Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computer Networks 47 (2005) 1–21
COMPUTER NETWORKS
www.elsevier.com/locate/comnet

# Measuring IP and TCP behavior on edge nodes with Tstat

M. Mellia [a,*], R. Lo Cigno [b], F. Neri [a]

[a] Politecnico di Torino—Dipartimento di Elettronica, Corso Duca degli Abruzzi 24, I-10129 Torino, Italy
[b] Dipartimento di Informatica e Telecomunicazioni—Università di Trento, Via Sommarive 14, I-38050 Povo, Trento, Italy

Received 7 October 2003; received in revised form 28 March 2004; accepted 22 June 2004
Available online 13 August 2004

Responsible Editor: V. Tsaoussidis

**Abstract**

Field measurements have always been the starting point for network design and planning; however, their statistical analysis beyond simple traffic volume estimation is not so common. In this paper we present and discuss Tstat, a tool for the collection and statistical analysis of TCP/IP traffic, which, in addition to recognized performance figures, infers TCP connection status from traces. Besides briefly discussing its rationale and use, we present part of the performance figures that can be obtained, and we highlight the insight that such figures can give on TCP/IP protocols and the Internet, thereby supporting the usefulness of a widespread use of Tstat or similar tools.

Analyzing Internet traffic is difficult because a large amount of performance figures can be devised in TCP/IP networks, but also because many performance figures can be derived only if both directions of bidirectional traffic are jointly considered. Tstat automatically correlates incoming and outgoing packets. Sophisticated statistics, obtained through data correlation between incoming and outgoing traffic, give reliable estimates of the network performance also from the user perspective.

Tstat computes over 80 different performance statistics at both the IP and TCP layers, allowing a good insight in the network performance. To support the latter statement, we discuss several of these figures computed on traffic measurements performed for a time period equivalent to more than three months spread during the years 2000–2003 on the access link of Politecnico di Torino.


*Keywords:* Internet traffic measurements; TCP performance; Passive measurements

## 1. Traffic measurements in the Internet

Designing and planning telecommunication networks was always based on traffic measurements, out of which traffic models and performance

* Corresponding author.
*E-mail address:* mellia@polito.it (M. Mellia).

1389-1286/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.comnet.2004.06.026

estimates are built. While this process proved to be reasonably simple in traditional, circuit-switched telephone networks, it seems to be much harder in packet-switched data networks. In the Internet in particular, the TCP/IP client–server communication paradigm introduces correlation both in space and time, and Internet traffic exhibits much stronger time dependencies than telephone traffic.

While a large part of this difficulty lies in the failure of traditional modeling paradigms [1,2], there are also several operative key problems to be solved in performing the measurements themselves and, most of all, in organizing the enormous amount of data that are collected through measurements.

First of all, the client–server communication paradigm implies that the traffic behavior does have a deeper meaning when the forward and backward directions of flows are jointly analyzed, otherwise half of the story goes unwritten, and it should be hardly inferred (with high risk of making mistakes). This problem makes the measurement process inherently difficult. If measurements are taken on the network edge, where the outgoing and incoming packets are necessarily coupled, correlating the bidirectional information can be easy, but it can prove harder in the backbone, where the peering contracts among providers and hot-potato routing often make the forward and backward routes disjoint [3].

Second, data traffic must be characterized at higher levels of detail and at different time granularities with respect to circuit-switched voice traffic. A telephone call can be represented as a constant bit rate (CBR) connection with a given duration, hence characterizing the arrival process and duration distribution of voice calls is sufficient. On a data network like the Internet, where the network protocol (IP) is connectionless and there is no connection admission control, even the notion of connection becomes blurred. Instead, there are clear (though possibly difficult to represent) notions of:

- *packet*: as a chunk of information with atomic characteristics, that is either delivered correctly to the destination or is lost (and most often has to be retransmitted);
- *flow*: as a linearly ordered set of correlated packets with timestamps, as in a TCP connection; a more precise definition of how `Tstat` identify flows is given in Section 1.2;
- *session*: whenever a user is accessing the Internet, receiving or transmitting data or simply reading a Web page, he/she activates a number of correlated flows collectively called session.

The layered structure of the TCP/IP protocol suite requires the analysis of traffic at least at the IP (packet), TCP/UDP (connection[1]), and Application/User-behavior (session) layers.

Starting from the works of Danzig [4–6], and Paxons [1,7], the interest in data collection, measurement and analysis to characterize either the network or the user behavior increased steadily, also because it was clear from the very beginning that "measuring" the Internet was not an easy job. The lack of a simple, yet satisfactory model, like the Erlang teletraffic theory in telephone networks, spawned a research effort in several directions.

At the packet layer, measurements show that the arrival process of packets is complex and correlated. We only recall the first seminal works [2,8], but many others exist that either analyze the traffic characteristics, or try to offer models to represent them. These works concentrate on packets, disregarding the interaction with higher-layer protocols.

At the application layer, after the birth of the Web, many works analyzed traces, typically log files of Web or proxy servers [9–11]. Traffic analysis at this layer gives insight into applications and user behavior, but often fails to correlate high-level phenomena with network characteristics. Other projects analyzed the same high-level phenomena using traffic traces, captured from large campus networks, like the work in [12], in which the authors characterize the HTTP protocol by using large traces collected at the university campus in Berkeley. In [13] the authors present data collected from a large Olympic Games server in 1996, whose findings are helpful to understand TCP behavior,

[1] In this paper we interchangeably use the term "TCP connection" and "TCP flow".

like loss recovery efficiency and ACK compression. In [14], the authors analyzed more than 23 millions HTTP connections, and derived a model for the connection inter-arrival time. Similarly, the authors of [15] analyze and derive models for the Web traffic, starting from the TCP/IP header analysis. More recently, there has been an increasing interest in joint measurements of IP packets and TCP flows, and to the network monitoring in itself, as testified by the work carried by several research groups [16–19].

To the best of our knowledge, however, there are no measurement tools, nor measurement-based works, that characterize the traffic at the three layers identified above, and in particular at the flow layer, rebuilding the status of individual TCP connections, correlating the observations with those at the packet and application layers, while `Tstat` was devised to operate at all three layers. This paper mainly concentrates on flow-layer analysis, and demonstrates the advantages and the deeper insight on traffic behavior permitted by this approach.

### 1.1. Available tools and related work

There are several tools available to derive "network measurements." We are interested in freely available *passive* tools which analyze traces without modifying the status of the network. Among them, many are based on the `libpcap` library developed with the `tcpdump` tool [20,21].

Among the freely available tools, `tcpanaly` [22] is an interesting tool for automatically analyzing a TCP implementation behavior by inspecting packet traces. While giving very detailed statistics on a single tracked TCP connection, it does not provide an automated generation of statistics, and cannot run on live links, as it performs a two-stage analysis. Another interesting tool is `tcptrace` [23], which rebuilds a TCP connection status from previously collected traces, matching data segments and ACKs. For each connection, it keeps track of elapsed time, bytes/segments sent and received, retransmissions, round trip times, window advertisements, throughput, etc. `Tstat` extends `tcptrace` by allowing real-time analysis of traces, and by creating aggregate and more detailed statistics. At the IP layer, `ntop` [24] collects statistics, enabling users to track relevant network activities, including traffic characterization, network utilization, network protocol usage. The `ntop` program is intended to be used by administrators of small-size networks, allowing the easy estimation of the network status and of the hosts connected to it. From a traffic analysis perspective, a limited set of functionalities that permit the study of the statistical properties of the traffic is available.

Intrusion detection systems, such as `Bro` [25], have different goals, but can be quite similar to `Tstat` in their operative details. Indeed `Bro`, to detect misbehaving or intruding users, keeps track and correlates packets to identify streams that are not bound to a permitted network application.

Several passive monitoring projects allows researchers to acquire deeper insight into IP traffic. Among them the two most famous are probably OCX-mon [26] from CAIDA, the Cooperative Association for Internet Data Analysis, and IPMON [27] from SprintLabs. Within those projects, several tools have been produced, but often they focus on a single measurement and lack flexibility.

Indeed, many more tools exist than those we just mentioned, but they are either *active* tools, as for example those based upon the classic `ping` or `traceroute` utilities, or commercial tools, whose source code is not available. The latter are hard to use for a research project, where the implementation of innovative measurement techniques is the focal point.

### 1.2. The role of `Tstat`

In light of the above discussion, a major motivation to develop "yet another" tool, named `Tstat` [28] was the need for automatic tools able to produce statistical data from collected network traces in flexible and innovative ways. `Tstat`, starting from standard software libraries, aims to offer network managers and researchers classic and novel performance indexes, as well as statistical data about Internet traffic. The implementation of `Tstat` is fully in line with the open software approach; it builds upon pre-existing software, and is open to the community for further enhancements.

Besides common IP statistics, derived from the analysis of the IP header, Tstat is able to rebuild the status of each TCP connection, looking at TCP headers in the forward and backward packet flows.

A TCP flow starts when the first SYN segment from the client is observed, and is terminated when the tear down sequence of segments is observed (either FIN-ACK from both half connections, or RST segment from one half connection). Given the possibility that a closing sequence of a flow under analysis is never observed (for example because of host or network misbehavior), a timer is used to trigger a garbage collection procedure to free memory: if no segments of a given flow are observed in between two garbage collections, the used flow status memory is freed, and the flow is considered abnormally terminated. In this paper, we set the garbage collector timer to 15 min, which was shown to guarantee very good flow identification [29].

The TCP flow analysis allows the derivation of novel statistics, such as the congestion window size, out-of-sequence segments, duplicated segments, etc. The detailed description of Tstat is deferred to Section 5 (after we have discussed some interesting results obtained with it) and, most of all, to the Tstat home page [28].

In summary, the project that led to the creation of this paper offers two different contributions in the field of Internet traffic measurements:

- A tool for gathering and elaborating Internet measurements. The tool is easy-to-use, it is free, open-source, and available to the research community [28].
- The identification and description of several interesting performance measurements that can be obtained from the Internet with the tool above, discussing their relation with the network behavior and its possible evolution. Such measurements are both at the IP and TCP layers.

What is still missing, to complete the three-layer measurement scenario depicted at page 2, is the identification of the session layer. This task proved by now to be resilient to a number of efforts, mainly due to intrinsic difficulties in finding a commonly accepted definition of session and a way to measure it.

In the rest of the paper we assume that the reader is familiar with the Internet terminology, which can be found for example in [30–32]. First, in Sections 2–4, we discuss several performance indexes, most of them innovative, applied to measurements done on the access router of Politecnico di Torino. In Section 5, some details about Tstat are given, concentrating on those that we deem most interesting. Finally, Section 6 provides concluding remarks.

## 2. Trace analysis

Tstat can collect measurements at any point in the Internet, as long as it is possible to sniff packets from a link. Fig. 1 shows the setup we used to collect traces at the ingress link of Politecnico di Torino. During the roughly three years since the first working version of Tstat was available, several traces were collected and elaborated from different points of the Internet. Results of the analysis of traces that were collected at Politecnico di Torino are available on the Tstat Web site [28] and discussed in this paper. These data are collected on the Internet access link of Politecnico di Torino, i.e., between the border router of Politecnico and the access router of GARR/B-TEN [33], the Italian and European Research network. Within the Politecnico campus LAN, there are approximately 7000 hosts; most of them are clients, but several servers are also regularly accessed from the outside. The backbone of our campus LAN is based on a switched Fast Ethernet infrastructure. It behaves as a stub Internet subnetwork: there is a single point of access to the GARR/B-TEN network and, through it, to the public Internet.

Among the available data we selected three different time periods to highlight the features of Tstat:

- JUN.00: from 6/1/2000 to 6/11/2000. The bandwidth of the access link was 4 Mbit/s, and the link between the GARR network and the corresponding US peering was 45 Mbit/s.

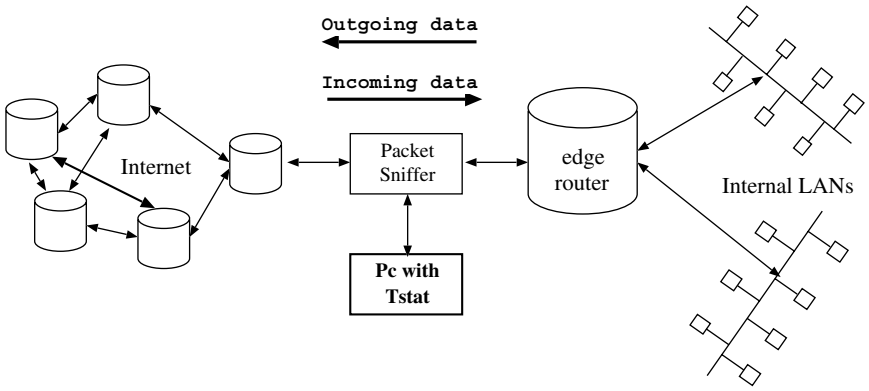


Fig. 1. The measuring setup at the edge router of Politecnico di Torino.

- JAN.01: from 1/19/2001 to 2/11/2001. The bandwidth of the access link was 16 Mbit/s, and the link between the GARR network and the corresponding US peering was 622 Mbit/s.
- OCT.02: from 10/22/2002 to 10/31/2002. The bandwidth of the access link was 28 Mbit/s, and the link between the GARR network and the corresponding US peering was 2.5 Gbit/s.[2]

The three periods are characterized by a significant upgrade in network capacity. In particular, the link between GARR and the US was a bottleneck during JUN.00, but not during JAN.01 and OCT.02. This link plays a key role, since a large part of the traffic comes from US sites and all of it crosses this link. During 2002 a significant upgrade of both access link and US peering became available. In addition, during this period, a more strict regulation of the network facilities was introduced by means of a firewall architecture which blocked (most of) the peer-to-peer traffic. In the remaining of the paper, every time we observe a meaningful difference in the measurements during different periods, we report all of them. Otherwise, we report only the most recent one.

Table 1 summarizes the three traces reporting the total number of packet and flows traced and the share of protocols above IP. Not surprisingly, the largest part of the traffic is transported using TCP, with the UDP traffic at about 5%; other protocols are practically negligible. The number of TCP flows analyzed is larger than 4 and 7 millions in the first two periods respectively, and grows to about 27 millions in the last period, testifying to the traffic growth which followed the network upgrades.

Table 1
Quantitative data amount summary of the analyzed traces

| Period | Packets [$10^6$] | Protocol share [%] | | | Flows [$10^6$] |
|---|---|---|---|---|---|
| | | Other | UDP | TCP | |
| JUN.00 | 242.7 | 0.75 | 5.96 | 93.29 | 4.48 |
| JAN.01 | 401.8 | 0.59 | 4.31 | 95.10 | 7.06 |
| OCT.02 | 1123.6 | 0.21 | 4.88 | 94.91 | 26.9 |

Fig. 2 analyzes the traffic load during the ten measurements days of OCT.02. The upper plot reports the traffic volume subdivided per IP payload type normalized to the link capacity. The lower plot reports the TCP flow arrival rate (flows/s) during the same period of time. The alternating effects between days and nights, and between working days and weekend days is clearly visible for both TCP and UDP traffic. Also the correlation between the traffic volume and the connections arrival rate is straightforward. On the contrary, the percentage of "other" protocols (ICMP, OSPF, GRE, ...), is smaller than 0.1% of the link capacity, and more bursty. A network outage[3] is visible on the 24th of October, which is clearly evidenced

[2] In all cases, the data-link layer of the campus access link is based on an AAL-5 ATM Virtual Circuit.

[3] Probably a failure of the internal switched LAN that manages traffic between different departments.

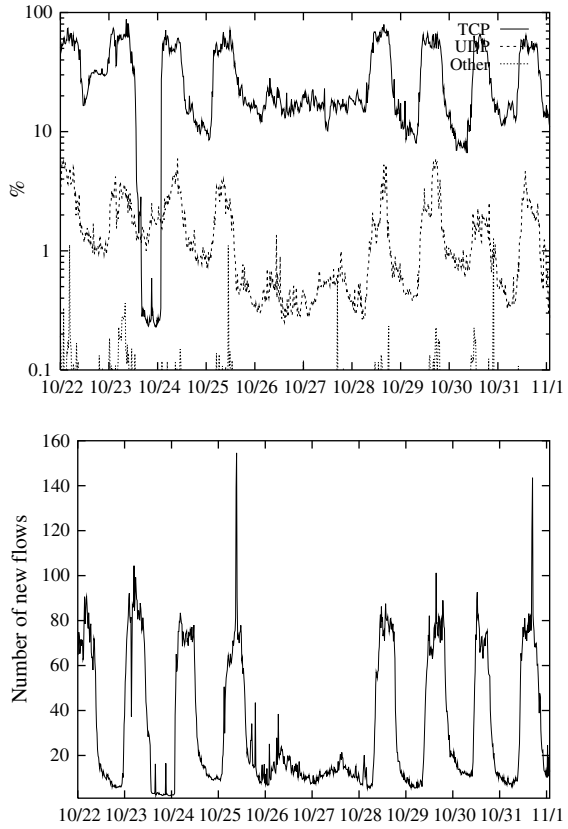


Fig. 2. Percentage of the incoming traffic per IP payload normalized to the link capacity (top) and TCP flows arrival rate in flows/s (bottom) during OCT.02.

by the sudden decrease of TCP packets/flows. Similarly, the TCP flows time-series plot shows also two spikes in the number of flows tracked by `Tstat`, which are due to probably a port scan attempt.

With a quick analysis of these plots we can define the *busy* period at Politecnico di Torino: 8 AM–6 PM, Monday–Friday. In the remaining of the paper, we report results averaged only on *busy* periods, since results outside these periods are far less interesting. Notice that an edge router, even a quite active one like the access router at Politecnico, never handles more than a hundred of flow openings per second.

The statistics we present are collected distinguishing between *clients* and *servers*, i.e., hosts that actively open a connection, and hosts that reply to the connection request, but also identifying *internal* and *external* hosts, i.e., hosts located in the campus LAN, and hosts located outside of it. Thus *incoming* and *outgoing* packets/flows are identified (see Fig. 1).

## 3. IP layer measurements

Most measurement campaigns in data networks concentrated on traffic volumes, on packet inter-arrival times, and on similar metrics. We avoid reporting this kind of results because they do not differ much from previously published ones, and because we think that, using the presented data processing tool, more interesting performance figures can be derived. We report only the most interesting statistics that can be gathered looking at the IP protocol header, referring the reader to [28] to discover further statistics.

Fig. 3 plots the distribution of the *hit count* for incoming traffic, i.e., the relative number of times in which the same source IP address was seen, either at the IP layer (top plot) or at the TCP layer (bottom plot), i.e., considering only packets with the SYN flag set. Each position in the horizontal axis corresponds to a different IP source address. IP addresses are sorted by decreasing occurrence rate. The log/log scale plot in the inside box draws the entire distribution, while the larger, linear scale plot magnifies the first 100 positions of the distribution. More than 200,000 different IP addresses were observed, with the top 10 sending about 5% of the packets, and 10% of the flows. It is interesting to notice that the distribution of the traffic is very similar during the three different time periods, but, looking at the top 100 IP addresses (not reported for privacy reasons), little correlation can be found: the most frequently contacted hosts are different, but the relative quantity of traffic they send is, surprisingly, the same. This confirms the difficulties in predicting the traffic pattern on the Internet. A further interesting feature is the similarity of the TCP flow and the IP packet distributions. The reason lies in the dominance of short flows in the overall traffic.

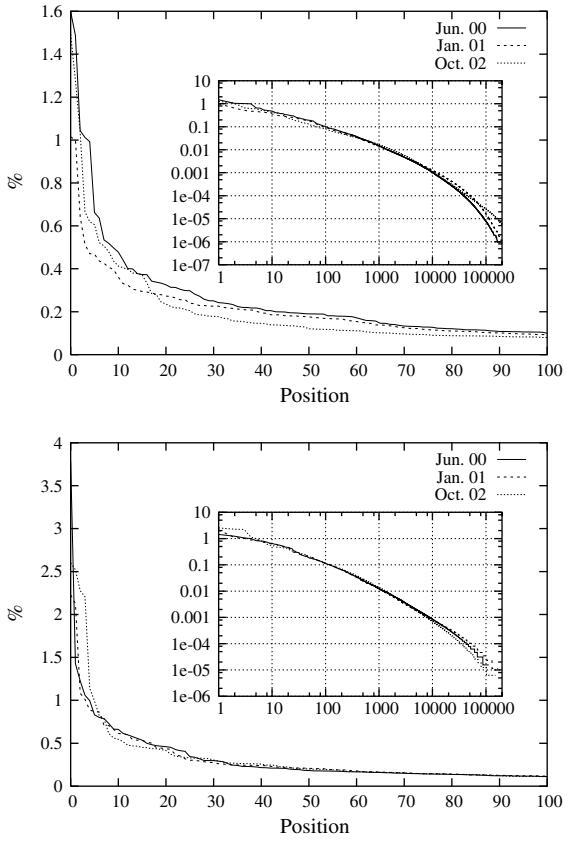


Fig. 3. Distribution of the incoming traffic vs the source IP address; the upper plot refers to the percentage of packets, while the bottom one refers to the percentage of flows.

A simple way to measure the *diameter* of the Internet, is to look at the number of hops between the client and the server. Fig. 4 reports the distribution of the IP Time To Live (TTL) field content, distinguishing between incoming and outgoing packets. For the outgoing traffic, the TTL distribution is concentrated on the initial values set by the different operating systems: 128 (Win 98/NT/2000/Xp), 64 (Linux, SunOs 5.8), 32 (Win 95), 60 (Digital OSF/1) being the most common. For the incoming traffic, instead, we can see very similar distribution at the left of each peak in the 'out' distribution, reflecting the number of routers traversed by packets before arriving at the measurement point. The zoomed plot in the box, shows that, supposing that the outside hosts set

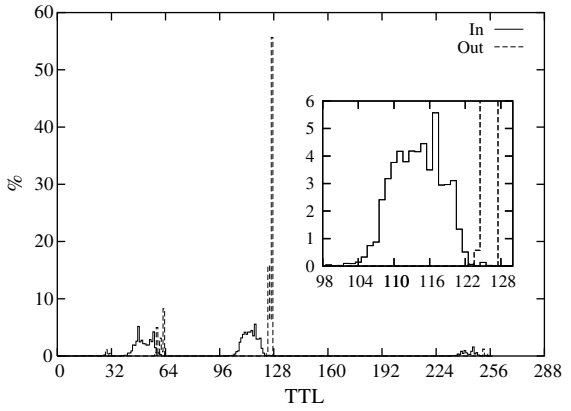


Fig. 4. Distribution of the TTL field value for outgoing and incoming packets; OCT.02 period.

the TTL to 128,[4] the number of hops traversed by packets is between 5 and 25 hops.[5] From this measurement, it is easy to see that the paths in the Internet, as seen from Politecnico, very rarely have more than 25–30 hops, and that the majority of them fall between 10 and 15 hops.

Fig. 5 reports the packets size distribution, averaged over the OCT.02 period. It confirms that packets are either very small, i.e., pure acknowledgment packets, or very large, i.e., full Ethernet MTUs, with a rather small percentage of 576-bytes-long minimum MTU. Moreover, since the majority of incoming packets are large, and the majority of outgoing packets are small (though hardly readable, outgoing packets 1500-bytes-long are roughly 10%), the network at Politecnico di Torino classifies as a "client" network, as we expected.

## 4. TCP layer measurements

The most interesting (and novel) performance figures derived with `Tstat` are at the TCP layer.

[4] While it is impossible to know the initial TTL value, the value 128 is the most probable one, as testified by the default TTL set by most operating systems.

[5] The minimum is intrinsically due to the topological position of the Politecnico gateway in the GARR network [33].

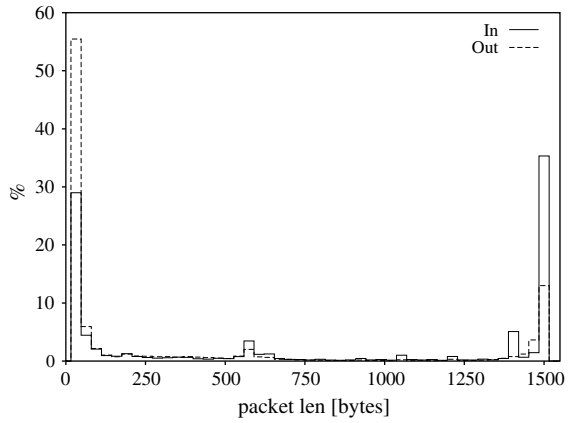


Fig. 5. Packet size distribution for incoming and outgoing packets; OCT.02 period.

The *TCP options* [34,35] negotiated during the three-way handshake, can significantly influence the TCP behavior, but how much are they now widespread and used on the Internet? Table 2 reports our findings, showing the percentage of clients that request an option in the "client" column; if the server positively replies, then the option is successfully negotiated and accounted in the "succ." column, which reports the percentage relative to the number of requests and not to the total number of flows; otherwise it will not be used. The "unset" percentage counts connections where no option was requested from either side. Finally, the "server" column reports the percentage of servers that, although they did not receive the option request, do sent an option acknowledgment, featuring a broken behavior. For example, looking at the SACK option, we see that about 30% of clients declared its SACK capabilities, that were accepted only by 16.9% of servers in JUN.00, 35.2% in JAN.01, and 71.8% in OCT.02. Note again that 0.1%, 0.3% and 0.001% of the servers send a positive acknowledgment to clients that did not request the SACK option, which is a mistake in the implementation of the SACK negotiation procedure.

Table 2
Percentage of hosts negotiating TCP options

| Option | Succ. | Client | Server | Unset |
|---|---|---|---|---|
| | JUN.00 | | | |
| SACK | 16.9 | 29.5 | 0.1 | 70.4 |
| WinScale | 56.7 | 19.2 | 1.3 | 79.5 |
| TimeStamp | 99.9 | 3.1 | 0.1 | 96.9 |
| | JAN.01 | | | |
| SACK | 35.2 | 32.9 | 0.3 | 66.7 |
| WinScale | 46.7 | 10.9 | 1.2 | 87.9 |
| TimeStamp | 99.9 | 4.5 | 0.0 | 95.5 |
| | OCT.02 | | | |
| SACK | 71.8 | 70.6 | 0.001 | 30.28 |
| WinScale | 82.7 | 18.6 | 0.0 | 81.33 |
| TimeStamp | 80.1 | 14.6 | 0.0 | 85.4 |

In general we can state that, while the SACK option is increasingly used, the use of the Window Scale and the TimeStamp options is still rare. Note also that servers positively reply to most of the TimeStamp option requests, which is, however, laterly used by less than 15% of clients.

Table 3 reports the most common *MSS negotiation*, during different time periods. It confirms the results in Fig. 5: the most common MSS is derived from the Ethernet MTU of 1500 bytes, followed by the default option, which is used by about 7% of clients and 16.5% of servers[6] connections in JAN.01, while it is practically negligible in OCT.02. Note that there is a non-negligible amount of hosts that advertise MSS = 512, which correspond to MTU = 552, which is smaller than the minimum that must be supported by all hosts [36].

### 4.1. TCP flow layer analysis

Flow measurements can be obtained when it is possible to correlate the TCP connections in both directions. This is true also for relatively "popular" measurements, such as the *flow size dimension* reported in Fig. 6, i.e., the byte-wise dimension of the payload transported on each half connection. The correlation is needed to properly decide the opening and closing of TCP connections, and to identify the appropriate sequence numbers. Fig. 6 reports 3 plots. The lower one shows the tail of the distribution in log–log scale, showing a clear linear trend, typical of heavy tailed distributions

[6] Declaring an MSS = 0 corresponds to request a minimum standard MSS of 536.

Table 3
Distribution of the declared MSS during the JAN.01 and OCT.02 periods

| OCT.02 | | | | JAN.01 | | | |
|---|---|---|---|---|---|---|---|
| Off. by client | | Off. by server | | Off. by client | | Off. by server | |
| MSS | % | MSS | % | MSS | % | MSS | % |
| 1460 | 91.5 | 1460 | 85.0 | 1460 | 88.6 | 1460 | 73.9 |
| 536 | 3.0 | 1380 | 10.7 | 536 | 6.9 | 0 | 13.4 |
| 1452 | 1.6 | 1360 | 1.0 | 512 | 2.2 | 512 | 5.2 |
| 1456 | 1.2 | 512 | 0.7 | 1456 | 0.8 | 536 | 3.1 |
| 1380 | 0.6 | 536 | 0.7 | 1380 | 0.3 | 1380 | 2.8 |

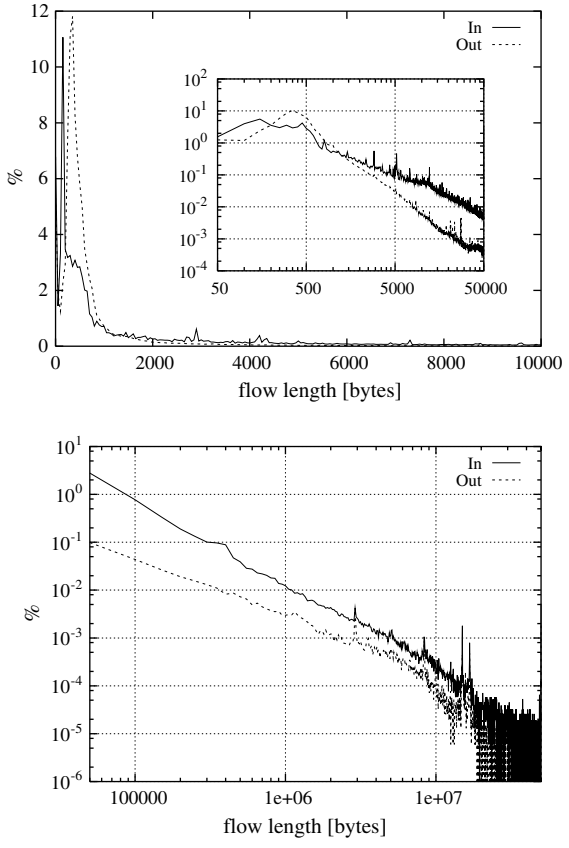


Fig. 6. Size distribution of incoming and outgoing flows; tail distribution in log–log scale (lower plot); zoom in linear and log–log scale of the portion near the origin (upper plots)—OCT.02.

[37]. The linear plot (upper, large one) shows a magnification near the origin, with the characteristic peak due to very short connections, typically of a few bytes. The inset in log–log scale shows the portion of the distribution where the mass is concentrated, and the linear decay begins. In particular, incoming flows shorter than 10 kbytes are 83% of the total analyzed flows, while outgoing flows shorter than 10 kbytes are 98.5%. Notice also that the dimension of incoming flows is consistently larger than that of outgoing flows. The figure refers to the OCT.02 trace; no meaningful differences are present when considering the other two time periods.

The *TCP port number* distribution, which directly translates in traffic generated by the most popular applications, is reported in Table 4, sorted by decreasing percentage for the JAN.01 and OCT.02 periods. Results are reported for each application in percentage of flows, segments (including signaling and control ones), and bytes (considering the payload only). The *application average flow size* on both directions of TCP flows is reported in Table 5 for JAN.01, which was selected because of the largest amount of non-filtered different services. These measurements take a different look at the problem; indeed, considering the JAN.01 portion of Table 4, the fact that the largest portion of the Internet traffic is Web-browsing is no news at all, and that FTP amounts to roughly 10% of the byte traffic, although the number of FTP flows are marginal, is also well known. Referring to Table 5, the different amount of data transferred by the applications in the client–server and server–client directions is instead not as well known, though not surprising. This asymmetry is much more evident when expressed in bytes than is segments, hinting to a large number of control segments (acknowledgments) sent

Table 4
Percentage of TCP traffic generated by common applications in flows, segments and bytes; JAN.01 and OCT.02 periods

| Service | Port | Flow % | Segments % | Bytes % |
|---|---|---|---|---|
| JAN.01 | | | | |
| HTTP | 80 | 81.48 | 62.78 | 61.27 |
| SMTP | 25 | 2.98 | 2.51 | 2.04 |
| HTTPS | 443 | 1.66 | 0.87 | 0.52 |
| POP | 110 | 1.26 | 0.93 | 0.42 |
| AUTH | 113 | 0.54 | 0.07 | 0.00 |
| FTP control | 21 | 0.54 | 0.54 | 0.50 |
| GNUTELLA | 6346 | 0.53 | 2.44 | 1.58 |
| FTP data | 20 | 0.51 | 6.04 | 9.46 |
| SSL-telnet | 992 | 0.45 | 0.13 | 0.00 |
| DNS | 53 | 0.31 | 0.03 | 0.01 |
| OCT.02 | | | | |
| HTTP | 80 | 89.3 | 60.19 | 62.84 |
| SMTP | 25 | 2.44 | 1.4 | 0.02 |
| POP | 110 | 2.391 | 0.01 | 0.42 |
| HTTPS | 443 | 1.47 | 1.2 | 0.1 |
| Proxy | 3128 | 0.56 | 0.001 | 0.3 |
| Proxy | 8080 | 0.51 | 0.004 | 0.15 |
| Sock | 1080 | 0.4 | 1.56 | 0.53 |
| FTP control | 21 | 0.2 | 0.002 | 0.1 |
| NNTP | 119 | 0.02 | 2.0 | 2.7 |

without data to piggyback them on. For example, a HTTP client sends to the server about 1 kbytes of data, and receives about 16 kbytes as reply. But more than 15 segments go from the client to the server, while less than 20 segments are required to transport the data from the server to the client.

Considering the OCT.02 data in Table 4, we observe that the introduction of the filtering rules on the firewall caused a modification on the traffic services spread. Indeed, Gnutella and other peer-to-peer traffic is not allowed anymore (at least on the standard service ports), and the FTP traffic is also affected, as the FTP data port has been blocked as well.

Fig. 7 confirms the intuition given by Table 5. The figure reports the index of *asymmetry* $\xi$ of connections, obtained as the ratio between the client–server and the total amount of transferred data:

$$\xi = \frac{\text{data}(C \to S)}{\text{data}(C \to S) + \text{data}(S \to C)},$$

$\xi$ is measured as either byte-wise net payload (upper plot), or segment-wise (bottom plot), which again includes all the signaling/control segments. The upper plot shows a clear trend to asymmetric connections, with many more bytes transferred from the server to the client ($\xi < 0.5$). If we consider the number of segments, instead, connections are almost perfectly symmetrical, as highlighted by the inset, magnifying the central portion of the distribution: over 25% of the connections are perfectly symmetrical, and the effect due to the delayed ACK implementation is not observed. This observation can have non-marginal effects in the design of routers, which, regardless of the asymmetry of the information in bytes, must always route and switch an almost equal number of packets, i.e., they must be able to perform the forwarding procedure at full speed considering a dominance of the smaller control packets.

Table 5
Distribution of the average amount of data per flow, in segments and bytes; JAN.01 period

| Service | Client to server | | Server to client | |
|---|---|---|---|---|
| | Segment | Byte | Segment | Byte |
| HTTP | 15.2 | 1189.9 | 19.5 | 15998.4 |
| SMTP | 21.2 | 15034.4 | 16.7 | 624.3 |
| HTTPS | 11.5 | 936.7 | 12.3 | 6255.8 |
| POP | 14.9 | 91.1 | 18.5 | 7489.0 |
| AUTH | 3.1 | 4.3 | 2.8 | 15.4 |
| FTP control | 23.7 | 11931.1 | 21.9 | 9254.3 |
| GNUTELLA | 101.5 | 23806.9 | 105.7 | 44393.9 |
| FTP data | 314.0 | 343921.3 | 223.5 | 82873.2 |
| SSL-telnet | 7.35 | 58.9 | 5.9 | 136.7 |
| DNS | 2.1 | 34.2 | 2.3 | 571.5 |

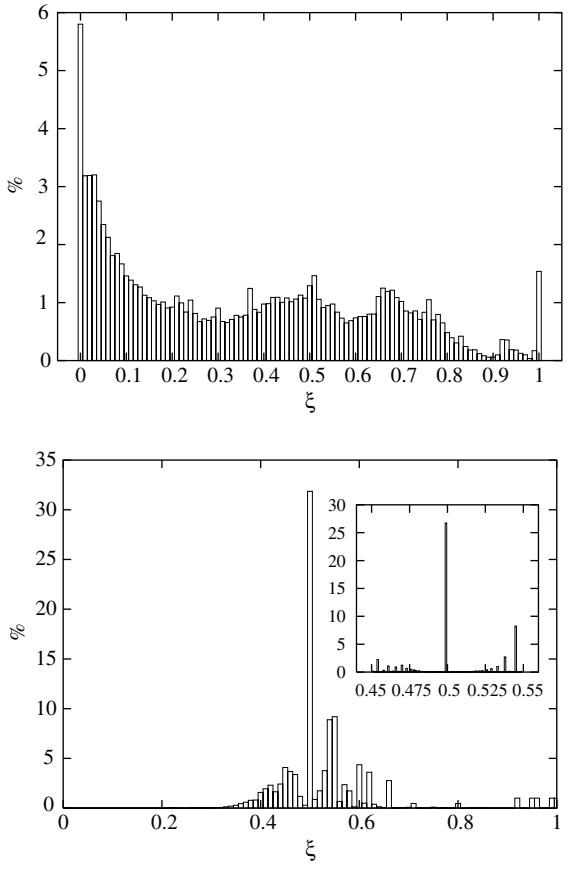


Fig. 7. Asymmetry distribution of connections expressed in bytes (upper plot) and segments (lower plot); OCT.02 period.

Around 1.8% of the flows exhibits on the contrary a total asymmetry in the client direction ($0.99 < \xi \leqslant 1$) in the byte-wise index. Those are due to connections where the data is sent by the host which opens the connection (thus being the client in the terminology used in this paper), i.e., data uploads.

Fig. 8 reports the distribution of the *connections completion time*, i.e., the time elapsed between the first segment in the three-way-handshake and the last segment that closes the connection in both directions (either an RST segment or a FIN in both directions). Obviously, this measurement depends upon the application, data size, path characteristics, network congestion, possible packet drops, etc. There are however several interesting features in the plot. First of all, most of the connections complete in just a few seconds: indeed, about 52% last less than 1 s, 78% less than 10 s, only 5% of the connections are not terminated after more than 70 s. Moreover, the completion time tends to be heavy tailed, which is possibly related to the heavy tailed flow size. Finally, spikes in the plot are due to application layer timeouts (e.g., 30 s to caches and Web servers timers, 60 s to persistent HTTP connection timers), which are generally not considered at all in traffic analysis. Interestingly, in the inset, it is also possible to observe the effects of TCP retransmission timeout suffered by connections whose first segment is dropped, which is set by default at 3 s. Indeed, there is a similarity between the shape of the distribution close to zero and the shape after 3 s, apart from the absolute value.

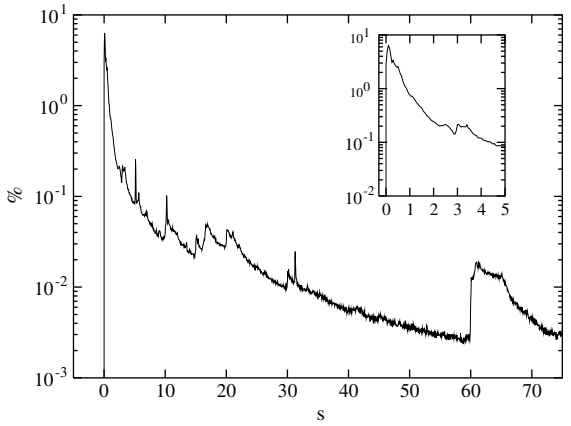


Fig. 8. Connection completion time distribution during the OCT.02 period.

### 4.2. Inferring TCP dynamics from measured data

We discuss here more sophisticated operations that can be done with `Tstat` on the raw measured data. By correlating flow-level and packet-level informations, it is possible to obtain details of TCP behavior and dynamics. When measurements are taken at an Internet edge node, and the *internal* part of the network hardly ever drops or reorders packets, `Tstat` can work as if co-located with transmitters for outgoing connections, and with receivers for incoming connections. Exploiting this opportunity, more insight can be obtained,

Table 6
Details on measurements of TCP three-way handshake and close procedures

| No. | Jun.00 | | | Jan.01 | | | Oct.02 | | |
|---|---|---|---|---|---|---|---|---|---|
| | SYN | FIN | RST | SYN | FIN | RST | SYN | FIN | RST |
| 0 | – | 12.54 | 85.68 | – | 12.99 | 92.18 | – | 13.53 | 94.36 |
| 1 | 98.56 | 86.65 | 14.32 | 95.14 | 84.55 | 7.82 | 97.95 | 84.01 | 5.36 |
| 2 | 1.22 | 0.76 | – | 4.13 | 2.32 | – | 1.8 | 2.36 | – |
| > | 0.22 | 0.05 | – | 0.73 | 0.14 | – | 0.24 | 0.20 | – |

without the need for using a 'specialized' TCP implementation that collects the measurements, as it is done for instance in [38,39].

Table 6 adds some insight on the connection opening and closing procedures of TCP. The measurement refers to outgoing packets only, i.e., to packets sent by hosts inside the Politecnico LAN. The first column reports the number of times a packet with the given flag set was observed. For each period, the other columns refer to the active open of connections (SYN), to the half-close procedure (FIN) or to the connection reset (RST).[7] Looking at the number of SYN segments required to open a connection, we observe that the number of times when more than one SYN segment is observed is not negligible (about 1.5% in Jun.00, 5% in Jan.01 and 2% in Oct.02). This may occur for many reasons: either the SYN or the SYN/ACK is lost; there is no route to the server; the server fails to respond in time to the open request. Firewall rules, traffic shaper policies, policing rules on the routers on the path, etc. may block packets. Which is the dominating reason is difficult to tell; however, the penalty paid for the retransmission of the SYN is very high due to the 3 s timeout. Notice also the increase in the number of duplicated SYNs from Jun.00 to Jan.01, when the average loss rate is decreased (see Table 7).

Most connections (>80%) finish using the half-close procedure (FIN), but the number of connection resets (RST) is far larger than the number of connections that can conceivably need to resort to it for network or host malfunctioning. This probably means that several applications/operating systems (most of all browsers) use the RST flag to terminate connections when, for instance, the user stops the page transfer or clicks on another hyperlink before the page transfer is over.

Fig. 9 plots the *advertised receiver window* (rwnd) observed looking at the TCP header. Both the *initial* (left hand plots) value and the value *averaged* over the whole flow duration (right hand plots) are reported, for the three measurement periods. Looking at the initial rwnd (i.e., the first announced rwnd value offered by the client) during Jun.00, about 50% of hosts advertise rwnd around 8 kbytes (6 segments considering the most common MSS), 9% advertise 16 kbytes, 30% uses about 32 kbytes, and the remaining 10% advertise values either close to 64 kbytes or sparse with no evident clustering. During Jan.01, a general, though not very large, increase in the initial rwnd is observed, in particular with some more hosts advertising windows around 64 kbytes. This trend is much more clear during Oct.02, in which the amount of connections advertising about 64 kbytes increases to more than 17%. Looking at the average rwnd advertised during the connection, we see that the distribution of the values changes very little, meaning that the receiver is rarely imposing a flow control over the sender; in other words, applications usually do not represent the performance bottleneck. Indeed, besides small modulations around the initial values, there are connections that increase the advertised window during the connection lifetime, which is a behavior we did not expect.[8] Notice that an 8 kbytes rwnd is

[7] An RST packet cannot be observed more than once, since when it is observed the connection is declared closed, and further packets are ignored by `Tstat`.

[8] Checking the TCP implementation of the most recent Microsoft OS confirms that the rwnd buffer value is tuned during the connection.

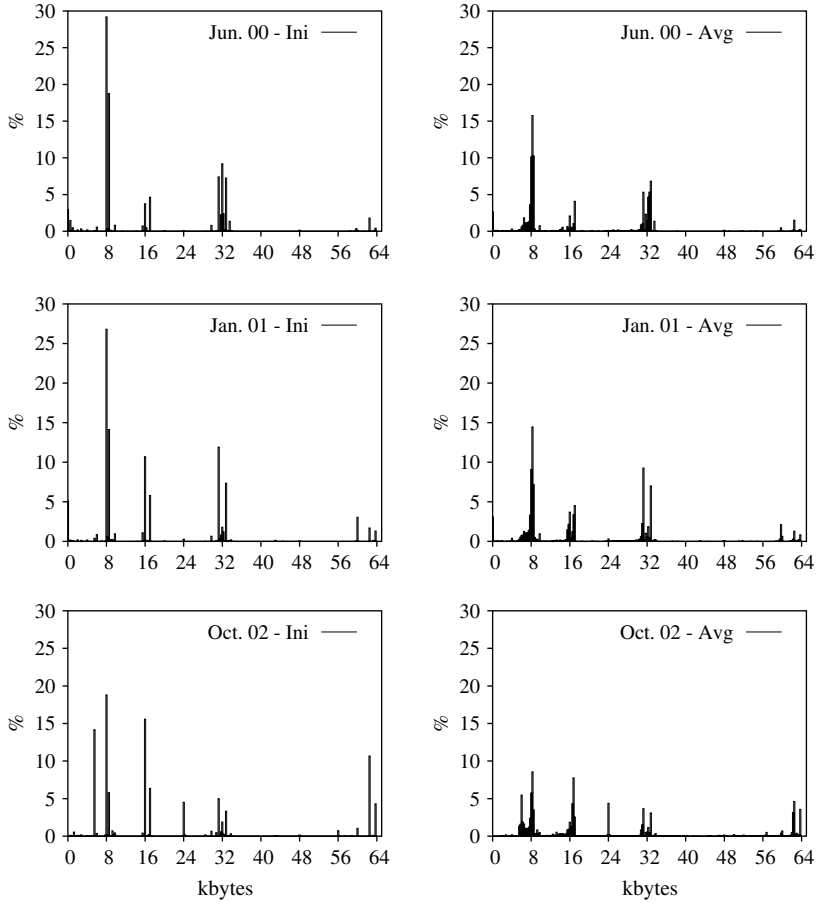


Fig. 9. Distribution of the "*rwnd*:" as advertised during handshake (Ini—left hand plots) and averaged during the whole connection (Avg—right hand plots).

surely a strong limitation on the maximum throughput that a connection can reach. For instance, with a 200 ms Round Trip Time, 8 kbytes correspond to about 40 kbytes/s.

In order to complete the picture, Fig. 10 plots the *estimated in-flight data* size for the *outgoing* flows, i.e., the bytes already sent from the source *inside* our LAN, and whose ACK is not yet received, evaluated looking at the sequence number and the corresponding acknowledgment number in the opposite direction. Given that measurements are collected very close to the sender, and that the rwnd is not a constraint, this is an estimate of the sender *congestion window*, at least when there is data to send. The discretization of the result clearly shows the effect of the segmentation

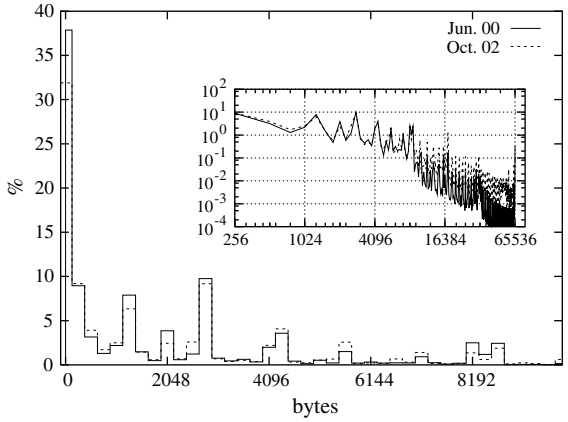


Fig. 10. TCP in-flight size distribution as seen from the measurement point for outgoing connections.

used by TCP. Moreover, since the flow length is frequently very short (see Fig. 6), the in-flight size is always concentrated on small values: more than 83% of the samples are indeed counted for in-flight sizes smaller than 4 kbytes. Finally, note that the increased network capacity in OCT.02 does not evidently affect the in-flight amount of data, and hence the estimated transmission window of TCP. This suggests that in the current Internet scenario, where most of the flows are very short, the main limitation to high throughputs seems to be the initial Slow Start phase of the TCP congestion control. Other effects of TCP on performance is delaying the data transfer both with the three-way handshake, and with unnecessary, very long timeouts when one of the first packets of the flow is dropped, an event that is due to traffic fluctuations, and not to congestion induced by the short flow itself.

To give more details, Fig. 11 reports for all time periods the *initial* (left) and *maximum* (right) in-flight size, expressed for each flow in number of the negotiated MSS. Only outgoing flows whose payload is larger than 10 MSS segments are considered, so that the TCP congestion control algorithm can operate. The initial in-flight size is the total amount of bytes sent by the sender before receiving the first acknowledgment, and should never be larger than either 4 MSS or 4380 bytes according to RFCs [40]. Interpreting the Initial in-flight size as a measurement of the first congestion window (cwnd) used by senders, we observe that most of current implementation (roughly 35%) set it to 2 segments (to avoid the penalty of the delayed ACK), while 15% set it to 1 segment, and about 5% uses 3 as initial cwnd value. But what is more interesting is the fact the a large part of the initial in-flight size is smaller than one full sized MSS (about 30% in JUN.00; 25% in JAN.01 and OCT.02), even if the total flow length is larger than 10 MSS. This is probably due to the influence of the application protocol, i.e., to the sequence of (initially small) HTTP request/replies that are carried over the same TCP connection in a persistent session.

The maximum in-flight size is instead a measurement of the maximum cwnd used by senders; it is subject to the congestion window growth rules (slow-start and congestion avoidance) and depends on the amount of data that have to be transferred. The interpretation of the results must take into account the fact that small flows (even if larger than 10 MSSs) terminate before the in-flight size/congestion window can grow, due to constraints imposed by the initial Slow Start phase. Observing the JUN.00 statistics, it can be noted that the maximum in-flight size is mostly limited to small values (2–4 segments). Only 7% of flows exploit a cwnd larger than 10 segments. Interestingly very few flows reach a maximum window size equal to 7, 8 or 9 segments, which is an indication that network conditions either limit the window to very small values or let it grow to fairly large ones. In JAN.01 and OCT.02, when the flow size distribution was almost unchanged (therefore allowing a fair comparison), the increased capacity of the access and US peering links allows the senders to exploit larger cwnd values (15% of flows reach cwnd >10 in both periods and all the distribution is shifted toward larger values). Moreover, recalling that the small rwnd set by a large percentage of clients (see Fig. 9) can imposes a hard limit (typically to about 6 segments) to the maximum in-flight size in the case of long flows, we can state that one of the limits to TCP performance is today represented by the rwnd imposed by the clients, which instead is in general always considered not to play an important role.

Loss analysis is quite difficult if the measurement tool does not have access either to Internet routers, where packets are lost, or to hosts, where losses are recovered. However, duplicated and out-of-sequence data can be used to infer the loss process. The *out-of-sequence burst* (OutB) size is the byte-wise length of contiguous data recorded out of sequence by `Tstat` in a flow. Similarly, the *duplicated data burst* (DupB) size is the byte-wise length of contiguous duplicated data recorded by `Tstat` in a flow. An OutB can be observed if either a packet reordering was performed in the network (e.g., as a consequence of a path change), or if packets were dropped along the path. A DupB, instead, is observed either when a packet is replicated in the network, or when a transmitter retransmits data already "seen" by `Tstat`, possibly because lost data is being recovered. Given

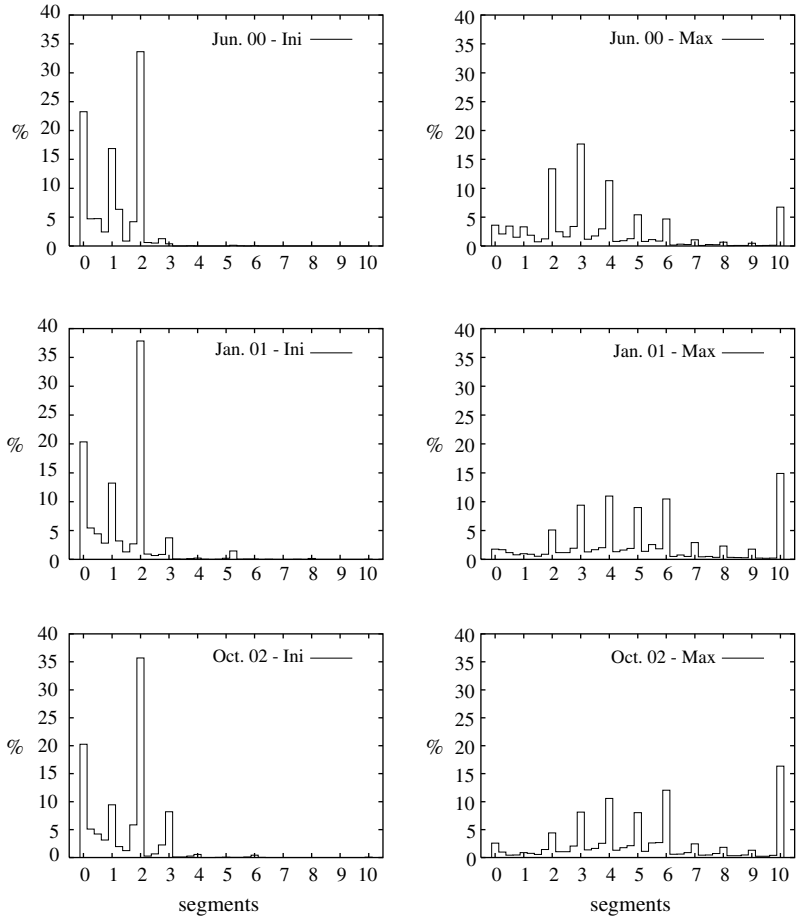


Fig. 11. Estimated initial and maximum TCP in-flight size, expressed in number of MSS-sized segments.

the measuring point, for outgoing flows `Tstat` sees all the retransmitted data as duplicated.

Table 7 reports the *probability* of observing OutB and DupB events, evaluated with respect to the number of observed segments and flows, i.e., the ratio between the total OutB (or DupB) recorded events and the number of packets or flows observed during in the same period of time.

Starting from OutB, we see that a small amount of OutB events are recorded on outgoing flows, thanks to the simple LAN topology, which, being a 100 Mbit/s switched Ethernet, rarely drops packets (recall that the access link capacity is either 4, 16, or 28 Mbit/s). On the contrary, the probability of observing an OutB is rather large for the incoming data: 3.4% of packets are received out of sequence in JUN.00, corresponding to a 43% of probability when related to the flows. Looking at

Table 7
OutB or DupB events rate, computed with respect to the number of packets and flows

| Period | vs Packet | | vs Flow | |
|---|---|---|---|---|
| | In % | Out % | In % | Out % |
| | P{OutB} | | | |
| JUN.00 | 3.44 | 0.07 | 43.41 | 0.43 |
| JAN.01 | 1.70 | 0.03 | 23.03 | 0.99 |
| OCT.02 | 1.61 | 0.20 | 22.12 | 8.90 |
| | P{DupB} | | | |
| JUN.00 | 1.45 | 1.47 | 18.27 | 18.59 |
| JAN.01 | 1.31 | 1.09 | 17.65 | 14.81 |
| OCT.02 | 0.73 | 0.93 | 13.12 | 13.01 |

the measurement referring to JAN.01 we observe a halved chance of OutB, 1.7% and 23% respectively, which remains constant during OCT.02. This is mainly due to the increased capacity of the access and the US peering links, which reduced the dropping probability; however, the loss probability remains relatively high.

Looking at the DupB probabilities, and recalling that the internal LAN can be considered as a sequential, drop-free environment, the duplicated bursts recorded on *outgoing* data can be ascribed to dropping events recovered by servers. A duplicate segment burst may also occur when the sender fails to selectively retransmit lost segments. On the contrary, *incoming* DupB events are due to retransmissions from external hosts of already received data. Neglecting the possibility that the network duplicates packets, this phenomenon can have two reasons: (i) the last ACK is lost, hence the packet is retransmitted after a timeout (recall that many connections are only 1 or 2 segments long); and (ii) senders fail to selectively retransmit lost segments, for example using a window larger than one MSS after a timeout.

To give the reader more details about the data involved in these events, Figs. 12 and 13 plot the distribution of the size of the OutB and DupB events respectively. The spikes are again an effect of the segmentation introduced by TCP, which concentrates the measurement on multiples of the most common MSSs (equal to around 1500 bytes). For example, looking at the size of OutB on the incoming data, we see that the peak of the gap distribution corresponding to one MSS comprises 60% of the total measured events. Assuming an OutB event to be the consequence of a packet drop, this is a strong indication that only one packet is dropped. Nonetheless, 10% of the OutB are accounting for two consecutive drops, 3% for three drops and even longer bursts are not negligible. The outgoing distribution is less interesting, given the smaller absolute chance of observing an outgoing OutB event, but the same overall behavior can be observed.

Very similar conclusions can be inferred from Fig. 13. In particular, the incoming DupB size is concentrated around 1500 bytes, or one MSS. On the contrary, the outgoing DupB size, related

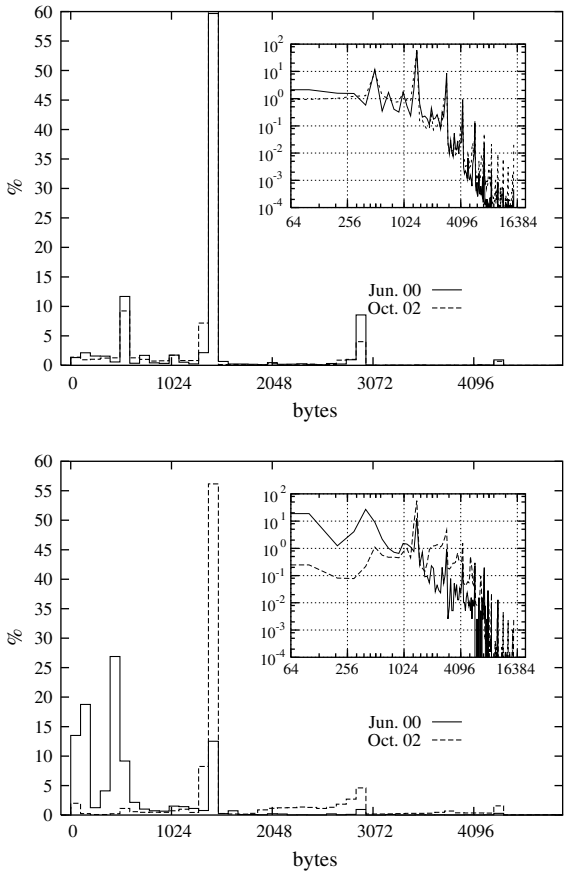


Fig. 12. Dimension distribution of out-of-sequence data bursts observed in incoming (top) and outgoing (bottom) connections in JUN.00 and OCT.02.

to retransmission due to segments lost in the external network, has a larger weight on smaller values. This is due to the fact that the majority of the outgoing flows are really small (see Fig. 6), and thus the payload size of TCP segments is often smaller than the MSS. Also in this case we can clearly see the spikes due to 1–3 and longer bursts of lost packets.

The insets in the two plots of Figs. 12 and 13 draw the same distribution in log–log scale, in order to be able to appreciate the tail behavior of the distribution: the trend of the spikes due to consecutive losses is linear, just like those of the flow dimension and the completion times. We do not have any explanation for this, apart from conjecturing that also the congestion periods of the

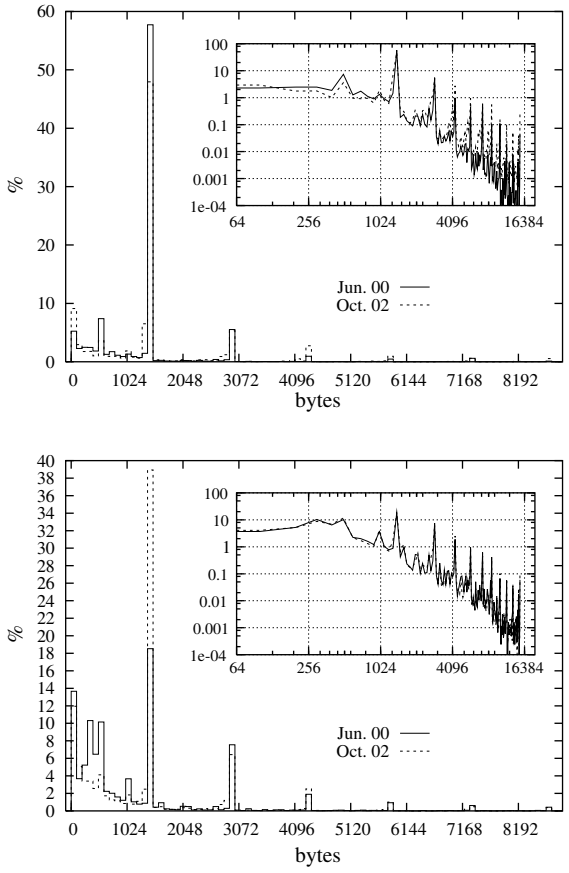


Fig. 13. Dimension distribution of duplicated data bursts observed in incoming (top) and outgoing (bottom) connections in JUN.00 and OCT.02.

Internet do show a heavy tail behavior, but this assertion surely requires more investigation.

Fig. 14 completes the picture on loss analysis, reporting the pdf of the number of lost packets given that the flow is at least 10 segments long, and that it was subject to losses. In general it can be observed that in OCT.02 bursts are shorter than in JUN.00. In both periods, however, the decay of the pdf tail is polynomial as already observed, with a non-marginal probability (2–3%) of losing more than 20 packets in a single flow (the last bin accumulates all the weight of the distribution tail). This behavior is due in part to the heavy tail distribution of the file size, but possibly also to other phenomena, since the number of flows with more than 20 segments is roughly around 10%.

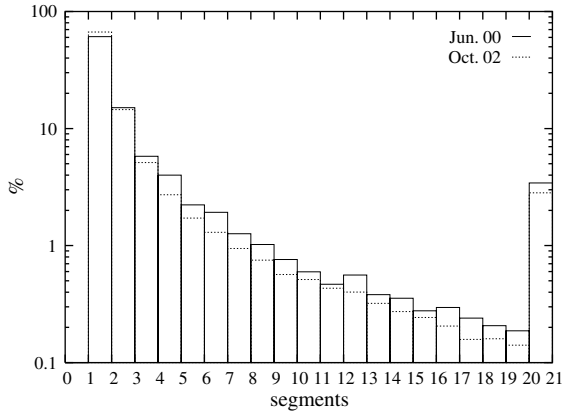


Fig. 14. Density probability function of flows suffering multiple packet losses, conditioned on the probability of losing at least one packet; only flows larger than 10 MSS are considered.

## 5. The tool: `Tstat`

The need to generate in an open tool statistical analysis from collected network traces was a major motivation to develop `Tstat`. Started as an evolution of `TCPtrace`, `Tstat` analyzes traces in real time, using common PC hardware, or starting from previously recorded traces in various dump formats, such as the one supported by the `libpcap` library [20], and the `DAG systems` [41] used to monitor gigabit link speed. It is written in standard C, and runs on Linux systems, but should run also on other UNIX-like operating systems.

As seen in the previous sections, besides common IP statistics, derived from the analysis of the IP header, `Tstat` also rebuilds each TCP connection status looking at TCP headers in the forward and backward packet flows.

Instead of dumping single measured data, for each measured quantity `Tstat` builds a histogram, estimating the distribution of that given quantity, while sequentially analyzing measurement data. Periodically, `Tstat` generates a dump of all collected histograms. A set of companion tools are available to generate both time plots, or aggregated plots over different time spans. Moreover, a Web interface [28] is available, which allows the user to browse all collected data, select the desired metric, and directly produce graphs, as well

as to retrieve the raw data that can then be later used for further analysis.

More than 80 different histogram types are available, comprising both IP and TCP statistics. They range from classic measurements directly available from the packet headers (e.g., percentage of TCP or UDP packets, packet length distribution, TCP port distribution), to advanced measurements, related to TCP (e.g., average congestion window, RTT estimates, out-of-sequence data, duplicated data). A complete log also keeps track of all analyzed TCP flows, and it is useful for post-processing purposes.

Since information is dumped only in aggregate form, the storage requirements of elaborated traces produced by `Tstat` is not particularly daunting. For example, the whole dataset produced from the OCT.02 period uses about 4 Gbytes of data, while the corresponding original packet level traces would use about 120 Gbytes of data.

A detailed, ''user manual-like'' description of the tool would be cumbersome in this context, thus we refer the reader to [28], where the tool is described at length.

The core features of `Tstat` are implemented in the trace analyzer. The basic assumption of the trace analyzer is that both the forward and backward packets are observed so that bidirectional TCP connections can be analyzed. If the measuring point is an access node to a campus network, such as the one depicted in Fig. 1, then the analyzer is fed by all the data coming in and going out the internal LANs. If, on the contrary, the measuring point is a generic backbone link, then there is no guarantee to observe both forward and backward packets.

A trace is an uninterrupted (bidirectional) flow of dumped packet headers, whatever it comes out to be. For instance, in our measuring campaigns we used traces of 6 h, to avoid the risk of excessive file sizes, but a trace can be anything from a short measurement on a slow link to weeks of live data collection on a fast interface pipelining the collected headers to the trace analyzer.

Fig. 15 shows the schematic block diagram of the trace analyzer, and in particular the logical blocks through which the program moves for each analyzed packet. Each packet is first analyzed by the IP module, which takes care of all the statistics at the packet layer. Then, if it is a UDP or TCP segment, all per-segment statistics are computed. In case it is a TCP segment, it goes through the TCP flow analyzer module.[9] This is the most complex part of `Tstat`. First, the module decides if the segment belongs to an already identified flow, using the classic five-tuple, i.e., IP source/destination addresses, IP protocol type and TCP source/destination ports. If not, then a new flow is identified only if the `SYN` flag is set, as `Tstat` processes only *complete* flows. Once the segment is successfully identified, all the state variables related to the flow are updated. Finally, in case the segment correctly closes a flows, all the flow-layer statistics are updated and the flow data structures are released.

A very efficient memory management core has been implemented, based on reuse techniques that minimize memory usage. Hashing functions and dynamically allocated lists are used to identify if the packet currently under analysis belongs to any of the tracked flows. When a flow closing sequence is observed, the flow is marked as *complete*, and all the TCP flow-layer measurements are computed. As mentioned above, a garbage collection procedure is used to free memory for unterminated flows: if no segments of a given flow are observed in between two garbage collections, the flow status memory used is freed, and the flow is considered abnormally terminated. TCP segments that belong to flows whose opening sequence is not recorded (because either they were started before running the trace analyzer, or early declared closed by the garbage collection procedure) are discarded and not analyzed.

To give an intuition of the speed achieved by `Tstat`, using an off-the-shelf 1GHz computer with 1 Gbyte of RAM, a 6 h-long trace from a 24 Mbit/s link is precessed in about 15 min, with a total memory footprint of only 50 Mbytes. To process a 3 h trace recorded on a OC48 backbone link (link utilization was about 100Mbps), Tstat used up to 900 Mbytes of RAM in about 1 h of

[9] UDP ''flows'' are currently not identified nor processed. This is left for future work.

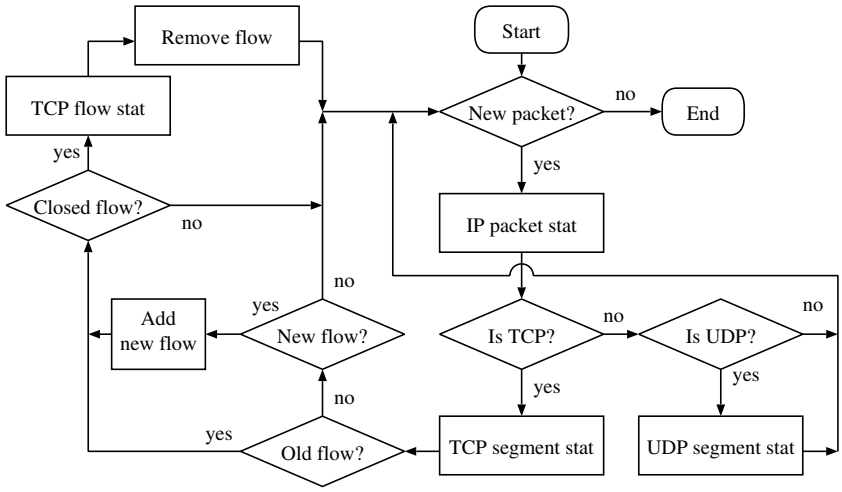


Fig. 15. Block diagram of the trace analyzer.

CPU time. In this scenario, only 13% of the observed flows were bidirectional, so that the majority of tracked flows were stored in memory until the garbage collector procedure destroyed them, as a correct TCP closing procedure could not be observed.

`Tstat` has been designed in a very modular way, so that it is easy to integrate modules for new measurements. For example, we are currently considering the extension of `Tstat` to analyze also the RTP protocol, adding an "RTP segment stat" module, so as to analyze real-time, multimedia traffic. The integration of new functionalities has to be directly written in the C programming language. While this forces the user to understand the `Tstat` internal databases, it does not impose any limitation on the functionalities that can be implemented, while allowing the very efficient implementation which is required by real-time measurements.

## 6. Conclusions

This paper presented `Tstat`, an open tool for Internet traffic data collection and statistical elaboration. The tool offers roughly 80 different types of plots and measurements, ranging from the simple amount of observed traffic, to complex reconstructions of the flow evolution.

The major novelty of the tool is its capability of correlating the outgoing and incoming flows at a single edge router, thus inferring the performance and behavior of individual bidirectional flows observed during the measurement period.

The advantages offered by this usually not available capability where shown by discussing some statistical analysis performed on data collected at the ingress router of Politecnico di Torino. The presented results offer a deep insight in the behavior of both the IP and the TCP protocols, highlighting several characteristics of the traffic that, to the best of our knowledge, were not previously observed with non-intrusive measurements on 'normal' traffic, but were only obtained by injecting ad-hoc flows in the Internet, or observed in simulations.

An important contribution of the paper, besides the presentation of `Tstat`, is the definition of innovative ways of computing interesting performance figures from easily available data traces.

We plan to improve the tool, enlarging the number of possible statistical elaborations that can be done on the data. In particular we are working on the possibility of inferring the behavior of TCP taking into account the different phases and algorithms (slow start, congestion avoidance, fast recovery, etc.) that characterize it.

Finally, the analysis of additional data collected at different edge and core routers is under way, in

order to generate enough statistics to support Internet modeling and planning efforts.

## Acknowledgments

The authors wish to thank the CeSIT network administrators of our campus network for the support offered in collecting traces and allowing us to have access to our campus access point. We would also like to thank the people of the international research community who downloaded `Tstat` and gave us valuable feedbacks that helped in developing the tool and in removing bugs. This work was supported by the Italian Ministry for Education and Scientific Research through the TANGO Project.

## References

[1] V. Paxson, S. Floyd, Wide-area traffic: the failure of poisson modeling, IEEE/ACM Transactions on Networking 3 (3) (1995) 226–244.
[2] M.E. Crovella, A. Bestavros, Self similarity in world wide Web traffic: evidence and possible causes, IEEE/ACM Transactions on Networking 5 (6) (1997) 835–846.
[3] V. Paxson, End-to-end routing behavior in the Internet, IEEE/ACM Transactions on Networking 5 (5) (1997) 601–615.
[4] R. Caceres, P. Danzig, S. Jamin, D. Mitzel, Characteristics of wide-area TCP/IP conversations, in: ACM SIGCOMM, Zurich, Switzerland, 1991, pp. 101–112.
[5] P. Danzig, S. Jamin, tcplib: A library of TCP internetwork traffic characteristics, 1991.
[6] P. Danzig, S. Jamin, R. Caceres, D. Mitzel, D. Mestrin, An empirical workload model for driving wide-area TCP/IP network simulations, Internetworking: Research and Experience 3 (1) (1992) 1–26.
[7] V. Paxons, Empirically derived analytic models of wide-area TCP connections, IEEE/ACM Transactions on Networking 2 (1994) 316–336.
[8] W.E. Leland, M.S. Taqqu, W. Willinger, V. Wilson, On the self-similar nature of ethernet traffic (extended version), IEEE/ACM Transactions on Networking 2 (1) (1994) 1–15.
[9] B.M. Duska, D. Marwood, M.J. Feeley, The measured access characteristics of World-Wide-Web client proxy caches, in: USENIX Symposium on Internet Technologies and Systems, Monterey, CA, 1997, pp. 23–36.
[10] L. Fan, P. Cao, J. Almeida, A. Broder, Summary cache: a scalable wide-area Web cache sharing protocol, in: ACM SIGCOMM, Vancouver, BC, Canada, 1998, pp. 254–265.
[11] A. Feldmann, R. Caceres, F. Douglis, G. Glass, M. Rabinovich, Performance of Web proxy caching in heterogeneous bandwidth environments, in: IEEE INFOCOM, New York, 1999, pp. 107–116.
[12] B. Mah, An empirical model of HTTP network traffic, in: IEEE INFOCOM, Kobe, Japan, 1997, pp. 592–604.
[13] H. Balakrishnan, M. Stemm, S. Seshan, V. Padmanabhan, R.H. Katz, TCP behavior of a busy internet server: analysis and solutions, in: IEEE INFOCOM, San Francisco, CA, 1998, pp. 252–262.
[14] W.S. Cleveland, D. LinD, X. Sun, IP packet generation: statistical models for TCP start times based on connection-rate superposition, in: ACM SIGMETRICS, Santa Clara, CA, 2000, pp. 166–177.
[15] F.D. Smith, F. Hernandez, K. Jeffay, D. Ott, What TCP/IP protocol headers can tell us about the Web, in: ACM SIGMETRICS, Cambridge, MA, 2001, pp. 245–256.
[16] S. Jaiswal, G. Iannaccone, C. Diot, J. Kurose, D. Towsley, Measurement and classification of out-of-sequence packets in a Tier-1 IP backbone, in: IEEE INFOCOM, San Francisco, CA, 2003, pp. 1199–1209.
[17] Z.-L. Zhang, V. Ribeiro, S. Moon, C. Diot., Small-time Scaling behaviors of internet backbone traffic: an empirical study, in: IEEE INFOCOM, 2003, pp. 1826–1836.
[18] L. Li, M. Thottan, B. Yao, S. Paul, Distributed network monitoring with bounded link utilization in IP networks, in: IEEE INFOCOM, 2003, pp. 1189–1198.
[19] C.D.H. Jiang, Passive estimation of TCP round-trip times, ACM SIGCOMM Computer Communication Review 32 (3) (2002) 75–88.
[20] S. McCanne, C. Leres, V. Jacobson, libpcap, Available from <http://www.tcpdump.org>, 2001.
[21] S. McCanne, C. Leres, V. Jacobson, Tcpdump, Available from <http://www.tcpdump.org>, 2001.
[22] V. Paxson, Automated packet trace analysis of TCP implementations, in: ACM SIGCOMM, Cannes, France, 1997, pp. 167–179.
[23] S. Ostermann, Tcptrace, Available from <http://www.tcptrace.org>, version 5.2, 2001.
[24] L. Deri, S. Suin, Effective traffic measurement using ntop, IEEE Communications Magazine 38 (2000) 138–143.
[25] V. Paxson, Bro: a system for detecting network intruders in real-time, Computer Networks 31 (23–24) (1999) 2435–2463.
[26] N. Measurement, N.A. Group, Towards a systemic understanding of the Internet organism: a framework for the creation of a network analysis infrastructure. Overview, Available from <http://moat.nlanr.net/>, 1998.
[27] S.A.T. Laboratories, IP monitoring project (ipmon), Available from <http://ipmon.sprintlabs.com/>, 2002.
[28] M. Mellia, A. Carpani, R. Lo Cigno, `Tstat` Web page, Available from <http://tstat.tlc.polito.it/>, 2004.
[29] G. Iannaccone, C. Diot, I. Graham, N. McKeown, Monitoring very high speed links, in: ACM Internet Measurement Workshop, San Francisco, CA, 2001, pp. 267–271.
[30] J. Postel, RFC 791: Internet Protocol, September 1981.
[31] J. Postel, RFC 793: Transmission Control Protocol, September 1981.

[32] M. Allman, V. Paxson, W. Stevens, RFC 2581: TCP Congestion Control, 1999.
[33] GARR, GARR—The Italian Academic and Research Network, Available from <http://www.garr.it/garrb-home-engl.shtml>, 2001.
[34] V. Jacobson, R. Braden, D. Borman, RFC 1323: TCP Extensions for High Performance, May 1992.
[35] M. Mathis, J. Madhavi, S. Floyd, A. Romanow, RFC 2018: TCP Selective Acknowledgment Options, October 1996.
[36] J. Postel, RFC 897: The TCP Maximum Segment Size and Related Topics, November 1983.
[37] W. Willinger, V. Paxson, M.S. Taqqu, Self-similarity and heavy tails: structural modeling of network traffic, in: R.J. Adler, R.E. Feldman, M.S. Taqqu (Eds.), A Practical Guide to Heavy Tails: Statistical Techniques and Applications, Birkhauser, Boston, 1998, pp. 27–53.
[38] S. Savage, Sting: a TCP-based network measurement tool, in: USENIX Symposium on Internet Technologies and Systems, Boulder, CO, 1999, pp. 71–79.
[39] J. Padhye, S. Floyd, On inferring TCP behavior, in: ACM SIGCOMM, San Diego, CA, 2001, pp. 287–298.
[40] M. Allman, S. Floyd, C. Partridge, RFC 2414: Increasing TCP's Initial Window, September 1998.
[41] J. Cleary, S. Donnelly, I. Graham, A. McGregor, M. Pearson, Design principles for accurate passive measurement, in: Proc PAM2000: The First Passive and Active Measurement Workshop, Hamilton, New Zealand, 2000, pp. 1–7.

**Marco Mellia** was born in Torino, Italy, in 1971. He received his degree in Electronic Engineering in 1997, and a Ph.D. in Telecommunications Engineering in 2001, both from Politecnico di Torino. From March to October 1999 he was with the CS Department at Carnegie Mellon University as visiting scholar. Since April 2001, he is with Electronics Department of Politecnico di Torino as an Assistant Professor. His research interests are in the fields of All-Optical Networks, Traffic measurement and modeling, QoS Routing algorithms.

**Renato Lo Cigno** was born in Ivrea, Italy in 1963. He received a Dr. Ing. degree in Electronic Engineering from Politecnico di Torino in 1988. Since then he has been with the telecommunication research group of the Electronics Department of Politecnico di Torino, first as Research Engineer and then as an Assistant Professor. Starting November 2002 he holds a position as an Associate Professor at the University of Trento, Italy. From June 1998 to February 1999, he was at the CS Department at UCLA as Visiting Scholar under grant CNR 203.15.8. He is co-author of about 80 journal and conference papers in the area of communication networks and systems. He was in the program committee of IEEE Globecom and IEEE ICNP.

His current research interests are in performance evaluation of wired and wireless networks, modeling and simulation techniques, QoS management including routing and congestion control.

**Fabio Neri** is full Professor at the Electronics Department of Politecnico di Torino, Turin, Italy. He received his Dr. Ing. and Ph.D. degrees in Electrical Engineering from Politecnico di Torino in 1981 and 1987, respectively. He has been a visiting researcher at the Computer Science Department of the University of California at Los Angeles, CA, at the Optical Data Networks Research Department of Bell Laboratories/Lucent Technologies in Holmdel, NJ, and at British Telecom Advanced Communication Research in Ipswich, UK.

His research interests are in the fields of performance evaluation of communication networks, high-speed and all-optical networks, packet switching architectures, discrete event simulation, and queuing theory. He has served several IEEE conferences, inluding Globecom and Infocom, and was the Technical Program Co-Chair of the 1999 IEEE Workshop on Local and Metropolitan Area Networks.