Measuring IP and TCP behavior with Tstat

M. Mellia, R. Lo Cigno, F. Neri
Dipartimento di Elettronica – Politecnico di Torino
Corso Duca degli Abruzzi, 24 – I-10129 Torino, Italy
e-mail: {mellia,locigno,neri}@mail.tlc.polito.it

Abstract

Field measurements have always been the starting point for networks planning and dimensioning; however, their statistical analysis beyond simple traffic volume estimation is not so common. In this paper we present and discuss `Tstat`, a new tool for the collection and statistical analysis of TCP/IP traffic, that is able to infer TCP connection status from traces. Besides briefly discussing its use, we present part of the performance figures that can be obtained and the insight that such figures can give on TCP/IP protocols and the Internet.

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 flows. While standard performance measures, such as flow dimensions, traffic distribution, etc., remain at the base of traffic evaluation, more sophisticated indices, like the out-of-order probability and gap dimension in TCP connections, obtained through data correlation between the incoming and outgoing traffic, give reliable estimates of the network performance also from the user perspective.

`Tstat` derives about 80 different performance indices at both the IP and TCP level, allowing a very deep insight in the network performance. Several of these indices are discussed on traffic measurements performed for more than 2 months on the access link of our Institution.

I. Traffic Measurements in the Internet

Planning and dimensioning telecom networks was always based on traffic measurements, upon which estimates and models are built to be used with the appropriate mathematical tools. 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, specially in the Internet, where the TCP/IP client-server communication paradigm introduces correlation both in space and time among traffic relations.

This work was supported by the Italian Ministry for Education and Scientific Research through the PLANET-IP Project.
A Preliminary version of this paper will be presented at IEEE Globecom 2002.

While a large part of this difficulty lies in the failure of traditional modeling paradigms [1], [2], there are also several key points 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 meaning only when the forward and backward traffic are jointly analyzed, otherwise half of the story goes unwritten, and should be wearingly inferred (with high risk of making mistakes). This problem makes measuring inherently difficult. If measures are taken on the network edge, where the outgoing and incoming flows are necessarily coupled, correlating the flows can be easy, but it can prove harder in the backbone, where the peering contracts among providers often make the forward and backward routes disjoint [3].

Second, data traffic must be characterized to a higher level of detail than circuit switched voice traffic. A telephone call is represented as a constant bit rate (CBR) connection with a given duration, hence characterizing the arrival and duration processes of voice calls is sufficient for telephone traffic. In a data network like the Internet, where the network level is connectionless and there are no connection admission control functionalities, even the notion of connection becomes quite fuzzy. 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 concatenation of correlated packets, as in a TCP connection;
- *session*: whenever a user is accessing the Internet, maybe just reading a web page without transmitting or receiving anything.

The layered structure of the TCP/IP protocol suite requires the analysis of traffic at least at the IP (packet), TCP/UDP (flow), and Application/User–behavior (session) levels, in order to have a picture of the traffic clear enough to allow the interpretation of data.

Starting from the works of Danzig [4], [5], [6], and Paxons and [1], [7] the interest in data collection, measurement and analysis to characterize either the network or the users 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 level, measurements show that the arrival process of packets is complex and correlated. We only recall the first works [2], [8], but many others exist that either analyze the traffic characteristics or try to offer models to represent it. These works concentrate on packets, disregarding the interaction with higher layer protocols.

At the application level, after the birth of the Web, many works analyzed traces, typically log files of Web or proxy servers [9], [10], [11]. Traffic analysis at this level gives insight into applications and user behavior, but sometimes fails to correlate high-level phenomena with network characteristics. Other projects analyzed the same high-level phenomena using traffic traces, captured form large campus networks, like the work in [12], where the authors characterize the HTTP protocol by using large traces collected at the university campus in Berkeley. Similarly in [13] the authors present data collected from a large Olympic Games server in 1996, whose findings are helpful to understand TCP behavior, like loss recovery efficiency and ACK compression. In [14], authors analyzed more than 23 millions of HTTP connections, and derived a model for the connection inter-arrival time. More recently, the authors of [15] analyze and derive models for the Web traffic, starting from the TCP/IP header analysis.

To the best of our knowledge, however, there are no measurement tools, or measurement based works, that characterize the traffic at the three levels identified above, and in particular at the flow level, rebuilding the status of individual TCP connections, correlating the observations with those at the packet and application levels. This paper concentrates on flow level analysis.

A. Available tools

There are several tools available to derive "network measures." 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 [16], [17], that allows an analysis at different protocol levels.

For example, `tcpanaly` is a tool for automatically analyzing a TCP implementation's behavior by in-

specting packet traces of the TCP's activity. Another interesting tool is `tcptrace` [18], which is able to rebuild a TCP connection status from 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. At IP level, `ntop` [19] is able to collect statistics, enabling users to track relevant network activities, including traffic characterization, network utilization, network protocol usage.

Indeed, many more tools exist than those we considered, but they are either *active* tools, like for example the classic `ping` or `traceroute` utilities, or commercial tools, whose source code is not available. Hence they are hard to use for a research project, where the implementation of innovative ideas is the focal point.

All of these tools, however, have a narrowband scope, and lack an organized data base for collecting traces, as well as post-elaboration and statistical analysis techniques. Finally, they do not have a user interface allowing easy operation.

B. The Role of `Tstat`

In light of the above discussion, a major motivation to develop a new tool, named `Tstat` [20] was the lack of automatic tools able to produce statistical data from collected network traces. `Tstat`, starting from standard software libraries, is able to offer network managers and researchers important information about classic and novel performance indices and statistical data about Internet traffic.

Besides common IP statistics, derived from the analysis of the IP header, `Tstat` is also able to rebuild the status of each TCP connection, looking at the TCP header in the forward and backward packet flows. The TCP flow analysis allows the derivation of novel statistics, such as, for example, the congestion window size, out-of-sequence segments, duplicated segments, etc. The detailed description of `Tstat` is deferred to Sect. V (after we have commented the most interesting results obtained with it) and, most of all, to the `Tstat` home page [20].

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

- A new tool for gathering and elaborating Internet measurements. The tool is easy-to-use, it is free and available to the community [20].

- The identification and description of several interesting performance measures that can be obtained from the Internet with the above tool, discussing their relation with the network behavior and its possible evolution. Such measures are both at the IP and TCP level.

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

In the remaining of the paper we assume that the reader is familiar with the Internet terminology, which can be found for example in [21], [22], [23]. First, in Sects. II, III and IV, we discuss several performance indices, most of them innovative, applied to measurements done on the access router of our Institution. In Sects. V, some details about `Tstat` are given, concentrating on those that we deem most interesting. Finally, Sect. VI ends the paper.

II. 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 our Institution. During the roughly two 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 our Institution are available on `Tstat` web site [20] 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 [24], the Italian and European Research network. Within the Politecnico Campus LAN, there are approximately 7,000 access points; most of them are clients, but several servers are regularly accessed from the outside. The backbone of our Campus LAN is based on a switched Fast Ethernet infrastructure. 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 the earliest one and one of the latest periods to highlight `Tstat` features:

- 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 and the corresponding US peering was 45 Mbit/s;

• 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 and the corresponding US peering was 622 Mbit/s.

The two 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. This link plays a key role, since large part of the traffic comes from U.S. sites and all of it cross this link. In the remaining of the paper, every time we observe a meaningful difference in the measures during different periods, we report both of them. Otherwise, we report only the most recent one.

Table I summarizes the traces reporting the total number of packet and flows traced and the share of protocols above IP. Not surprisingly, the larger part of the traffic is transported using TCP, being the UDP traffic about 5%; other protocols are practically negligible. The number of TCP flows is larger than 4 and 7 millions in the two periods.

Fig. 2 analyzes the traffic load during a week in JAN.01. The upper plot reports the traffic volume divided per IP payload type normalized versus 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 constant around 0.2% of the link capacity. With a quick analysis of these plots we can define the *busy* period at our Institution: 8 AM–6 PM, Monday to 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 as Politecnico access is, never handles more than a few tens 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 our campus LAN, and hosts located outside it, with respect to the edge node where measurements are collected. Thus *incoming* and *outgoing* packets/flows are identified (see Fig. 1).

III. IP Level Measures

Most measurements campaigns in data networks concentrated on the traffic volumes, packet inter-arrival times and similar measures. We avoid reporting similar results because they do not differ much from previously published ones, and because we think that, using the data elaboration tool presented, other and more interesting performance figures can be derived, which allow a deeper insight in the Internet behavior. Thus, we report only the most interesting statistics that can be gathered looking at the IP protocol header, referring the reader to [20] to analyze all the figures he might be interested into.

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 level (top plot) or at the TCP flow level (bottom plot), i.e., considering only packets with the SYN flag set. 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 hosts were observed, with the top 10 sending about 5% of the packets, and 10% of the flows. It is interesting to note that the distribution of the traffic is very similar during the two different time periods, but, looking at the top 100 IP addresses (not reported for privacy), 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 in the Internet. A further interesting feature is the similarity of the TCP flow and the IP packet distribution. The reason lies probably in the dominance of short, web-browsing flows in the overall traffic.

A simple way to measure the *span* of the Internet, is to look at the number of hops between the client and the server. Fig. 4 reports the distribution of the 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), 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 the TTL to 128, the number of hops traversed by packets is between 5 and 25 hops[1]. From this measure 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 one week in JAN.01. It confirms that either packets are very small i.e., pure control packets, or very large, i.e., Ethernet full size MTU, 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 large are roughly 10%), the Politecnico network classifies as a "client" network, as we expected.

IV. TCP LEVEL MEASURES

`Tstat` offers the most interesting (and novel) performance figures correlating packets and flows at the TCP level.

The *TCP options* [25], [26] negotiated during the three way handshake, can significantly influence TCP behavior, but just how much they are now widespread and used in the Internet? Table II reports our findings, showing the percentage of clients that requests an option in the "client" column; if the server positively replies, then the option is successfully negotiated and accounted in the "succ." column; otherwise it will not be used. The "unset" percentage counts connections where the option was not present, 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 acknowledge, featuring a "weird" behavior. For example, looking at the SACK option, we see that about 30% of clients declared its SACK capabilities, that were accepted only by 5% of servers during JUN.00 and 11.6% during JAN.01. Note again that 0.1% and 0.3% of the servers in the two periods send a positive acknowledgment to clients that did not request the option.

In general we can state that TCP options are rarely used, and, while the SACK option is increasingly used, the use of the Window Scale and the Timestamp options is either constant or decreasing. Note also that servers positively reply to practically all the Timestamp option requests, which is however used by less than 5% of clients.

[1] The minimum is intrinsically due to the topological position of the Politecnico gateway [24].

Table III reports the most common *MSS negotiation*, which confirms the results in Fig. 5: the most common one is derived from the Ethernet MTU of 1500 bytes, followed by the default option, which is used by about 7% of client and 18% of server[2] connections. 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.

A. TCP flow level analysis

Flow-level measures require to correlate the TCP connections in both directions for their derivation. This is true also for relatively "popular" measures, 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 [27]. 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 *TCP port number* distribution, which directly translates in traffic generated by the most popular applications, is reported in Table IV, sorted by decreasing percentage. 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 V. These measures take a different look at the problem; indeed, 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, altough the number of FTP flows are marginal, is also well known. 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 asym-

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

metry is much more evident when expressed in bytes than is segments, hinting to a large number of control segments (acknowledgments) sent without data to piggyback them on. For example, a HTTP client sends to the server about 1 kbyte 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 from the server to the client are required to transport the data.

Fig. 7 confirms the intuition given by Table V. The figure reports the index of *asymmetry* ξ of connections, obtained as the ratio between the client-server and the total amount of transferred data:

$$\xi = \frac{data(C \rightarrow S)}{data(C \rightarrow S) + data(S \rightarrow C)}.$$

ξ 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: 25% of the connections are perfectly symmetrical, and no effect is observed due to the delayed ACK implementation. 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 yet considering the smaller control packets.

Around 3% of the flows exhibits on the contrary a total asymmetry in the client direction ($0.99 < \xi \leq 1$) in the byte-wise index. Those are due to FTP-data connection, were the data that is sent by the host which opens the connection (thus being the client in the terminology used in this paper).

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 measure 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 second, 78% less than 10 second, only 5% of the connections are not terminated after more than 70 s, where the histogram collects all the remaining flows. 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 level timeouts (e.g., 15 seconds correspond to a timer in the AUTH protocol, 30 seconds to caches and web servers timers) which are generally not considered at all in traffic analysis. Interestingly, in the inset, it is also possible to observe the retransmission timeout suffered by TCP flows whose first segment is dropped, which is set by default at 3 seconds. Indeed, there is a close similitude between the shape of the distribution close to zero and the shape after 3 seconds, apart from the absolute value.

B. Inferring TCP Dynamics from Measured Data

We discuss here more sophisticated elaborations that can be done with `Tstat` on the raw measured data. Correlating flow level informations with packet level ones, it is possible to obtain details of TCP behavior and dynamics. When measurements are taken at the Internet edge node, and the *internal* part of the network does not drop or reorder packets, `Tstat` can work as if co-located with transmitters for outgoing connections, and with receivers for incoming connections. Exploiting this capability, more insight can be obtained, without the need for using a 'specialized' TCP implementation that collects the measurements itself, like for instance in [28], [29].

Table VI adds some insight on the connection opening and closing procedures of TCP. The measure refers to outgoing packets only, i.e., for 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)[3]. Looking at the number of SYN segments required to open a connection, we observe that the number of times more than one packet is requires is not negligible (about 1.5% in JUN.00 and 5% in JAN.01). This may occour for three reasons: i) either the SYN or the SYN/ACK is lost; ii) there is no route to the server; and iii) the server fails to respond in time to the open request. Which is the dominating reason is impossible to

[3] 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`.

tell; however, the penalty paid for the retransmission of the SYN is very high due to the 3 s timeout. Notice also that the duplicated SYNs probability increases from JUN.00 to JAN.01, when the average loss rate is decreased (see Table VII).

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* (lower plots) value and the value *averaged* over the whole flow duration (upper plots) are reported, for both measurement periods. Looking at the initial rwnd during JUN.00, about 50% of hosts advertise rwnd around 8 kbytes (6 segments considering the most common MTU), 9% advertise 16 kbytes, 30% uses about 32 kbytes, and the remaining 10% advertise values either close to 64 kbytes or sparse withouth any possible clustering. These values are obtained summing together all the bins around 8, 16, and 32. During JAN.01, a general, though not very large, increase in the initial rwnd is observed, specially with some more hosts advertising windows around 64 kbytes. Notice that an 8 kbytes rwnd is surely a strong limitation on the maximum throughput a connection can reach. For instance, with a 200 ms Round Trip Time, 8 kbytes correspond to about 40 kbytes/s. 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 never represent the 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.

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 discrete-like result clearly shows the effect of the segmentation used by TCP. Moreover, being the flow length 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 JAN.01 does not apparently 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, and the main limitation to high throughputs seems to be the receiver buffer, the dynamic, sliding window implementation in TCP rarely comes into play. The only effect of TCP on the 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 the traffic fluctuations, and not to congestion induced by the short flow itself.

To give more details, Fig. 11 reports the *Initial* (top) and *Maximum* (bottom) in-flight size for both time periods. Only outgoing flows whose payload is larger than 10 MSS segments are considered so that TCP congestion control algorithm can operate. Interpreting the initial in-flight size as a measure of the first congestion window (cwnd) used by senders, we observe that most of current implementation (>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 really interesting is the fact the a large part of the initial in-flight size is smaller than 1 full sized MSS (about 30% in JUN.00 25% in JAN.01), even if the total flow length is larger than 10 MSS. The distribution in JUN.00 and JAN.01 are similar, with a larger percentage of flows that start with an initial cwnd equal to 2 in JAN.01.

The maximum in-flight size is instead a measure of the maximum cwnd used by senders. Looking at the JUN.00 statistics, we observe that a relatively small cwnd (2,3,4 segments) is generally reached, and only 7% of flows can exploit a cwnd larger that 10, with very few flows using cwnd equal to 7,8,9. In JAN.01 instead, the increased capacity of the access and US peering links allows the senders to exploit larger cwnd values (for example, 15% of flows reach cwnd>10 in this second scenario). But the small rwnd set by clients (see Fig. 9) imposes an hard limit to about 6 segment to the maximum cwnd. This again underlines that one of the bottleneck to the TCP performance is the rwnd imposed by the clients, which instead is in general always

considered to not play an important role.

The analysis of losses is quite difficult if the measurement tool does not have access either to the 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 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 after a packet drop when the transmitter recovers the lost data, and may retransmit data already "seen" by `Tstat`. Given the measuring point, for outgoing flows `Tstat` sees all the retransmitted data as duplicated.

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

Starting from OutB, we see that practically no OutB events are recorded on the outgoing flows, thanks to the simple LAN topology, which, being a 100 Mbit/s switched Ethernet, rarely drop packets (recall that the access link capacity is either 4 Mbit/s or 16 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 the measure referring to JAN.01 we observe a halved chance of OutB, 1.7% and 23% respectively. 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 very high.

Looking at the DupB probabilities, and recalling that the internal LAN can be considered as a sequential, drop-free environment, the duplicated burst recorded on the *outgoing* data can be ascribed to dropping events recovered by the servers. Thus a measure of the dropping probability is derived. On the contrary, *incoming* DupB events are due to retransmissions from external hosts of already received data. Neglecting the possibil-

ity 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); ii) senders that 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 due to the segmentation effect introduced by TCP, which concentrate the measure 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 moda of the gap distribution is equal to one MSS, corresponding to a 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 2 consecutive drops, 3% for 3 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 conclusion 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, that is due to retransmission following 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 not equal to MSS. Also in this case we can clearly see the spikes due to 1, 2, 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 Internet do show a heavy tail behavior, but this assertion surely requires more investigation.

Fig. 14, complete 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

JAN.01 bursts are shorter than in JUN.00. In both periods, however, the decay of the pdf tail is polinomial 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%.

V. THE TOOL: TSTAT

The lack of automatic tools able to generate statistical data from collected network traces was a major motivation to develop `Tstat`. Started as an evolution of `TCPtrace`, it is able to analyze traces in real time, using common PC hardware, or to start from previously recorded traces in various dump formats, such as the one supported by the `libpcap` library [16], and the `DAG systems` [30] used to monitor Gigabit speed links. It is written in standard C, and runs on Linux systems, but should run also on other Unix versions.

Besides common IP statistics, derived from the analysis of the IP header, `Tstat` is also able to rebuild 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, collecting the distribution of that given quantity. Periodically, it produces a dump of all the histograms it collected. A set of companion tools are available to produce both time plots, or aggregated plots over different time spans. Moreover, a Web interface [20] is available, which allows the user to travel among all collected data, select the desired quantity, 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 measures directly available from the packet headers (e.g., percentage of TCP or UDP packets, packet length distribution, TCP port distribution, ...), to advanced measures, related to TCP (e.g., average congestion window, RTT estimates, out-of-sequence data, duplicated data, ...). A complete log also keeps track of all the TCP flow analyzed, and is it useful for post-processing purposes.

The detailed, “user manual–like” description of the tool would be cumbersome in this context, thus we refer the reader to [20], 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 the one depicted in Figure 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 the uninterrupted (bidirectional) flow of dumped packet headers, whatever it comes out to be. For instance, in our measuring campaigns we used traces of 6 hours, 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.

Figure 15 shows the schematic block diagram of the trace analyzer, and in particular the logical block 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 packet layer. Then, if it is a UDP or TCP segment, all the per-segment statistics are performed. In case it is a TCP segment, it goes through the TCP flow analyzer module. This is the most complex part of `Tstat`. First, the module tries to find if it the segment belongs to an already identified flow, using the classic n-tuple. 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 statistics related to the flow analysis are performed. Finally, in case the segment correctly closes a flows, all the flow-level statistics are updated.

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-level measures are computed. 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 flow status memory used is freed, and the flow is considered

abnormally terminated. In this paper, we set the garbage collector timer to 15 minutes, which was shown to guarantee very good flow identification [31].

TCP segments that belong to flows whose opening sequence is not recorded (because either were started before running the trace analyzer, or early declared closed by the garbage procedure) are discarded and not analyzed.

To give an intuition of the speed `Tstat` can guarantee, using a off-the-shelf 1GHz computer with 1Gb of RAM, a 6 hours long peak trace from a 24 Mbit/s link is precessed in about 15 minutes, with a total memory footprint of only 50MB. To process a 3 hours trace recorded on a OC48 backbone link (link utilization was about 100Mbps), Tstat used up to 900MB of RAM in about 1 hour of CPU time. In this second scenario, only 13% of observed flows were bidirectional, so that the majority of tracked flows are stored in memory until the garbage collector procedure destroys them, as a correct TCP closing procedure cannot be observed.

`Tstat` has been designed in a very modular way, so that it is easy to integrate modules for new measures. For example, we are currently considering the extension of `Tstat` to analyze also the RTP protocol with a "RTP segment stat," so as to be able to analyze real-time, multimedia traffic.

VI. Conclusions

This paper presented a novel tool for Internet traffic data collection and its statistical elaboration. The tool offers roughly 80 different types of plots and measurement figures, 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.

Exploiting this capability, we have presented and discussed some statistical analysis performed on data collected at the ingress router of our Institution. The results presented 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 never observed on 'normal' traffic, but were only generated by injecting ad-hoc flows in the Internet or observed in simulations.

We plan to improve the tool, enlarging the number of possible statistical elaborations that can be done on the data, but most of all we plan to analyze more data collected at different edge and core routers, in order to generate enough statistics to support Internet modeling and planning efforts.

ACKNOWLEDGMENT

The authors wish to thank the Ce.S.I.T. network administrator 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 who downloaded `Tstat` and gave us a valuable feedback that helped in trimming down bugs.

REFERENCES

[1] V. Paxson and S. Floyd, "Wide-Area Traffic: The Failure of Poisson Modeling," *IEEE/ACM Transactions on Networking*, vol. 3, no. 3, pp. 226–244, June 1995.

[2] M. E. Crovella and A. Bestavros, "Self Similarity in World Wide Web Traffic: Evidence and Possible Causes," *IEEE/ACM Transactions on Networking*, vol. 5, no. 6, pp. 835–846, 1997.

[3] V. Paxson, "End-to-end routing behavior in the Internet," *IEEE/ACM Transactions on Networking*, vol. 5, no. 5, pp. 601–615, 1997.

[4] R. Caceres, P. Danzig, S. Jamin, and D. Mitzel, "Characteristics of Wide-Area TCP/IP Conversations," *ACM SIGCOMM '91*, 1991.

[5] P. Danzig and S. Jamin, "tcplib: A library of TCP Internetwork Traffic Characteristics," *USC Technical report*, 1991.

[6] P. Danzig, S. Jamin, R. Caceres, D. Mitzel, and D. Mestrin, "An Empirical Workload Model for Driving Wide-Area TCP/IP Network Simulations," *Internetworking: Research and Experience*, vol. 3, no. 1, pp. 1–26, 1992.

[7] V. Paxons, "Empirically Derived Analytic Models of Wide-Area TCP Connections," *IEEE/ACM Transactions on Networking*, vol. 2, pp. 316–336, August 1994.

[8] W. E. Leland, M. S. Taqqu, W. Willinger, and V. Wilson, "On the self-similar nature of ethernet traffic (extended version)," *IEEE/ACM transactions on networking*, vol. 2, no. 1, pp. 1–15, 1994.

[9] B. M. Duska, D. Marwood, and M. J. Feeley, "The Measured Access Characteristics of World-Wide-Web Client Proxy Caches," *USENIX Symposium on Internet Technologies and Systems*, pp. 23–36, December 1997.

[10] L. Fan, P. Cao, J. Almeida, and A. Broder, "Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol," *ACM SIGCOMM '98*, pp. 254–265, 1998.

[11] A. Feldmann, R. Caceres, F. Douglis, G. Glass, and M. Rabinovich, "Performance of Web Proxy Caching in Heterogeneous Bandwidth Environments," *IEEE INFOCOM '99*, pp. 107–116, 1999.

[12] B. Mah, "An Empirical Model of HTTP Network Traffic," *IEEE INFOCOM '97*, April 1997.

[13] H. Balakrishnan, M. Stemm, S. Seshan, V. Padmanabhan, and R. H. Katz, "TCP Behavior of a Busy Internet Server: Analysis and Solutions," *IEEE INFOCOM '98*, pp. 252–262, March 1998.

[14] W. S. Cleveland, D. LinD, and X. Sun, "IP Packet Generation: Statistical Models for TCP Start Times Based on Connection-Rate Superposition," *ACM SIGMETRICS 2000*, pp. 166–177, June 2000.

[15] F. Donelson Smith, F. Hernandez, K. Jeffay, and D. Ott, "What tcp/ip protocol headers can tell us about the web," *ACM SIGMETRICS '01*, pp. 245–256, June 2001.

[16] S. McCanne, C. Leres, and V. Jacobson, "libpcap," http://www.tcpdump.org, 2001.

[17] S. McCanne, C. Leres, and V. Jacobson, "Tcpdump," http://www.tcpdump.org, 2001.

[18] S. Ostermann, "tcptrace," 2001, Version 5.2.

[19] L. Deri and S. Suin, "Effective traffic measurement using ntop," *IEEE Communications Magazine*, vol. 38, pp. 138–143, May 2000.

[20] M. Mellia, A. Carpani, and R. Lo Cigno, "`Tstat` web page," http://tstat.tlc.polito.it/, 2001.

[21] J. Postel, "RFC 791: Internet Protocol," Sept. 1981.

[22] J. Postel, "RFC 793: Transmission control protocol," Sept. 1981.

[23] M. Allman, V. Paxson, and W. Stevens, "RFC 2581: TCP Congestion Control," 1999.

[24] GARR, "GARR - The Italian Academic and Research Network," http://www.garr.it/garr-b-home-engl.shtml, 2001.

[25] V. Jacobson, R. Braden, and D. Borman, "RFC 1323: TCP extensions for high performance," May 1992.

[26] M. Mathis, J. Madhavi, S. Floyd, and A. Romanow, "RFC 2018: TCP Selective Acknowledgment Options," October 1996.

[27] W. Willinger, V. Paxson, and M. S. Taqqu, "Self-similarity and Heavy Tails: Structural Modeling of Network Traffic," *A Practical Guide to Heavy Tails: Statistical Techniques and Applications, R. J. Adler, R. E. Feldman and M. S. Taqqu, editors*, 1998.

[28] S. Savage, "Sting: A tcp-based network measurement tool," *USENIX Symposium on Internet Technologies and Systems, Boulder, CO, USA*, pp. 71–79, October 1999.

[29] J. Padhye and S. Floyd, "On inferring TCP behavior," *ACM SIGCOMM 2001, San Diego, CA, USA*, pp. 287–298, August 2001.

[30] J. Cleary, S. Donnelly, I. Graham, A. McGregor, and M. Pearson, "Design principles for accurate passive measurement," *Proc PAM2000: The First Passive and Active Measurement Workshop, Hamilton, New Zealand*, pp. 1–7, April 2000.

[31] G. Iannaccone, C. Diot, I. Graham, and N. McKeown, "Monitoring very high speed links," *ACM Internet Measurement Workshop, San Francisco*, November 2001.

TABLE I

SUMMARY OF THE ANALYZED TRACES

Period	Pkts	Protocol share [%]			Flows
	[10^6]	other	UDP	TCP	[10^6]
JUN.00	242.7	0.75	5.96	93.29	4.48
JAN.01	401.8	0.59	4.31	95.10	7.06

TABLE II

TCP OPTIONS NEGOTIATED

Option	succ.	client	server	unset
	JUN.00			
SACK	5.0	29.5	0.1	70.4
WinScale	10.9	19.2	1.3	79.5
TimeStamp	3.1	3.1	0.1	96.9
	JAN.01			
SACK	11.6	32.9	0.3	66.7
WinScale	5.1	10.9	1.2	87.9
TimeStamp	4.5	4.5	0.0	95.5

TABLE III

MSS DISTRIBUTION – JAN.01

Declared by Client		Declared by server	
MSS	%	MSS	%
1460	88.6%	1460	73.9%
536	6.9%	0	13.4%
512	2.2%	512	5.2%
1456	0.8%	536	3.1%
1380	0.3%	1380	2.8%

TABLE IV

PERCENTAGE OF TCP TRAFFIC GENERATED BY COMMON APPLICATIONS IN FLOWS, SEGMENTS AND BYTES – JAN.01

Service	Port	flow %	segm. %	bytes %
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

TABLE V

AVERAGE AMOUNT OF DATA PER FLOW, IN SEGMENTS AND BYTES – JAN.01

Service	Average			
	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

TABLE VI

DETAILS ON MEASURES OF TCP THREE-WAY HANDSHAKE AND CLOSE PROCEDURES

	JUN.00			JAN.01		
#	SYN	FIN	RST	SYN	FIN	RST
0	—	12.54	85.68	—	12.99	92.18
1	98.56	86.65	14.32	95.14	84.55	7.82
2	1.22	0.76	—	4.13	2.32	—
> 2	0.22	0.05	—	0.73	0.14	—

TABLE VII

OUTB OR DUPB EVENTS RATE, COMPUTED WITH RESPECT TO THE NUMBER OF PACKETS AND FLOWS

Period	P{OutB}			
	vs Pkt		vs flow	
	in %	out %	in %	out %
JUN.00	3.44	0.07	43.41	0.43
JAN.01	1.70	0.03	23.03	0.99

Period	P{DupB}			
	vs Pkt		vs flow	
	in %	out %	in %	out %
JUN.00	1.45	1.47	18.27	18.59
JAN.01	1.31	1.09	17.65	14.81

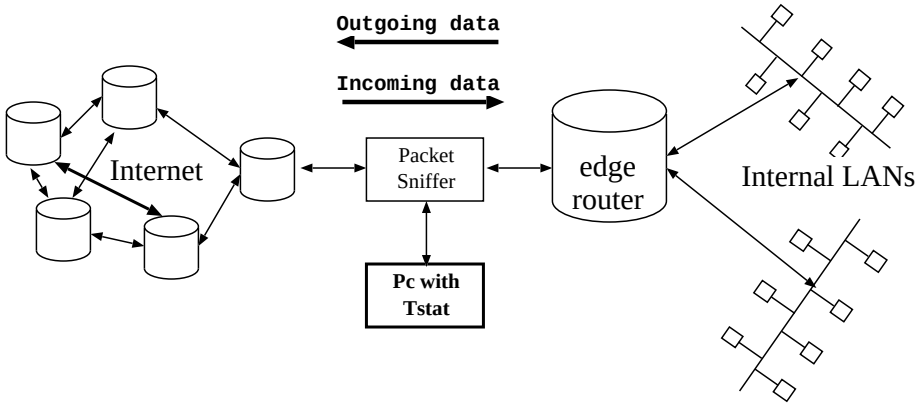

Fig. 1. The measuring setup at the Edge Router of Politecnico di Torino

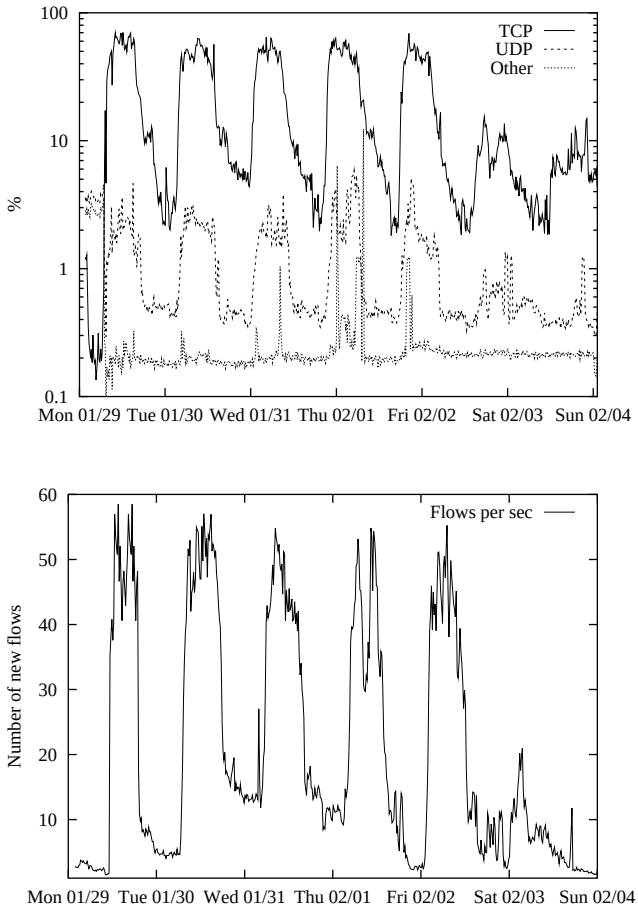

Fig. 2. Incoming IP payload normalized volume (top) and TCP flows arrival rate (bottom) during JAN.01

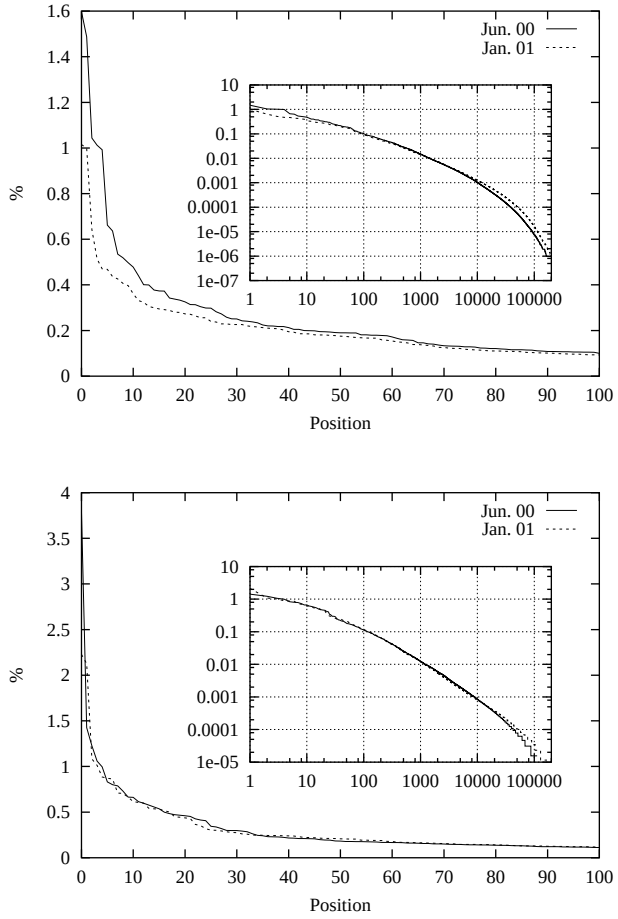

Fig. 3. Distribution of the incoming traffic versus the source IP address; the upper plot refers to percentage of packets, while the bottom one refers to the percentage of flows

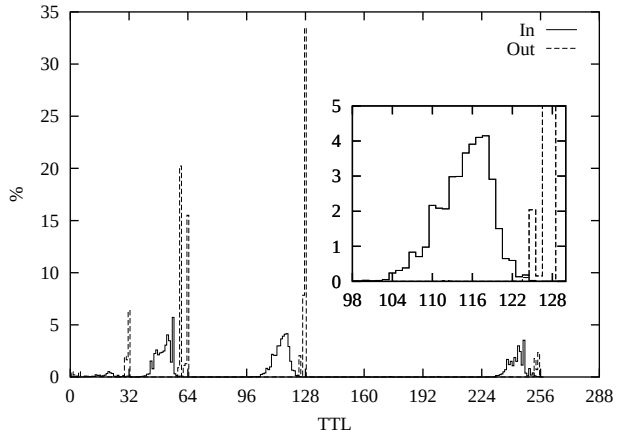

Fig. 4. Distribution of the TTL field value for outgoing and incoming packets – JAN.01

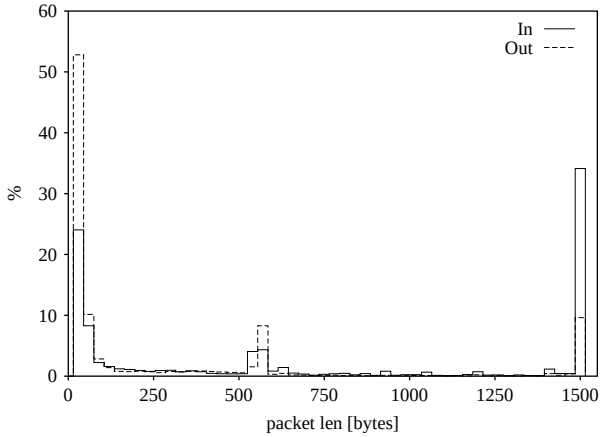

Fig. 5. Packet size distribution for incoming and outgoing packets – JAN.01

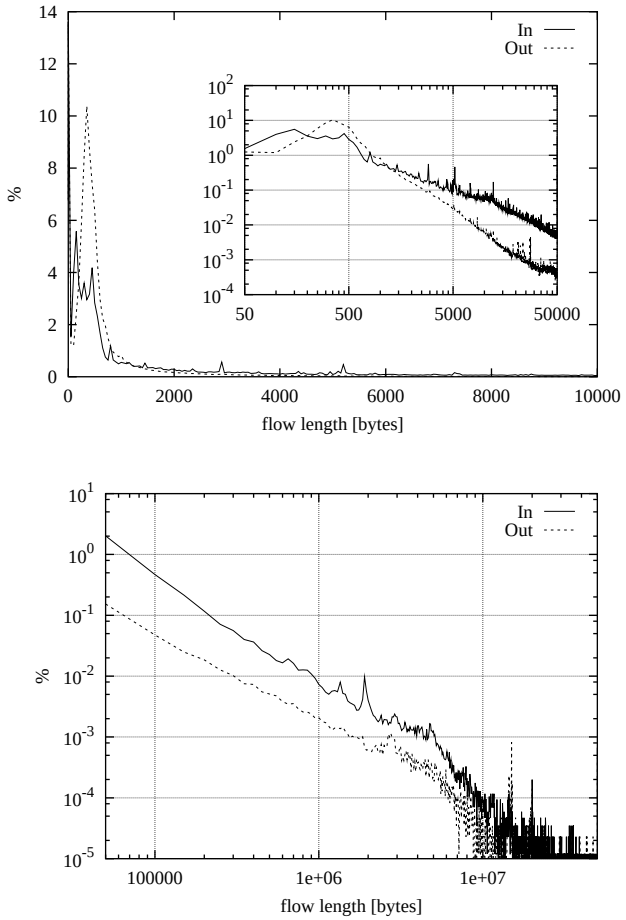

Fig. 6. Size distribution of incoming and outgoing flows; tail distribution in log-log scale (lower plot); zoom in linear and log-log scale of the portion near the origin (upper plots) – JAN.01

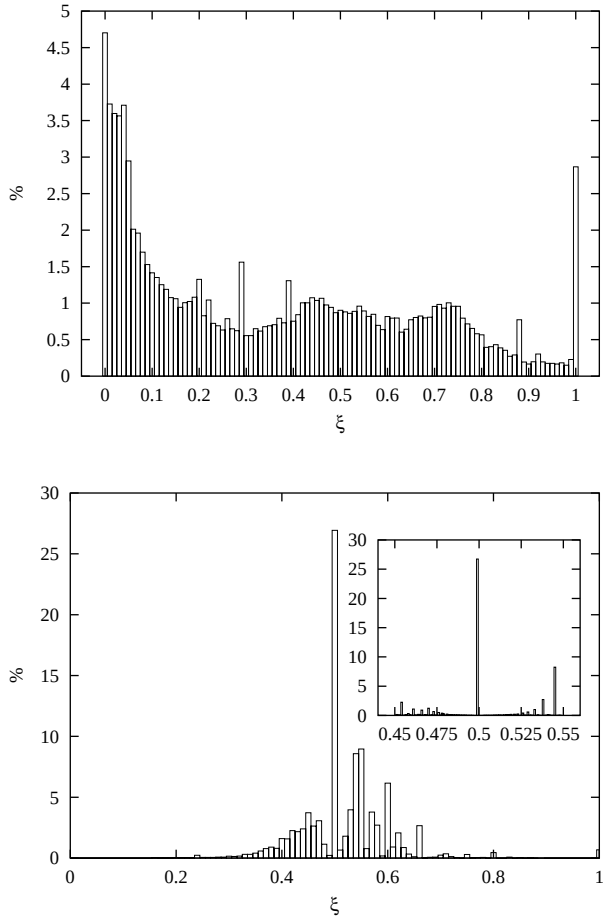

Fig. 7. Asymmetry distribution of connections expressed in bytes (upper plot) and segments (lower plot) – JAN.01

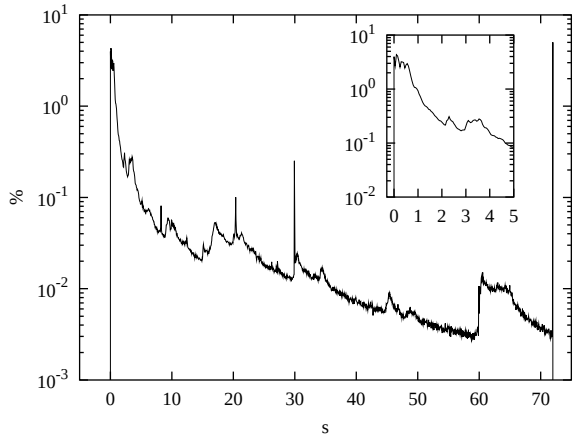

Fig. 8. Distribution of the connections completion time – JAN.01

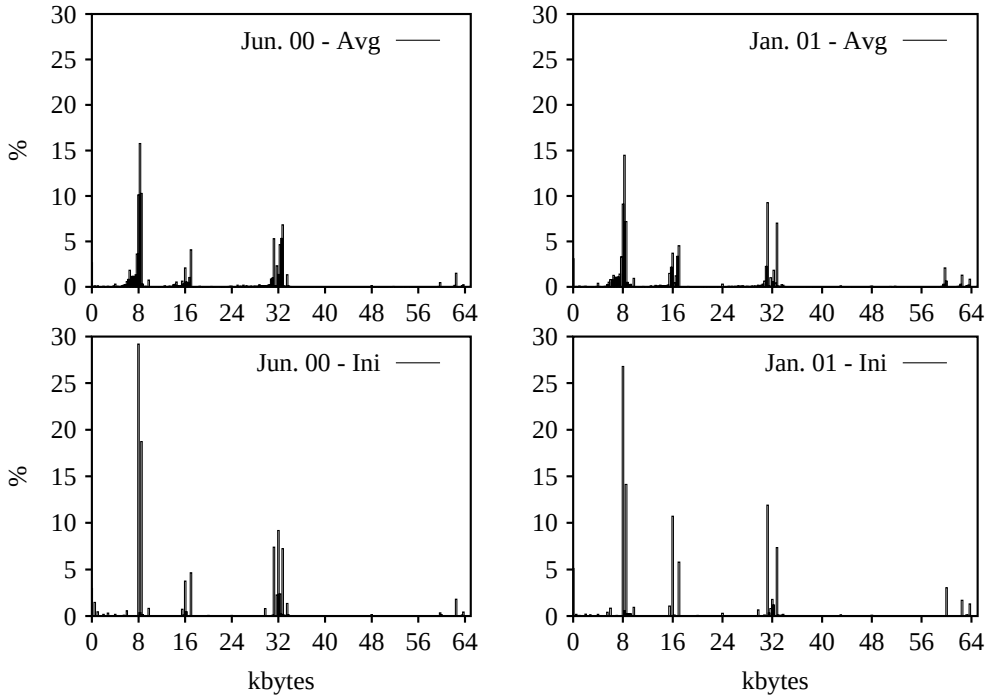

Fig. 9. Distribution of "*rwnd*:" as advertised during handshake (ini – lower plot) and averaged during the whole connection (ave – upper plots)

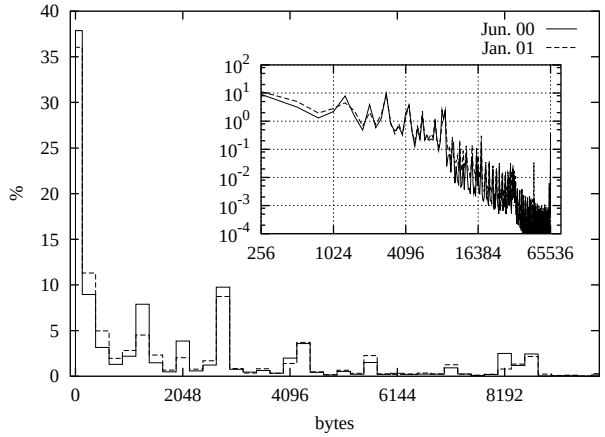

Fig. 10. TCP in-flight size distribution as seen from the measurement point for outgoing connections

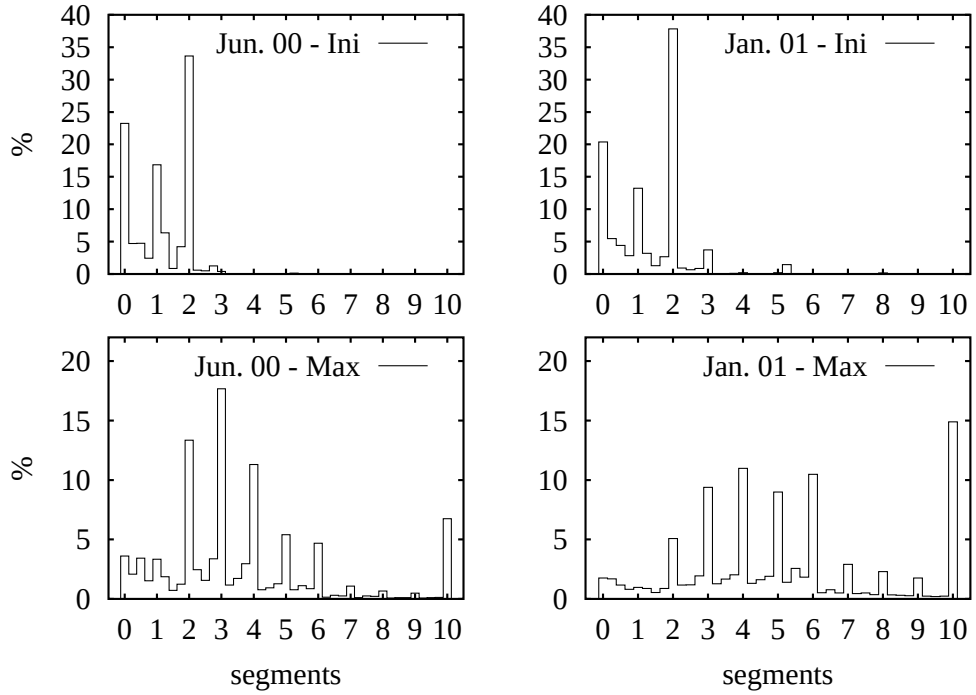

Fig. 11. Estimated initial and maximum TCP in-flight size

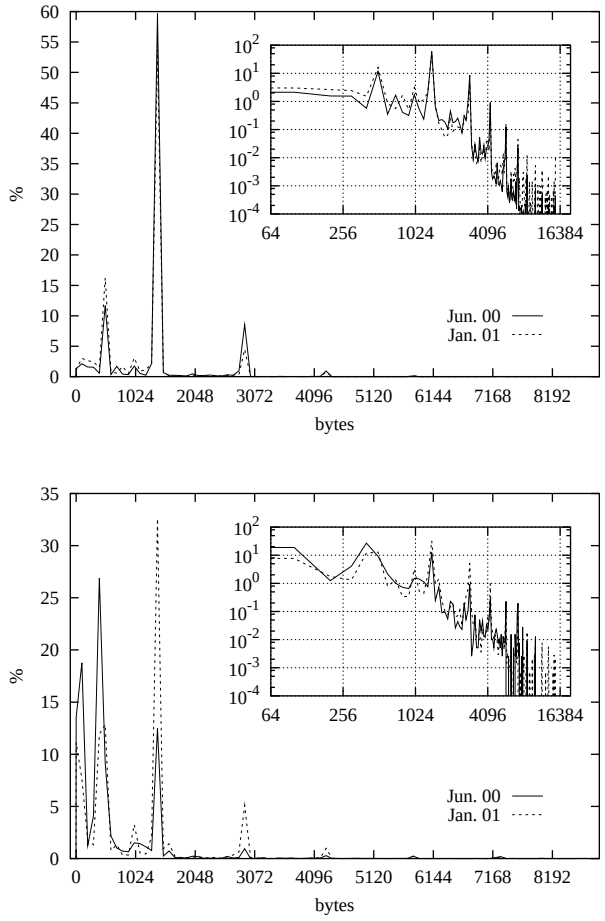

Fig. 12. Bursts of out-of-sequence data observed in incoming (top) and outgoing (bottom) connections

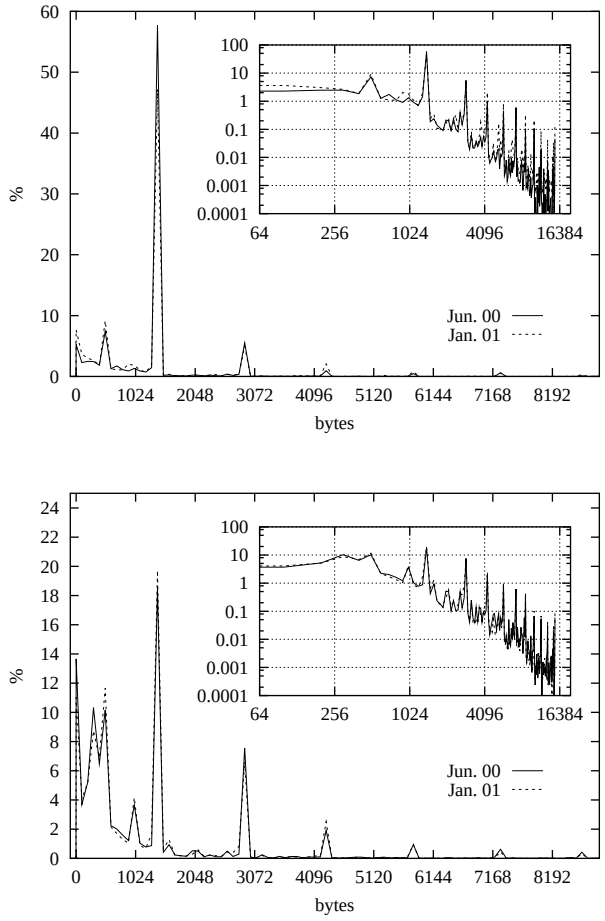

Fig. 13. Bursts of duplicated data observed in incoming (top) and outgoing (bottom) connections

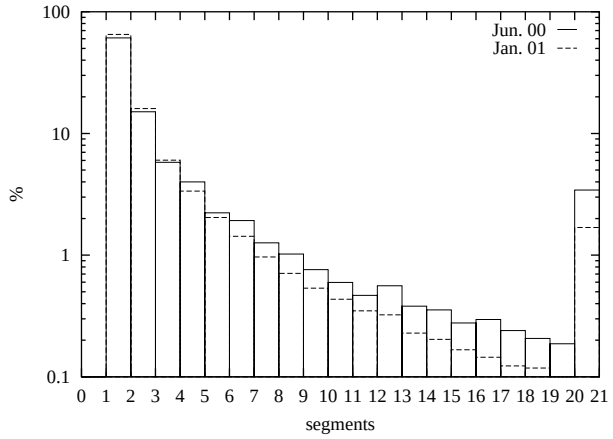

Fig. 14. Density function of the probability of suffering multiple packet losses for flows larger than 10 MSS, conditioned on the probability of losing at least one packet

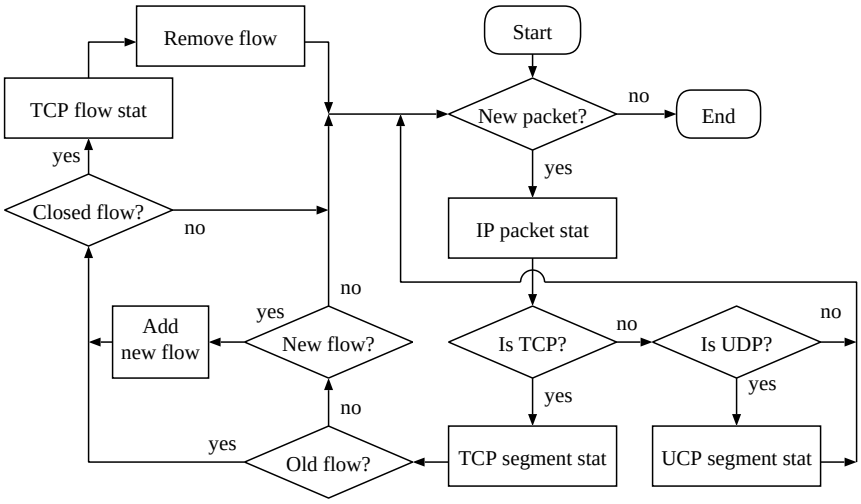

Fig. 15. Block diagram of the trace analyzer