwidth estimation.

[5] Note that the edge-router can estimate the full capacity of the bottleneck link since it can ensure that no downlink traffic is present at the time of the measurements.

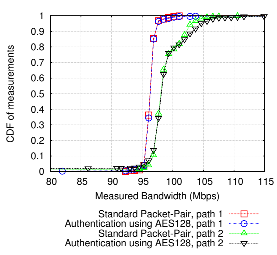


Fig. 8. Effect of Authentication on measurements

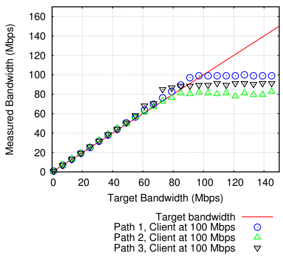


Fig. 9. Attacks on a 100 Mbps uplink connection

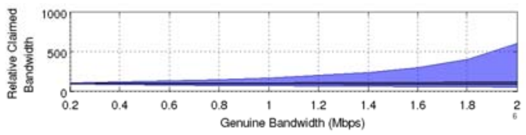
1) Mitigating Bandwidth Inflation Attacks in the Packet-Pair Technique: Given large probe sizes, the packet-pair technique ensures that the upper bound on the *upload* bandwidth that an *untrusted prover* can claim is bounded by its physical bottleneck bandwidth. This is depicted in Figure 9. In fact, the lower bound on the time dispersion between large packet-pairs is determined by the queuing on the bottleneck link. Even if the untrusted prover manipulates the transmission times of its reply probes, they will queue at the bottleneck link with high probability. Given this, the only viable strategy to claim a higher bandwidth would be to send each of the packet-pairs using different paths. However, this requires accurate knowledge of the network status; in practice, the attacker will only succeed with negligible probability. Note that an untrusted prover can also distribute its authentication credentials to other hosts under its control. In this case, the upper limit on the claimed bandwidth is bounded by the highest physical bandwidth of all the compromised hosts.

2) "Reference" Round-Trip Times: Theoretically, delay attacks can be alleviated if the verifier knows an estimate of the RTT to the prover. The verifier can acquire RTT estimates via offline measurements or from online servers that perform RTT measurements around the globe (e.g., [31]).

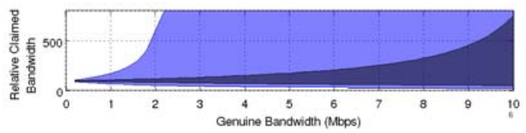
For instance, in the one-packet technique, if the verifier knows a reference RTT for a median-size probe packet in the set $\mathbb{S}$ of the variable probe sizes, then the bandwidth range that a prover can claim is bounded by the accuracy ϵ of the estimated reference RTT as follows:

$$\frac{B_{auth} \times (S_j - S_i)}{\epsilon \cdot B_{auth} + (S_j - S_i)} \leq B_{claimed} \leq \frac{B_{auth} \times (S_k - S_j)}{(S_k - S_j) - \epsilon \cdot B_{auth}},$$

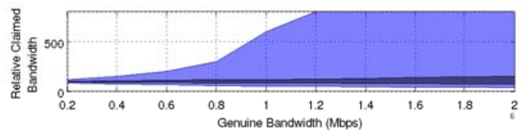
where $B_{claimed}$ is the bandwidth claimed by the untrusted prover, B_{auth} is the genuine bottleneck bandwidth on the prover's side, S_i is the size of the smallest probe packet used in the variable-size probing set $\mathbb{S}$ and S_k is the size of the largest probe packet in $\mathbb{S}$.



(a) $\epsilon = 5\ ms$.



(b) $\epsilon = 5\ ms$.



(c) $\epsilon = 10\ ms$.

Fig. 10. Range of achievable bandwidth claims for $\epsilon = 5\ ms$ and $\epsilon = 10\ ms$. The dark and light areas represent the achievable claims in the one-packet technique and the packet-pair technique, respectively.

Similarly, in the packet-pair technique, the *download* bandwidth that an untrusted prover can claim is equally bounded by the accuracy ϵ of the estimated RTT:

$$\frac{S}{\Delta_{dispersion} + \epsilon} \leq B_{claimed} \leq \frac{S}{\Delta_{dispersion} - \epsilon},$$

where S is the request probe packet size, $\Delta_{dispersion}$ is the time dispersion between the request probe packet-pairs originating at the bottleneck link and ϵ is the acceptable deviation in time from the reference RTT.

We investigate the benefits of this approach in Figure 10 for estimation errors $\epsilon = 5$ ms and $\epsilon = 10$ ms from the reference RTT. Given the variability of RTTs in current networks and the error ϵ in estimating the reference RTT, this technique can only *limit* the range of false claims (within 20 % of the genuine bandwidth[6]) in the case where the genuine bottleneck bandwidths are modest (typically < 10 Mbps). Our findings also show that this technique is not well-suited to upper-bound fake bandwidth claims in the packet-pair technique. This is due to the fact that the time dispersion between packets $\Delta_{dispersion}$ is comparable to typical values of ϵ, even when dealing with small bandwidths. It can, however, significantly lower-bound the claims of modest-bandwidth hosts.

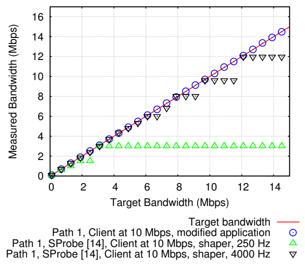
[6] This is rather acceptable compared to the estimation errors resulting from current bandwidth estimation tools.

3) Detecting Bandwidth Shapers in the Packet-Pair Technique: Bandwidth and traffic shapers (e.g., NetLimiter [28], NetEqualizer [29], HTB [32]) provide a simple mechanism to limit the amount of data a host transmits and accepts by delaying incoming and outgoing packets to match a specified rate limit. Due to their mode of operation, bandwidth shapers *cannot* alter the measurements conducted by the one-packet technique since they cannot limit the rate at which *individual* probe packets are sent. However, they present themselves as effortless routines to conduct delay attacks on the packet-pair technique. We implemented a prototype shaper script on the prover's side and we studied its impact on the Sprobe tool [10]. Our script uses iptable rules and the HTB traffic shaper [32] to throttle the bandwidth of the prover on the fly. We have also conducted upload bandwidth measurements on Sprobe using NetLimiter [28] running on a Windows XP kernel. Our measurements were conducted on **Path 1** (refer to Section 3.3).

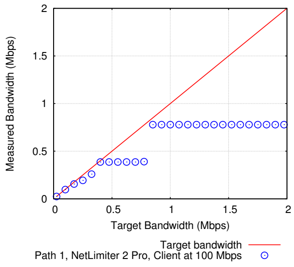
Our findings in Figure 11 suggest that current implementations of bandwidth shapers allow a verifier to *detect* their deployment on the prover's side. In fact, bandwidth shapers can only receive, store, and release packets whenever a system timer interrupt occurs [32]. This suggests that the maximum rate at which a *pair of packets* can be sent is bounded by the timer frequency of the underlying operating system: $B_{max} = S \cdot F_{sys}$, where B_{max} is the maximum achievable bandwidth claim, S is the packet size and F_{sys} is the system timer frequency. Furthermore, the achievable time dispersions between a packet-pair $T_{nominal}$ are inversely proportional to the system timer frequency F_{sys}. The achievable bandwidth claims are therefore computed as follows:

$$B_{claimed} = \frac{S \cdot F_{sys}}{i}, \forall i \in \mathbb{N}^*, \tag{8}$$

which explains the step-wise curves obtained in Figure 11. In most Linux systems, it is however possible to increase the system frequency through kernel re-compilation. As shown in Figure 11(a), a prover can achieve a higher upper bound on the claimed bandwidth by re-compiling its kernel to operate at a higher timer frequency.

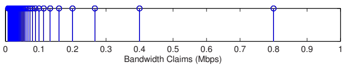


(a) Effect of the HTB Shaper.

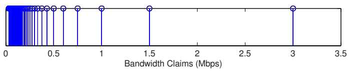


(b) Effect of NetLimiter.

Fig. 11. Effect of Bandwidth/Traffic Shapers



(a) $F_{sys} = 66.67\ Hz$



(b) $F_{sys} = 250\ Hz$

Fig. 12. Achievable bandwidth claims by traffic shapers in the packet-pair technique. A spike at value X indicates that bandwidth X can be achieved.

By comparing the time dispersions of the reply packet-pairs with $T_{nominal}$ for typical system frequencies, a verifier can suspect the presence of bandwidth shapers on the provers' side and can rule out the resulting estimate. This is especially true for small bandwidths (Figure 12).

We validate this claim via extensive measurements on 200 PlanetLab provers. To truthfully represent Internet nodes, we chose the PlanetLab nodes whose bandwidth distribution follows the distribution in the current Internet [38]. In accordance with the findings in [3], we assume that 40 % of the provers are selfish and make use of bandwidth shapers to vary their bandwidth claims over time. Untrusted provers can claim both higher (inflate) or lower (deflate) bandwidths by factors ranging from 1 to 10; as suggested in [3], we assume that high-bandwidth provers claim higher bandwidths with probability 0.1 and lower bandwidths with probability 0.9. Low-bandwidth provers claim higher bandwidth capabilities with probability 0.9.

In our experiments, we compute the median B_{med} of the measured bandwidths (typically 10 packet-pairs to remove noisy measurements) and we compare it to the *closest* bandwidth that a shaper can achieve B_{min} as follows: we assume a normal distribution[7] around B_{min} and we compute the probability P that B_{med} is within a threshold number of standard deviations $(n \cdot \sigma)$ of B_{min}:

$$P = \frac{1}{\sigma\sqrt{\pi}} \int_{B_{min}-n\sigma}^{B_{min}+n\sigma} e^{-\frac{(x-B_{min})^2}{2\sigma^2}}, \tag{9}$$

Our results are illustrated in Figure 13; a significant fraction (76 %) of provers that use bandwidth shapers were correctly identified, which confirms the feasibility of bandwidth shaper detection in the packet-pair technique. Since the granularity of bandwidths that a bandwidth shaper can produce is small for low bandwidths, it can emulate a large number of low bandwidths (Figure 12). Given this, and in the presence of cross-traffic, the claims of low-bandwidth provers (e.g., modem users) can be easily mis-judged to be originating from bandwidth shapers. This explains the false negatives obtained in identifying honest provers.

Note that some Linux built-in traffic shapers (Linux kernel *v.* 2.6.23) do not rely on system timer interrupts. Thus, other techniques will be needed to detect them in the future. Windows-based shapers (e.g., NetLimiter [28]) can, however, still be detected using the aforementioned method.

[7] Experimental results conducted on 200 different Internet paths confirm this assumption.

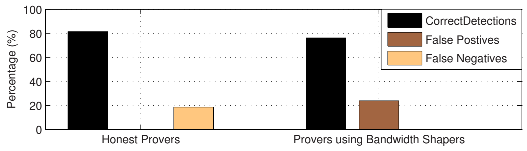


Fig. 13. Bandwidth Shaper Detection conducted on 200 different PlanetLab hosts

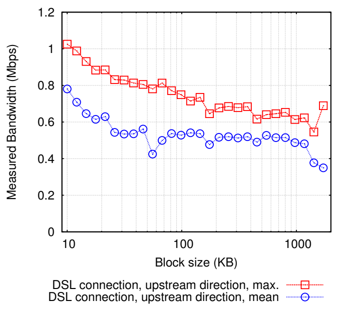


Fig. 14. Tradeoffs in the Block-Transfer method

4) Verifying Bandwidth Capability. The block transfer method is a conventional mean to measure the *available* bandwidth (the unused capacity) of a path. Here, the verifier asks the prover to download/upload a full block of data and computes the data transfer rate to estimate the residual bandwidth of the path to the prover. Several applications make use of this technique to estimate the *available* bandwidth of a path (e.g., BitTorrent [15]). In the block transfer method, the upper bound on the claimed bandwidth is guaranteed to match the genuine bottleneck bandwidth. This allows the verifier to *identify* whether the bottleneck bandwidth claimed by the prover can be achieved in practice. However, this solution incurs significant overhead in the network and depends on the traffic load in the path. Nevertheless, our findings suggest the existence of a potential tradeoff with respect to the required data transfer size. We conducted block transfer measurements from a high-speed verifier to a prover connected to the Internet by a 0.8 Mbps upload connection. Figure 14 depicts the tradeoff between the size of the transfer block and the accuracy of the bandwidth estimated by the verifier. Indeed, even blocks of moderate size (50-100 KB) result in an indicative bandwidth estimate, which might justify the use of this method to filter out suspicious bandwidth claims.

5) Reverse-Resolve DNS Names. By resolving a prover's IP address into its Domain Name Server (DNS), the verifier might deduce the prover's type of Internet connection and detect false bandwidth claims. For example, if the prover's DNS name contains the string "dsl", it is highly likely that it has a DSL Internet connection [10]. We evaluated the viability of this proposal through extensive experiments on *1,000,000* randomly chosen IPs. We classified the obtained DNS names depending on whether they contain the strings: "dsl", "cable", "dial", "isdn", "WLAN" and "T1" or "T3". Our findings indicate that 34 % of the IPs leak their host's bandwidth information[8] (Figure 15). This information can be used by the verifier to detect discrepancies in the measured bandwidth. For in-

[8] Our results could be further improved given better knowledge of the local providers specific to each country (e.g., AT&T for DSL in the USA).

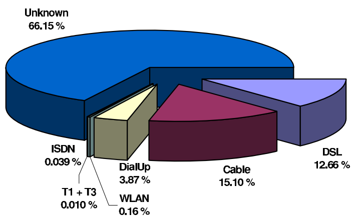


Fig. 15. Parsing results of the DNS names of 1,000,000 randomly chosen IPs around the globe

	Detection Rate	False Positives
Honest Provers	81.4 %	18.6 %
Bandwidth Shapers	75.6 %	21.2 %
Provers with a Modified Interface	68.1 %	31.9 %
Overall	**74.5 %**	**25.5 %**

Fig. 16. Detection Results on 200 PlanetLab Nodes

stance, if the verifier measures a 5 Mbps download bandwidth while the prover's DNS name is "smartuser.dialup.com", then it is highly likely that there was an attack on bandwidth estimation.

6) Additional Heuristics: Statistical outlier detection [22], [23], [24] can also be used to prevent untrusted hosts from faking their bandwidth claims. Using outlier detection methods, correlations between different measurements can be identified and discrepancies can be detected. Furthermore, it is often the case that various performance metrics implicitly exhibit well-defined correlations [22], [44], which might allow the verifier to detect inconsistencies. For example, it is unlikely that a host having a 5 Mbps download bandwidth will have a 10 Mbps upload bandwidth.

To evaluate the viability of this proposal, we refined the PlanetLab experiment described in Section 5.2-2. We assume that a dedicated server periodically monitors the bandwidth claims of hosts and keeps *history* of the recorded measurements. We consider the following setting: 20 % of the hosts modify their networking interface to fake their bandwidth claims, 20 % of the hosts make use of bandwidth shapers and the remaining 60 % are "honest" hosts. We use a combination of bandwidth shaping detection (described earlier) and outlier detection based on the Z-score test to identify malicious hosts that fake their bandwidth claims.

$$Z = \frac{\sqrt{n}(X - \mu)}{\sigma}, \tag{10}$$

Here, n is the number of measurements per host, μ is their mean and σ is the standard deviation from the mean. If the P-value of the Z-score is above a threshold value (0.05), the host is considered to be malicious.

Our findings (Table 16) suggest almost 75 % of the fake claims were successfully detected; most of those detections correspond to provers that use bandwidth shapers and/or that vary their bandwidth claims over time. Larger detection rates could be achieved by incorporating additional techniques, such as reverse-resolve DNS names and reputation-based approaches [33], [34], [36], [37] in the detection process. In the latter approach, each host can be associated with a *reputation value* that indicates how trustworthy it is. Interacting hosts measure their respective bandwidth and form an opinion about each other. Malicious hosts, claiming incorrect bandwidths or varying their bandwidth claims, will be associated with low reputation values and, therefore, will not be chosen in subsequent interactions.

6 Discussion and Outlook

A great deal of lessons can be extracted from the operation of current bottleneck bandwidth estimation tools. We therefore hope that our findings hint application designers on the design of secure bandwidth measurement tools:

"Security" Features of Current Techniques. Till recently, the accuracy and the overhead of bandwidth estimation techniques have highly influenced the decision of application designers to choose a certain estimation technique (e.g., one-packet, packet-pair) given the requirements of their applications [10], [11]. However, "security" is another important factor that needs to be taken into account to ensure consistent bandwidth measurements. Although the design of current techniques cannot give "clear-cut" security guarantees, our findings suggest that some techniques are likely to perform better than others in different adversarial settings. On one hand, the packet-pair technique cannot prevent untrusted provers from inflating nor deflating their download bandwidth claims. Although it can successfully deflate its upload bandwidth, an untrusted prover cannot inflate its upload bandwidth claims given large probes in the packet-pair technique. An important observation here is that bandwidth deflation/inflation attacks on the packet-pair technique can be achieved by bandwidth shapers, and thus can be easily realized by untrusted provers. On the other hand, delay attacks on the one-packet technique require more sophisticated users, capable of altering their networking interface, since bandwidth shapers cannot affect the measurements in the one-packet technique. Fortunately, bandwidth manipulation attacks mounted by modest-bandwidth provers might be successfully mitigated in both techniques (with the exception of download bandwidth inflation attacks) if the verifier knows an estimate of the RTT to the prover.

Active and Cooperative Measurements. Some previous work [10] argues that bandwidth estimation tools should be designed to work in uncooperative environments in order to scale to a large number of hosts. Although this is indeed a desirable property, we find support for uncooperative environments rather unrealistic. In fact, with the proliferation of "de-facto" security applications, such as home firewalls, probing techniques based on uncooperative TCP/UDP and ICMP functionality find less applicability in the near future as they are likely to be considered hostile by the end-hosts. Some routers already *filter* ICMP packets due to their potential malevolent use [11]. Therefore, support for cooperative measurements is inevitable in the near-future [10]. Furthermore, end-to-end security would impair the use of bandwidth monitoring tools in passive and uncooperative environments as it involves active end-host cooperation for source authentication; cooperative environments present themselves as vital "playgrounds" for secure end-to-end bandwidth monitoring in the current Internet.

Network Measurements as "First Class Citizens". Current measurement tools do not take into account the impact of untrusted hosts on bandwidth measurements. Given the current trends in designing a "clean-slate" future Internet, our findings indirectly motivate the need for a *secure* next-generation Internet. Since network measurements are gaining paramount importance in monitoring the performance of the Internet, *secure* infrastructural support for network measurements becomes rather a necessity. As shown in Section 4, by pushing functionality from end-hosts back to *dedicated* and *trusted* network components, several security threats can be eliminated. 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. This would indeed eliminate the need for active probing-based end-to-end *insecure* measurement tools.

7 Conclusions

In this paper, we analyzed and demonstrated the major security vulnerabilities of current bottleneck bandwidth estimation techniques. Given the increasing reliance on bandwidth estimation tools in current Internet services, these vulnerabilities might affect the performance of all the applications that make use of these tools. Another important aim of this work is to extract relevant lessons about the security prospects of existing bottleneck estimation techniques and to hint application designers on the choice of a bandwidth estimation technique that better suits their applications. Our findings suggest that it is very hard, if not impossible, to *fully* counter all security challenges against existing tools without requiring functionality from trusted network components. More specifically, delay attacks pose serious challenges to the consistency of bandwidth measurements. Nevertheless, we proposed other possible solutions and heuristics – that do not require infrastructural support – to mitigate attacks on existing tools and we showed via extensive measurements on PlanetLab nodes that they can alleviate a significant fraction of attacks on current bottleneck bandwidth measurement techniques.

Acknowledgments

The authors would like to thank the anonymous reviewers for their helpful suggestions and feedback.

References

1. Ratnasamy, S., McCanne, S.: Inference of Multicast Routing Tree Topologies and Bottleneck Bandwidths using End-to-end Measurements. In: Proceedings of IEEE INFOCOM (1999)
2. Schiely, M., Renfer, L., Felber, P.: Self-Organization in Cooperative Content Distribution Networks. In: Proceedings of NCA (2005)

3. Saroiu, S., Gummadi, P., Gribble, S.: A Measurement Study of Peer-to-Peer File Sharing Systems. In: MMCN (2002)
4. Lai, K., Baker, M.: Nettimer: A Tool for Measuring Bottleneck Link Bandwidth. In: USITS (2001)
5. Jocobson, V.: Pathchar (1997), `http://www.caida.org/tools/taxonomy/perftaxonomy.xml#pathchar`
6. Math, B.: pchar (1999), `http://www.caida.org/tools/taxonomy/perftaxonomy.xml#pchar`
7. Clink: a tool for estimating Internet link characteristics, `http://allendowney.com/research/clink/`
8. Carter, R.: Cprobe and bprobe Tools (1996), `http://cs-people.bu.edu/carter/tools/Tools.html`
9. Dovrolis, C.: pathrate (2001), `http://www.cis.udel.edu/~dovrolis/bwmeter.html`
10. Sariou, S., Gummadi, P., Gribble, S.: SProbe: A Fast Technique for Measuring Bottleneck Bandwidth in Uncooperative Environments. In: Proceedings of INFOCOM (2002)
11. Lai, K., Baker, M.: Measuring Link Bandwidths Using a Deterministic Model of Packet Delays. In: ACM SIGCOMM (2000)
12. KaZaA, `http://www.kazaa.com/`
13. Strauss, J., Katabi, D., Kaashoek, F.: A Measurement Study of Available Bandwidth Estimation Tools. In: IMC (2003)
14. Hu, N., Li, L., Mao, Z., Steenkiste, P., Wang, J.: A Measurement Study of Internet Bottlenecks. In: Proceedings of INFOCOM (2005)
15. BitTorrent, `http://www.bittorrent.org/protocol.html`
16. Carter, R., Crovella, M.: Measuring Bottleneck Link Speed in Packet-Switched Networks. In: Performance Evaluation (1996)
17. Dovrolis, C., Ramanathan, P., Moore, D.: What do packet dispersion techniques measure? In: Proceedings of INFOCOM (2001)
18. Prasad, R., Dovrolis, C., Murray, M., Claffy, K.: Bandwidth estimation: metrics, measurement techniques, and tools. IEEE Network (2003)
19. Revealed, the Internet's Biggest Security Hole, `http://blog.wired.com/27bstroke6/2008/08/revealed-the-in.html`
20. More on BGP Attacks, `http://blog.wired.com/27bstroke6/2008/08/how-to-intercep.html`
21. Speed Comparison of Popular Crypto Algorithms, `http://www.cryptopp.com/benchmarks.html`
22. Walters, A., Zage, D., Nita-Rotaru, C.: A Framework for Mitigating Attacks Against Measurement-Based Adaptation Mechanisms in Unstructured Multicast Overlay Networks. ACM/IEEE Transactions on Networking (2007)
23. Soule, A., Salamatian, K., Taft, N.: Combining Filtering and Statistical Methods for Anomaly Detection. In: Proceedings of IMC (2005)
24. Snader, R., Borisov, N.: EigenSpeed: Secure Peer-to-peer Bandwidth Evaluation. In: Proceedings of IPTPS (2009)
25. Savage, S., Cardwell, N., Wetherall, D., Anderson, T.: TCP Congestion Control with a Misbehaving Receiver. Computer Communication Review (1999)
26. Harris, B., Hunt, R.: TCP/IP security threats and attack methods. Computer Communications (1999)
27. Douceur, J.: The sybil attack. In: Druschel, P., Kaashoek, M.F., Rowstron, A. (eds.) IPTPS 2002. LNCS, vol. 2429, p. 251. Springer, Heidelberg (2002)

28. NetLimiter, `http://www.netlimiter.com/`
29. NetEqualizer, `http://www.netequalizer.com/`
30. Brands, S., Chaum, D.: Distance-bounding protocols. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 344–359. Springer, Heidelberg (1994)
31. The CAIDA DNS root/gTLD RTT Dataset, `https://data.caida.org/datasets/dns/root-gtld-rtt/`
32. HTB Traffic Shaper, `http://luxik.cdi.cz/~devik/qos/htb/`
33. Kamvar, S., Schlosser, M., Garcia-Molina, H.: The EigenTrust Algorithm for Reputation Management in P2P Networks. In: WWW (2003)
34. Sears, W., Yu, Z., Guan, Y.: An Adaptive Reputation-based Trust Framework for Peer-to-Peer Applications. In: NCA (2005)
35. Damiani, E., Vimercati, S., Paraboschi, S., Samarati, P.: Managing and Sharing Servents' Reputations in P2P Systems. IEEE Transactions on Knowledge and Data Engineering (2003)
36. Dimitriou, T., Karame, G., Christou, I.: SuperTrust: A Secure and Efficient Framework for Handling Trust in Super Peer Networks. In: Proceedings of ACM PODC (2007)
37. Karame, G., Christou, I., Dimitriou, T.: A Secure Hybrid Reputation Management System for Super-Peer Networks. In: Proceedings of IEEE CCNC (2008)
38. OECD, Broadband Growth and Policies in OECD Countries, `http://aui.es/IMG/pdf_Informe_OCDE_Banda_Ancha_en_el_Mundo.pdf`
39. Jin, H., Lotspiech, J.: Forensic Analysis for Tamper Resistant Software. In: Proceedings of ISSRE (2003)
40. Zeng, K., Yu, S., Ren, K., Lou, W.: Towards Secure Link Quality Measurement in Multihop Wireless Networks. In: Globecom (2008)
41. Courtay, O., Karroum, M., Duran, A.: Method and Devices for Secure Measurements of Time-Based Distance Between Two Devices. Patent no. WO/2006/136278 (2006)
42. Barford, P.: Measurement as a First Class Network Citizen. White Paper, `http://pages.cs.wisc.edu/~pb/sngi_whitepaper.pdf`
43. PlanetLab, `http://www.planet-lab.org/`
44. Jiang, G., Cybenko, G.: Temporal and spatial distributed event correlation for network security. In: American Control Conference (2004)