Computer Networks 52 (2008) 1846–1858

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

A root cause analysis toolkit for TCP

Matti Siekkinen [a], Guillaume Urvoy-Keller [b], Ernst W. Biersack [b,*], Denis Collange [c]

[a] University of Oslo, Department of Informatics, P.O. Box 1080 Blindern, NO-0316 Oslo, Norway
[b] Institut Eurecom, 2229 Route des Crêtes, Sophia Antipolis, France
[c] Orange Labs, 905, rue A. Einstein - 06921 Sophia Antipolis, France

ARTICLE INFO

Article history:
Received 23 July 2007
Received in revised form 29 January 2008
Accepted 13 March 2008
Available online 21 March 2008

Responsible Editor: R. LoCigno

Keywords:
TCP
Performance
Throughput limitation
Trouble shooting

ABSTRACT

TCP is the most widely used protocol for data transfer over the Internet and for most applications the performance metric of interest is throughput. Identifying the reasons for the throughput limitation of an observed connection is a complex task. We present a set of techniques, called the TCP root cause analysis toolkit, that allow users to determine from a passively captured packet trace the primary cause for the throughput limitation. We present the details of the toolkit and its validation and apply the toolkit to carry out a case study of ADSL traffic.

1. Introduction

Today, TCP carries the vast majority of Internet traffic. Throughput is the most important performance metric for many network applications that utilize TCP to transmit data. Therefore, knowledge about which factors determine the TCP throughput observed by an application is very useful as it can help improve the performance of the applications and of TCP itself. It can also help network operators to identify performance problems and bottlenecks in their network.

The throughput achieved by a given TCP connection is determined by many factors, such as the available bandwidth on the individual links of the end-to-end path and the TCP protocol mechanisms. In many cases it can also be the application itself that determines the throughput, as was first pointed out in [20]. Finding out which of the possible reasons is the cause for the observed throughput at a given time instant is not trivial. We call this kind of analysis *root cause analysis of TCP throughput*. It may be straightforward to do root cause analysis if one has such access to the transmitter host that allows inspecting the state of the TCP state machine and application at any given time (using for instance the Web100 utility [12]). However, it becomes much more difficult when one is only able to observe traffic at a given measurement point inside the network, which is a very common case for large scale measurement studies. For example, an ISP that would like to analyze the behavior of its thousands of clients would often do so by studying the aggregated traffic flow at the edge of the access network, since in many cases it is impossible to directly access the client hosts.

We have designed and implemented a set of algorithms, the *root cause analysis toolkit*, for doing root cause analysis of TCP throughput. This toolkit analyzes TCP/IP packet headers that are passively collected at a single measurement point. The algorithms are designed in such a way that they impose no restrictions on the location of the measurement point, similarly to the work presented in [9], for instance. This design choice clearly complicates the design

* Corresponding author. Tel.: +33 4 9300 8111; fax: +33 4 9300 8200.
E-mail addresses: siekkine@ifi.uio.no (M. Siekkinen), urvoy@eurecom.fr (G. Urvoy-Keller), erbi@eurecom.fr, ernst.biersack@eurecom.fr (E.W. Biersack).

1389-1286/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.comnet.2008.03.005

of the algorithms but is worth the effort, as it greatly increases the usability of the toolkit.

Our approach firsts isolates the bulk data transfer periods (BTP) and the application limited periods (ALP) within a TCP connection. A BTP is a period where the TCP sender never needs to wait for the application on top to provide data to transfer. On the other hand, when the TCP sender needs to wait for the application on top, we call that period an ALP. We presented in [17] an algorithm that identifies the BTPs and ALPs. Once BTPs have been identified, the *root cause analysis toolkit* analyzes them for TCP and IP layer throughput limitations, i.e. inferring the root causes for the BTPs, which will be the focus of this paper.

The contributions of this paper are the following: We describe a methodology to quantify TCP and IP level throughput limitations that is referred to as the root cause analysis toolkit. More specifically, we define a set of quantitative metrics, which we call *limitation scores*, that can be computed from the information contained in the packet headers collected at a single measurement point, and show how these scores can be used in a threshold-based classification scheme to derive a root cause for a given BTP. We also show how to infer suitable threshold values for the scheme. Finally, we apply the root cause analysis kit to study the performance of ADSL clients, part of which has already been presented in [15].

2. What limits the throughput of TCP?

The common view of a TCP transfer is that its transmission rate is limited by the network, i.e. by a link with a small capacity or a congested bottleneck link. We demonstrate in this section through examples that this view is too restrictive. Instead, the limitation causes may lie in different layers of the network stack, either in the end-points or in the middle of the TCP/IP data path. Zhang et al. [20] pioneered research into the origins of TCP throughput limitation causes. They defined a taxonomy of rate limitations (application, congestion, bandwidth, sender/receiver window, opportunity and transport limitations) and applied it to various packet traces. While our classification is greatly inspired by their work, we extend the scope of their work and discuss the difficulties of identifying certain causes through examples. We present in the next section the causes in a top-down manner, starting from the application level down to the network level.

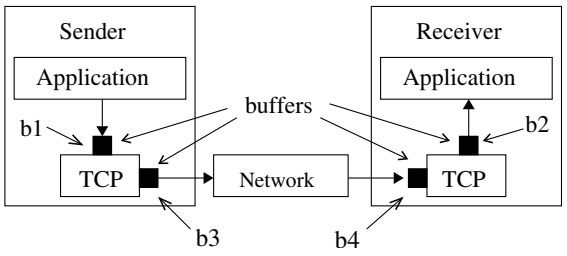

Fig. 1. Data flow from the sender to the receiver application through a single TCP connection.

2.1. Application

Fig. 1 describes the way data flows from sender to receiver application through a single TCP connection. The interaction happens through buffers: at the sender side the application stores data to be transmitted by TCP in buffer `b1`, while at the receiver side TCP stores correctly received and ordered data in buffer `b2`. This buffer is consequently read by the receiving application. Data that is received out of order is stored in buffer `b4` until it can be delivered in order and stored into buffer `b2`. In case the receiver application is unable to read the buffer `b2` as fast as TCP delivers data into it, the receiving TCP will notify the sending TCP by lowering the receiver advertised window. We discuss this case as a TCP layer phenomenon in Section 2.2.

We define two types of periods within a given TCP connection. When the application sends data constantly, buffer `b1` in Fig. 1 always contains data waiting to be transferred. We refer to such a period as a bulk transfer period (BTP). In other cases, the application limits the throughput achieved, i.e. TCP is unable to fully utilize the network resources due to lack of data to send. We call such a period an application limited period (ALP). Note that a TCP connection consists entirely of these two types of periods.

We depict the diverse classes of applications operating on top of TCP in Table 1. This classification illustrates the multiple ways in which the application can influence the TCP traffic. In this paper, we focus on the root cause analysis of BTPs while the problem of identifying BTPs and ALPs has been tackled in [17].

2.2. TCP layer

2.2.1. TCP end-point buffers

The achieved throughput of TCP can be limited by the size of the buffers allocated at the two end-points of a connection, i.e. buffers `b3` and `b4`, but also `b2` as discussed in the previous section. In this case, the receiver side buffer `b2` (between the TCP layer and the application layer) constrains the maximum number of outstanding bytes the other end is allowed to send at any given time instant. We call this limitation *receiver limitation*. Note that in theory, buffer `b4` can also limit the maximum outstanding bytes if a lot of bytes are received out of order and the buffer is small. However, we observed that this behavior is rare.

More precisely, we observe that receiver limitations can occur in two flavors: as an intentional or unintentional cause. Intentional limitation is imposed by the receiver application because it is unable to process data as fast as TCP delivers it (recall Section 2.1). Unintentional limitation can happen if the receiving TCP advertises an unnecessarily small window by default. For example, consider a situation where two applications communicate with each other through an end-to-end path with high bandwidth and relatively long delay. If the TCPs do not use receiver window scaling[1], the maximum advertised window size will be

[1] Some TCP implementations do not use window scaling by default. In addition, certain other implementations can fail to negotiate the scaling factors properly preventing the use of window scaling.

Table 1
Summary of different application types

Type	Main characteristics	Example applications
1	Constant application limited transmission rate, consists of a single ALP	Skype and other live streaming, and client rate limited eDonkey
2	User dependent transmission rate, typically a single ALP	Telnet and instant messaging applications
3	Transmission bursts separated by idle periods, applications using persistent connections, BTPs interspersed with ALPs	Web w/persistent HTTP connections, BitTorrent (choked and unchoked periods)
4	Transmit all data at once, single BTP	FTP
5	Mixture of 1 and 3	BitTorrent with rate limit imposed by client application

64 Kbytes. In this case it suffices to have more than 3.5 Mbit/s of available bandwidth with a RTT of 150 ms (typical for a transatlantic path) to be receiver limited with the maximum possible receiver advertised window.

At the sender side, the buffer `b3`, between the TCP layer and the MAC layer, constrains the maximum number of bytes in the retransmit queue. Consequently, the size of the sender buffer also constrains the amount of unacknowledged data that can be outstanding at any time. We call this limitation *sender limitation*.

2.2.2. Congestion avoidance mechanism: transport limitation

We declare a connection to be *transport limited* if the sending TCP is in congestion avoidance and experiences no losses, thus no limitation by the network (see Section 2.3). In addition, the sending TCP does not reach the limit set by the receiver advertised window before the end of the transfer. Hence, the remaining limitation cause is the congestion avoidance mechanism of the TCP protocol that slowly increases the size of the congestion window. This phenomenon may occur when the initial slow start threshold is set unnecessarily low. In this case the sending TCP enters congestion avoidance before experiencing any losses. Another example is a relatively short transfer through a path with a lot of available bandwidth and a large scaled receiver advertised window.

2.2.3. Short transfers: slow start mechanism

There is an additional type of limitation that can be considered as a limitation at the transport layer. This limitation occurs for short connections carrying so few bytes that the connection never leaves the slow start phase. Since it is the slow start behavior of TCP that limits the rate of the TCP transfer we do not classify these connections as application limited.

2.3. Network

A third category of limitation causes is due to the network. We focus on the case where one or more bottlenecks on the path limit the throughput of the connection (see [8] for a study on the location and lifetime of bottlenecks in the Internet). While other network factors, such as link failures or routing loops [18], might impact a TCP connection, we do not consider them in this paper as we can reasonably expect their frequency to be negligible.

Let us first define general metrics independent of the transport protocol [13]:

- Capacity C_i of link i: the maximum possible IP layer transfer rate at that link.
- End-to-end capacity C of a path: $C = \min_{1,\ldots,H} C_i$, where H is the number of hops in the path.
- Average available bandwidth A_i of a link i: $A_i = (1 - u_i)C_i$, where u_i is the average utilization of that link in a given time interval.
- Average available end-to-end bandwidth A of a path: $A = \min_{1,\ldots,H} A_i$.

We also define two TCP specific metrics:

- Bulk transfer capacity (BTC_i) of link i: the maximum capacity obtainable by a TCP connection at that link.
- Bulk transfer capacity (BTC) of a path: $\mathrm{BTC} = \min_{1,\ldots,H} \mathrm{BTC}_i$, where H is the number of hops in the path. The BTC depends on how the TCP throughput is affected by other flows.

As in [13], we call the link i with capacity $C_i = C$ the *narrow link* of the path and the link j with an average available bandwidth $A_j = A$ the *tight link* of the path. Furthermore, we define link k as the *bottleneck link* if it has a bulk transfer capacity $\mathrm{BTC}_k = \mathrm{BTC}$. Note that while, at a given time instant, there is a single bottleneck for a given connection, the location of the bottleneck as well as the bulk transfer capacity at the bottleneck can change over time.

If the bottleneck link explains the throughput limitation observed for a given TCP bulk transfer, it can be considered as *network limited*. Please note that the bottleneck link is not necessarily the same as the tight link.

We distinguish two different cases of network limitation depending on the type of bottleneck link on the path: *unshared and shared bottleneck limitations*. Intuitively, an unshared bottleneck limitation means that the considered TCP transfer utilizes alone all the capacity of the bottleneck link that is responsible for the throughput achieved. In a shared bottleneck case, there is cross traffic competing for the bandwidth of the bottleneck link.

A bottleneck link that limits the throughput of a given TCP connection typically generates packet losses when the buffer at the bottleneck link is overrun. However, it is possible that a TCP transfer whose rate is limited by a bottleneck link does not experience any losses. Whether losses occur depends primarily on the size of the buffer in the bottleneck link and the size of the receiver advertised window. To understand why, let us consider a scenario where a TCP transfer is set up on a given path.

Let MSS be the packet size in bytes that the TCP sender uses on the path, RTT the round-trip time when queues on the path are empty and W_r the receiver advertised window in (MSS size) packets. We further assume that the narrow link i on the path has capacity C_i and that available bandwidth also equals C_i (i.e. no cross traffic).

Now, suppose that $W_r > \frac{RTT \cdot C_i}{MSS}$, i.e. the receiver advertised window is larger than the maximum number of packets that can be transmitted through the narrow link per RTT. If $B > W_r - \frac{RTT \cdot C_i}{MSS}$, where B is the buffer size at the bottleneck link, then the congestion window of the TCP sender will reach the size W_r and no losses occur at the bottleneck link. Otherwise, losses should occur and TCP should back off.

The above described scenario is the simplest possible one, but the same reasoning applies also to the case of a shared bottleneck limitation. If the buffer of the bottleneck link is not overrun before the TCP sender exhausts the receiver advertised window, there will be no losses.

3. Related work

In [20], the authors introduced the TCP rate analysis tool (T-RAT). We implemented our own version of this tool, as it was not publicly available, and experimented with it. Unfortunately, T-RAT turned out to suffer from a number of limitations. The limitations of T-RAT originate from the fact that it was designed to work only on unidirectional traffic traces. While this choice may clearly increase the usability of the tool, it also severely decreases its accuracy in certain cases. We discuss in detail the limitations of T-RAT and perform a comparison against our methods in [14].

Our early work presented in [16] described initial methods to infer root causes of the throughput for TCP connections. While the underlying "divide-and-conquer" approach is similar to our current strategy, the algorithms have been refined substantially. Furthermore, in [16] we described the computation of quantitative metrics, but we did not address the issue of mapping these metrics to a single dominant root cause.

4. Approach

In [17], we make a strong case for filtering out application effect prior to analyzing transport and network layer aspects. In that paper, we describe the IM algorithm that partitions the packets of a given TCP connection into ALPs and BTPs. The algorithm relies only on the observation that (1) idle periods longer than the RTT of the connection preceded by a packet smaller than the MSS and (2) many consecutive packets smaller than the MSS are both indicators of application limitation.

In this paper, we focus on the root cause analysis of non-application limited traffic (i.e. the BTPs of a connection). Thus, we focus on the limitation cause at the transport and network layers.

Our approach consists of two phases. In the first phase, we compute for each BTP several *limitation scores* (Section 5). These scores are quantitative metrics that assess the level of a given limitation cause experienced by a BTP. We compute all these scores from information found in the packet trace.

In the second phase (Section 6), we use a classification algorithm that intends to associate a single limitation cause to each BTP whenever it is possible. In practice, we observed that, most of the time, a single cause is enough to explain a certain behavior. Please refer to [16] for some anecdotal counter-examples.

4.1. Scope of our work

We do not treat all possible scenarios in our root cause analysis of TCP traffic. We focus only on the analysis of long-lived connections and neglect the short ones. Previous work (e.g. [21,4]) has studied the performance of short TCP transfers from a measurement and analytical point of view. The slow start mechanism of the TCP protocol is typically considered as the main limitation cause for the short transfers. Our definition of a long-lived connection is such that the connection is not likely to be limited by the slow start mechanism of the TCP protocol. In practice, we set a threshold around 100–150 Kbytes. See [17] for details.

We also assume that the traffic analyzed passes through routers that all use FIFO scheduling. One crucial components of our root cause analysis technique, which imposes this limitation, is the capacity estimation tool PPrate [7]. PPrate uses packet dispersion techniques, which only work if packet scheduling is FIFO. While most of todays Internet routers use FIFO scheduling, there are cases, such as transmission over cable modems and wireless 802.11 access networks [11] where scheduling can be non-FIFO.

5. Quantitative analysis: the limitation scores

We compute the limitation scores by inspecting the packet headers. Packet header traces are captured and timestamped at a single measurement point along the path from a source to a destination. In order to be as generic as possible, we do not impose restrictions on the location of the measurement point relative to the TCP end-points. Tnis is important since we may use publicly available traces collected by third parties for which we do not have exact information about the measurement configuration. Instead, we infer the location of the measurement point from the traces (Section 5.1).

5.1. Determining position of measurement point

Most of our metrics are straightforward to compute if the packet trace is captured at the sender side.

Our first problem is thus to infer the location of the measurement point. Let us consider the case depicted in Fig. 2, where the measurement point A is close to the sender while points B and C are not. We can determine the measurement point with respect to the connection initiator (also the sender in Fig. 2) by measuring and comparing the delay between the SYN and SYN + ACK packets, and the delay between SYN + ACK and ACK packets, referred to as $d1$ and $d2$, respectively, for the measurement point A. We

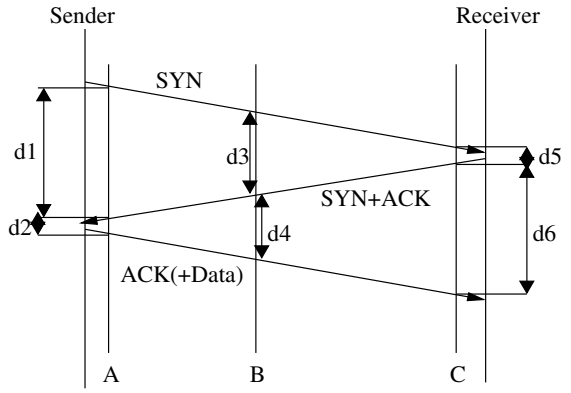

Fig. 2. Determining the measurement position from the three-way handshake of TCP.

conclude that A is close to the connection initiator if $\frac{d2}{d1} < 0.01$. Note that the connection initiator is not always the sender.

5.2. Metrics inferred from packet header information

We use the following five different time series and the path capacity estimate as bases for generating the limitation scores for the throughput of BTPs.

5.2.1. Time series of round-trip times

We need to compute running RTT estimates for each BTP throughout its lifetime. In the case where the measurement point is close to the sender, we compute the RTT for each acknowledged data packet as the time interval between the timestamps of the last transmission of a data packet and the first acknowledgment for this data packet.[2] If the measurement point is away from the sender and TCP timestamps are available, we use either the TCP timestamp-based method introduced in [19], otherwise we use the method described in [10].

5.2.2. Time series of inter-arrival times of acknowledgments

We compute the inter-arrival times of acknowledgments separately for each direction of a connection. The ACKs included in the computation are either acknowledging one or two data packets of size MSS or duplicate acknowledgments. Furthermore, to cancel the effect of delayed ACKs we divide by 2 the inter-arrival time of ACKs that acknowledge two data packets, which can be detected by comparing the difference of the current and the previously seen highest ACK number to the MSS.

5.2.3. Time series of retransmissions

We cannot assume to observe all retransmitted packets twice since the packets may be lost before the measurement point, especially if the measurement point is far from the sender. Therefore, simply counting bytes carried by packets seen more than once is not sufficient. A packet is considered to be a retransmission if (i) the packet carries an end-sequence number lower than or equal to any previously observed one; and (ii) the packet has an IPID[3] value higher than any previously observed values. With the help of the IPID we remove false positive retransmissions caused by reordering of packets by the network that can occur if the measurement point is far from the sender. Similar analysis of out of order packets was also done in [5].

5.2.4. Time series of receiver advertised window

We compute a time series for the receiver advertised window, which consists of time-weighted average values over a given time interval: Each time a packet is received from the other end, the receiver window indication in the packet will be considered as the current receiver window value until either the end of the time window is reached or a new packet is received. This technique is valid if the measurement point is located at the sender side. However, if the measurement point is away from the sender, we virtually shift in time the observed timestamp values by the time delay between the sender and the observation point. For example, in Fig. 2, when the measurement point is at C, we would shift in time the timestamp values of packets sent by the receiver by $+\frac{d6}{2}$, which is the estimated time at which this packet should arrive at the sender. $d6$ is obtained from the running RTT estimates and is, thus, continuously updated.

5.2.5. Time series of outstanding bytes

Another metric of interest is the amount of data bytes sent and not yet acknowledged at a given time instant. Since the computation is done by inspecting both directions of the traffic, we again need to take into account the location of the measurement point.

If the measurement point is close to the sender, we produce the time series by calculating the difference between the highest data packet sequence number and the highest acknowledgment sequence number seen for each packet and then averaging these values over a time window, in the same way that we do for the receiver advertised window values.

If the measurement point is away from the sender, we do the computation by shifting in time the timestamp values of arriving packets. For example, in Fig. 2, we would shift the timestamp values of data packets arriving from the sender at C by $-\frac{d6}{2}$ and of acknowledgments arriving from the receiver at C by $+\frac{d6}{2}$.

5.2.6. Path capacity

As mentioned in Section 4.1, we use a tool called PPrate to estimate the capacity of the IP path of a given connection. This tool applies statistical analysis on the inter-arrival times of packets in order to infer an estimate for the capacity of the path. In our case, we use the inter-arrival

[2] We must take into account that packets may be sent multiple times, in the case of losses, and similarly acknowledged multiple times, in the case of lost or piggy-backed acknowledgments.

[3] IPID = IP identification number is assumed to be unique for each IP packet originating from a given sender. However, this number is only 16 bits long and therefore wrap around of this number needs to be taken into account. Note that actual implementation of the IPID counter differs from one OS to the other [6].

times of acknowledgments that we compute as explained above.

5.3. Limitation scores

We compute four different limitation scores for each connection: receiver window limitation score, burstiness score, retransmission score, and dispersion score. The first two ones are used to identify receiver limitation through the advertised window or the transport layer. The two other scores are used to identify different cases of network limitations, i.e. limitations by unshared and shared bottleneck links.

5.3.1. Receiver window limitation score

We use two time series to compute the receiver window limitation score: the outstanding bytes time series and the receiver advertised window time series. Both of these time series are computed using a time window equal to the minimum RTT of the connection observed. In this way, we ensure that we capture rapid dynamics in the receiver advertised window. The difference of the values of these two time series indicates how close the TCP sender's congestion window is to the limit set by the receiver window.

Specifically, for each pair of values in the two time series, we compute their difference and generate a binary variable with value 1 if this difference is less than $lb * \text{MSS}$ and 0 otherwise, where lb is a small value (typically $lb \in \{1, 2, 3\}$). The receiver window limitation score is the average value of the resulting binary time series for the analyzed bulk transfer period. We experimented with different values for the lb threshold and observed that the choice among the values $lb \in \{1, 2, 3\}$ is clearly not critical (see [14] for details).

5.3.2. Burstiness score (b-score)

The outstanding bytes of the TCP sender can reach the limit of the receiver advertised window in two different cases:

- When the buffer of the TCP receiver is too small and the transfer is, thus, receiver limited.
- When the transfer is network limited and the link buffers on the path, and specifically on the bottleneck link, are large enough to prevent losses.

We introduce the burstiness score, a.k.a. b-score, to distinguish these two cases. If the receiver window limits the throughput, the sending TCP needs to wait for acknowledgments for a burst of packets during a significant part of the RTT before it can send new packets. In contrast, if there is a shared bottleneck link that limits the throughput, the inter-spacing pattern of packets is smoothed out by the cross traffic at the bottleneck link and a higher throughput would not be achieved even for larger receiver window values. Instead, growing the receiver window would at some point lead to buffer overflow at the bottleneck link, causing retransmissions of packets and reduction of the congestion window value, which would lower the overall throughput.

We next present two definitions of the b-score value. The first one is in line with the scenario depicted above while the second one is easier to compute as it does not rely on the somewhat complex estimation of the capacity of the path.

Consider the receiver limited scenario depicted in Fig. 3 where y is the time the TCP sender waits because it has exhausted its receiver window, and x is the time it takes to send a receiver advertised window full of packets. Hence, the throughput is less than the available bandwidth on the path because of the waiting time y, i.e. the sender does not "fill the pipe" because of y. Thus, our first definition of the b-score is:

$$\mathscr{B} = 1 - \frac{x}{x+y} = 1 - \frac{(W_r - 1)\frac{\text{MSS}}{C}}{\text{RTT}}, \tag{1}$$

where W_r is the average receiver advertised window size divided by MSS and C is the path capacity (Section 2.3).

Based on the above definition, if the limitation cause experienced by a connection is receiver window limitation, $\mathscr{B}$ should be close to 1. On the other hand, when the limitation cause is a bottleneck link with large enough buffer, inter-arrivals of packets should be controlled by the bottleneck link and hence $\mathscr{B}$ should be close to 0.

The above definition requires the computation of both, RTT and C. It is not always easy to estimate those two values. That is why we compute an approximation for $\mathscr{B}$, based on the inter-arrival time (IAT) of packets at the measurement point. By definition of the average inter-arrival time value $\overline{\text{IAT}}$, one obtains that:

$$\overline{\text{IAT}} = \frac{\text{RTT}}{W_r}.$$

In addition, we observe that the highest quantiles of the IAT random variable should correspond to y, see Fig. 3. The exact quantile to be chosen is to be related to the value of the advertised window. Intuitively, if the advertised window corresponds to $n+1$ MSS packets, one should pick each nth IAT value. Thus, the quantile p that we picked in the IAT distribution for a receiver window of W_r packets is the $100 \cdot \left(1 - \frac{1}{W_r}\right)$-th one. The quantile that we obtain should then equal y. Combining the above definitions with Eq. (1), we obtain an approximation for $\mathscr{B}$, that we term b-score:

$$\text{b-score} = \frac{\text{IAT}_p}{\overline{\text{IAT}} \cdot W_r} \approx \mathscr{B}. \tag{2}$$

We use the b-score instead of $\mathscr{B}$ in our root cause analysis toolkit to quantify the burstiness of a connection.

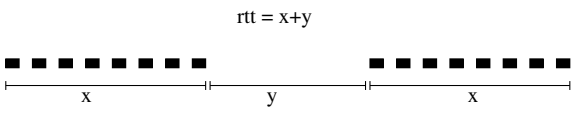

Fig. 3. Inter-arrival times of receiver window limited transfer. Black rectangles are packets sent and time runs from right to left.

5.3.3. Retransmission score

The retransmission score for a BTP is computed as the ratio of the amount of data retransmitted divided by the total amount of data transmitted during this period. Note that since TCP may perform unnecessary retransmissions, this score does not exactly correspond to the loss rate. However, we can expect these quantities to be strongly correlated most of the time.

5.3.4. Dispersion score

The objective of the dispersion score is to assess the impact of the bottleneck link on the throughput of a connection. We define the *dispersion score* as follows (C is the capacity of the path, and *tput* is the average throughput of the BTP):

$$\text{dispersion score} = 1 - \frac{tput}{C}. \quad (3)$$

A correct estimation of C, the capacity of the path, is very important for the computation of this score.

Let us first consider the case where the network limitation cause is a non shared bottleneck link on the path. The bottleneck is evidently the narrow link of the path (refer to Section 2.3 for the definition). Since the BTP is network limited and the narrow link is not shared, we have $tput \approx C$ and the dispersion score should be close to zero. In all other cases, including the shared bottleneck limitation, the dispersion score is greater than zero.

5.3.5. Validation

We validated the computation of the receiver window limitation score in [14]. We concluded that, in 90% of the cases the maximum discrepancy between the actual and estimated values of the receiver window limitation score remains below 20%. The accuracy of the dispersion score is evaluated in [7]. As for the b-score, we perform a detailed evaluation under different conditions in [14], some of which we discuss in Section 6.

6. Interpreting the limitation scores

Quantitative scores aim at uncovering the rate limitation causes of each BTP. In theory, a given BTP can experience several limitation causes simultaneously. Note, however, that all combinations of causes are not possible. Being simultaneously limited by congestion (in a shared bottleneck) and a too low advertised window is possible. On the other hand, transport layer and unshared bottleneck limitations are exclusive, since with the former no losses are experienced. In practice however, it turns out that most of the time, a single dominant limitation cause is found for a BTP (and also for a connection). Our classification scheme is based on this empirical observation, though it leaves the door open to the case of mixed limitation causes.

6.1. Classification scheme

We present here a threshold-based classification scheme of BTPs where each score has a threshold attached to it. We consider the causes one after another by elimination. At each step, one score is evaluated against a threshold and, as a result, one or more causes are eliminated. The cause that remains in the end is estimated to be the root cause we are looking for.

Our scheme corresponds to the flow chart depicted in Fig. 4. We introduce a set of thresholds which we calibrate later in Section 7. We first describe the different steps in the flow chart.

The first step is to determine whether the root cause is a bottleneck link that is unshared. For that, we compare the dispersion score against the corresponding threshold *th1*. If the dispersion score is low enough, it means that the BTP achieves a throughput close to the capacity of the path. Note that even if the throughput of a BTP is limited by an unshared bottleneck, the average throughput computed for the entire BTP (total bytes divided by total duration) can be smaller than the capacity of the path for two reasons. The first reason is that TCP needs to tune to reach the final transmission rate by growing the congestion window gradually, which lowers the average throughput of the BTP, supposing that the BTP starts from the beginning of the connection. The second reason is that TCP may periodically overrun the buffer at the bottleneck link unless the number of outstanding bytes of the TCP sender is limited by the receiver advertised window. In this situation, TCP may experience momentarily lower throughput due to loss recovery, depending on the loss recovery strategy. Some TCP versions recover fast (e.g. SACK enabled Newreno) with no significant reduction in the throughput, while other versions may suffer much more. Naturally, the degree of reduction in the throughput depends also on the number of packets lost when the buffer is overrun.

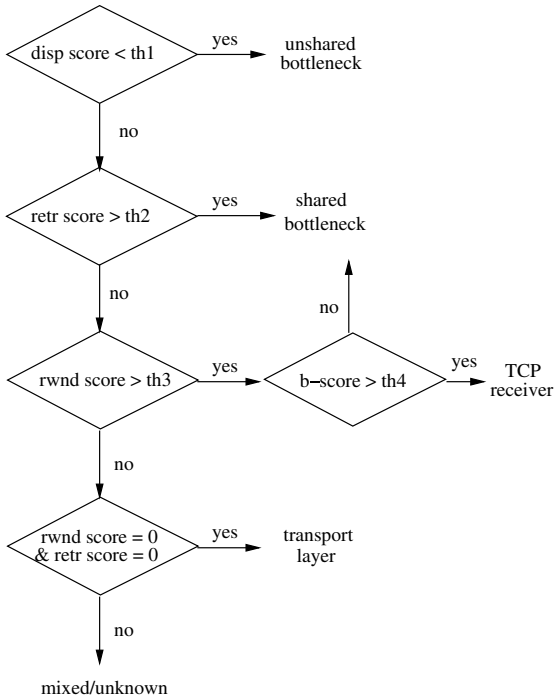

Fig. 4. Root cause classification scheme.

The second step, comparing the retransmission score to the corresponding threshold *th2*, identifies BTPs limited by a shared bottleneck link. Remember that retransmissions are clear indications of the presence of a bottleneck link limitation. Since the first step eliminates those BTPs whose throughput is limited by an unshared bottleneck link, a shared bottleneck link is the obvious cause for the BTPs with high retransmission score.

If no retransmissions are observed and there is no unshared bottleneck link limiting the throughput, our scheme inspects the receiver window limitation score. In the third step, this score is compared to the threshold *th3*. If the score is above the threshold, the transmission rate of the BTP is determined either as receiver limited or limited by a shared bottleneck link. The final separation between the receiver limited BTPs and shared bottleneck link limited BTPs is done by comparing the b-score to the threshold *th4*.

The final step distinguishes the BTPs whose throughput is transport limited, i.e. limited by the TCP protocol (either slow start or congestion avoidance mechanism), from those that cannot be classified. We consider as transport limited those BTPs that do not experience any retransmissions and no limitation by the receiver advertised window. Otherwise the BTP is classified as limited by a mixture of causes or an unknown cause.

6.2. Discussion

In the classification scheme that we presented above, we did not discuss the specific order in which we apply the tests. We justify our choices in this section. Inspection of the dispersion score needs to be done before we determine anything about shared bottleneck limited transfers. Otherwise a large part of the transfers limited by an unshared bottleneck link would be classified as shared bottleneck limited because of a high enough retransmission score for instance. The ordering between steps 2 and 3 (shared bottleneck and receiver limitations) does not matter due to the b-score test. Eventually, the step to identify transport limited BTPs could be performed at any point.

7. Inferring the threshold values

The flow chart of our classification scheme presented in Fig. 4 contains four thresholds. We present in this section a method to infer these threshold values from measurements. We try to be as general as possible, but any such method is always unable to capture all the dynamics of the Internet. Simulations are not an option because they fail to capture the diversity of Internet traffic. We rather use a controlled approach where we download traffic from the Internet that we constrain in different ways when it crosses the edge, so as to obtain enough samples for each limitation cause.

7.1. Experimental setup

We proceeded as follows to obtain traffic corresponding to each of the three different rate limitation causes: unshared bottleneck, shared bottleneck, and receiver limitations.

We performed FTP downloads initiated from a machine at Institut Eurecom, which also recorded the traffic traces. We selected all the FTP mirror sites for the Fedora Core Linux distribution [2] that are located in 42 different countries distributed over all five continents. We selected randomly a new server from the list whenever the previous download was finished and downloaded each time the same set of files of different sizes with a total amount between 40 and 60 Mbytes. The number of simultaneous downloads was controlled in order to produce unshared and shared bottleneck limited traffic. We used rshaper [3] to create an artificial bottleneck link at a machine (at the boundary of the Eurecom network) when needed and similarly NISTNet [1] to delay packets in order to increase RTTs.

7.1.1. Unshared bottleneck limited transfers

In order to generate traffic that should be limited, with high probability, by an unshared bottleneck link, we created an artificial bottleneck and downloaded from one single server at a time. We performed the experiments with three different bottleneck link capacities: 0.5, 1, 2 Mbit/s.

7.1.2. Shared bottleneck limited transfers

To generate transfers likely to be limited by a shared bottleneck, we downloaded from several servers simultaneously. Our download script ensured that downloads from 10 servers were continuously ongoing. We used the following bottleneck link capacities during these experiments: 1, 3, 5, 10 Mbit/s. We checked that the bottleneck link was fully utilized during all the experiments.

7.1.3. Receiver limited transfers

Transfers are typically receiver limited in the case of a high bandwidth delay product where the sender fully exhausts the receiver advertised window before the ACKs arrive. That is why we used NISTNet to delay packets in order to increase the RTT. We experimented with different amounts of delay added to the RTT: 100, 200, 400, 500 ms. We checked that during the experiments the aggregate throughput never exceeded the link capacity at the edge of Eurecom in order to be sure that there was no shared bottleneck on our side.

7.2. Setting the thresholds

We set the threshold for the retransmission score (*th2* in Fig. 4) to 1%. It is an empirically justified choice: this value seemed to work the best for the data sets from these controlled experiments. More details are in [14].

As for the receiver window limitation score, we chose a threshold of 50% (*th3* in Fig. 4). We chose such a value in order to capture those BTPs that experience a throughput limitation by the receiver or by a shared bottleneck link a majority of the time.

The threshold for the dispersion score (*th1* in Fig. 4) distinguishes the BTPs whose transmission rate is limited by an unshared bottleneck link from the rest. We inspected the CDF plots of the dispersion score for BTPs generated

while downloading from a single server at a time. We noticed that a threshold of 0.2 for the unshared bottleneck limitation seems to be a good choice.

The b-score threshold (*th4* in Fig. 4) separates the receiver limited BTPs from those that are limited by a shared bottleneck link. Therefore, to determine a suitable value for the threshold, we analyzed the traffic from the experiments where we generated traffic corresponding to these two rate limitations and analyze the b-score values for the resulting two types of traffic in order to find a suitable threshold value.

Fig. 5 shows CDF plots of the b-score for BTPs from two different types of experiments with 10 simultaneous downloads: experiments where an artificial bottleneck was set up with different capacities, resulting in shared bottleneck limited traffic, and experiments where different amounts of delay were added, which in turn generated receiver limited traffic with high probability due to the increased bandwidth delay product of the path. We selected for this analysis only those BTPs that had a receiver window limitation score above 0.5, retransmission score below 0.01 and dispersion score above 0.2, that is, the ones that would require the use of the b-score in the classification process (Fig. 4).

In order to find the *th2* threshold value that gives the best separation between these two types of traffic, we compare the CDF plots of the two types of traffic against each other. We try to select a single b-score value that is at the same time larger than most b-score values for the shared bottleneck experiments and smaller than most b-score values for the experiments with added delay. We do this (see Fig. 6) by computing the difference $F1-F2$ for each pair of CDFs plotted in Fig. 5, where $F1$ is a CDF from the shared bottleneck experiments and $F2$ is a CDF from the experiments with added delay. By computing the mean of the maxima of all combinations of $F1-F2$ plotted in the figure, we obtain $th2 = 0.25$. This threshold is plotted as a vertical line in Fig. 6.

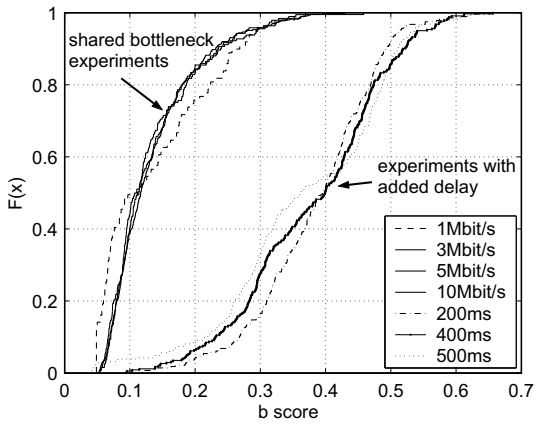

Fig. 5. CDF plots of the b-score when downloading multiple files simultaneously through a shared bottleneck link or with added delay.

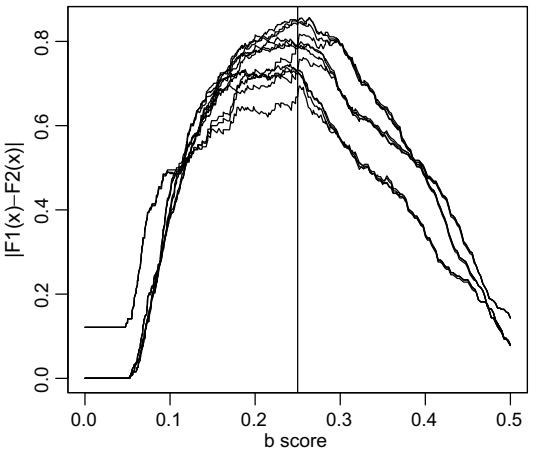

Fig. 6. Difference of CDF plots between experiments with an artificial bottleneck and added delay, results with 100 ms are excluded. The best matching threshold is found at 0.25 (vertical line).

7.3. Discussion

After having chosen the values for the thresholds, we checked whether our choices of threshold values lead to a correct classification of the controlled FTP downloads we performed. In particular, we wanted to see how well the downloads from a single server were separated by the threshold in the cases of a shared bottleneck or receiver window limitation. We also checked that there are no correlations between locations, e.g. we verified that we did not end up choosing mostly US servers in the experiments. This would lead to choosing a threshold value that only works for such transatlantic paths. The geographical locations of the servers were resolved using the IP2location service (http://www.ip2location.com). In addition, we classified all the traces from the controlled experiments using the chosen thresholds and further investigated the traffic classified as mixed/unknown. We observed that the latter represents between zero and approximately 25% of the total traffic depending on the particular data set. Detailed analysis results are presented in [14].

8. Application to ADSL client traffic

Usually, there will always be some misclassified periods of a connection, no matter how thorough the validation procedure is. Instead of doing more controlled experiments, we chose to demonstrate in this section how the root cause analysis techniques (RCA) can be applied to a large ADSL trace. This analysis also serves as a validation of our root cause analysis toolkit.

8.1. Measurement setup and dataset

A typical ADSL architecture is organized as follows (see Fig. 7): the broadband access server (BAS) aggregates the traffic issued from many digital subscriber line

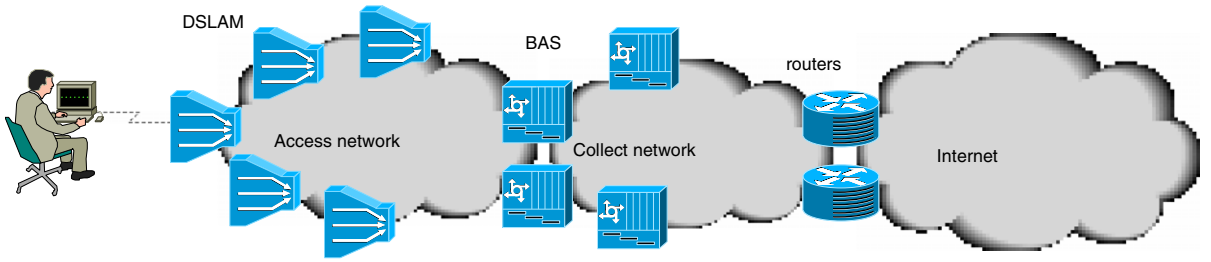

Fig. 7. Architecture of the ADSL platform monitored.

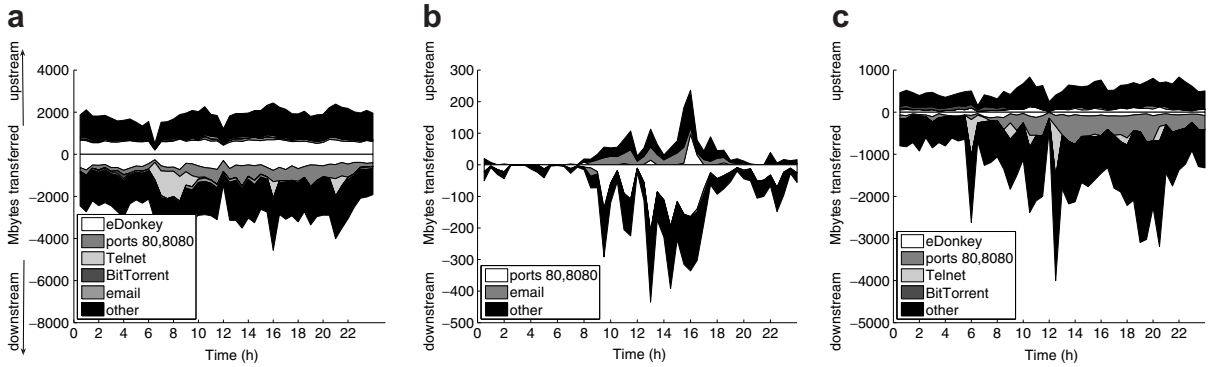

Fig. 8. Amount of bytes experiencing a particular root cause. Note the different scales. (a) Application limitation. (b) Access link saturation. (c) Network limitation due to a distant link.

access multiplexers (DSLAMs) before forwarding it through the two local routers to an IP backbone. Each client is connected to one DSLAM using one ATM Virtual Circuit. The traffic of a client is controlled by the up and down capacities of this access link. A variety of subscription types of the ADSL access service is defined through different combinations of up-link and down-link capacities.

We applied our root cause analysis toolkit to traffic captured via two probes located between a BAS and the first two routers of the IP backbone of an ADSL network. Each probe captures packets flowing through a single router. This BAS multiplexes the traffic of three DSLAMs and connects about 3000 clients to the Internet. We capture all IP, TCP and UDP packet headers going through the BAS without loss of packets. Trace files are stored in tcpdump format.

We collected one full day (Friday March 10, 2006) of traffic, which represents approximately 290 Gbyte of TCP traffic in total, out of which 64% is downstream and 36% upstream. This day can be considered as a typical day in terms of data volumes generated by the clients. Out of those 3000 clients, 1335 generated enough data to allow for a root cause analysis. We consider only those clients in further analysis. In addition to the packet trace, we have a list of IP addresses that belong to local clients, which allows us to distinguish the upstream traffic from the downstream traffic. However, we do not know the clients' subscription rates, i.e. their up-link and down-link capacities.

8.2. Throughput limitation causes experienced by major applications

In this section, we exemplify our root cause analysis toolkit on the aforementioned trace. Specifically, we investigate what are the most important applications that experience the different limitation causes, namely (i) application limited, (ii) saturated access link, and (iii) bottleneck at distant link. For each 30-min period, we associate bytes flagged with limitations to different applications based on the used TCP ports. Note that the amount of mixed/unknown causes (see Fig. 4) we obtained is negligible, i.e. a single limitation is associated to each BTP.

Fig. 8a shows the main applications that generate traffic that is application limited. If we look at the evolution of the total volume of traffic that is application limited, we see very little variation in time and an upload volume almost as big as the download volume, both being around 2 Gbyte per 30 min periods. The largest single application that generates application limited traffic is, as expected, a P2P application, namely eDonkey. However, if we look by volume, the largest category is "other", which contains traffic that we were not able to relate to any specific application. The overall symmetry of upload and download volumes for the "other" category as well as a manual analysis of the traffic of some heavy hitters strongly suggest that the "other" category contains a significant fraction of P2P traffic.

Fig. 8b shows the main applications that saturate the access link.[4] For this limitation cause, no traffic originating from recognized P2P applications was seen. Instead, a significant portion of traffic saturating the up-link is e-mail. For the down-link it is mainly traffic on ports 80 and 8080 and traffic for which the application could not be identified. The fact that the traffic using ports 80 and 8080 primarily saturates only down-link suggests that it could be real Web traffic that consists of small upstream requests and larger downstream replies from the server, as opposed to P2P traffic which is typically more symmetric. If we look at the absolute volumes, we see that most of the activity is concentrated during the day time, with the peak being in the early afternoon.

Fig. 8c shows the main applications that see their throughput limited by a link that is not the client's access link. The category other, which contains the applications that could not be identified, clearly dominates in terms of volume. Otherwise, we observe a mixture of applications. It was expected that this set of applications be diverse since this type of network limitation can occur at any point of the network, almost independently of the application behavior.

In the download direction, the total traffic that is limited by a distant bottleneck reaches in the late afternoon a proportion that, in terms of volume, is similar to the download traffic that is application limited. The fact that this traffic peaks late afternoon may be an indication of higher overall network utilization just after working hours, not only within the network we monitor but at a wider scale. Note that at the same time, the amount of traffic limited by the access link is very low (Fig. 8b), which could indicate that these two groups represent different types of clients.

Finally, we would like to point out that a comparison of the absolute traffic volumes of Fig. 8a–c reveal that the application limitation category represents the vast majority of the total number of transmitted bytes.

8.3. Closer look at an example client

Finally, we selected an interesting example client and studied it in more detail. The client selected is active the entire 24 h and belongs to the set of the top 15% clients in terms of number of bytes transmitted.

In order to visualize the activity of the client, we show *activity diagrams* that plot all connections of the client as rectangles on a coordinate system where the *x*-axis represents time and *y*-axis the throughput. The width of each rectangle represents the duration, the height represents the average throughput, and, consequently, the area represents the volume of the connection.[5] The starting time of the connection is the left boundary and the ending time the right boundary of the rectangle. Each rectangle is placed vertically as low as possible in such a way that no rectangle overlaps with another one. Upstream connections are plotted above *x*-axis and downstream connections below the *x*-axis. The fill pattern of the rectangle represents the application that generated the connection. The different applications are identified using the TCP port number. We selected only connections that transmit at least 10 Kbytes for the sake of clarity.

By analyzing the utilization of the access link for this client, we see that while being all the time active, the client transfers data with modest rates. However, there are occasional short periods of time when the client achieves high throughput. Fig. 9a shows three-hours of typical activity for this client. The same kind of activity persists throughout the day. We make two main observations: First, the client used mainly eDonkey, which produced the low

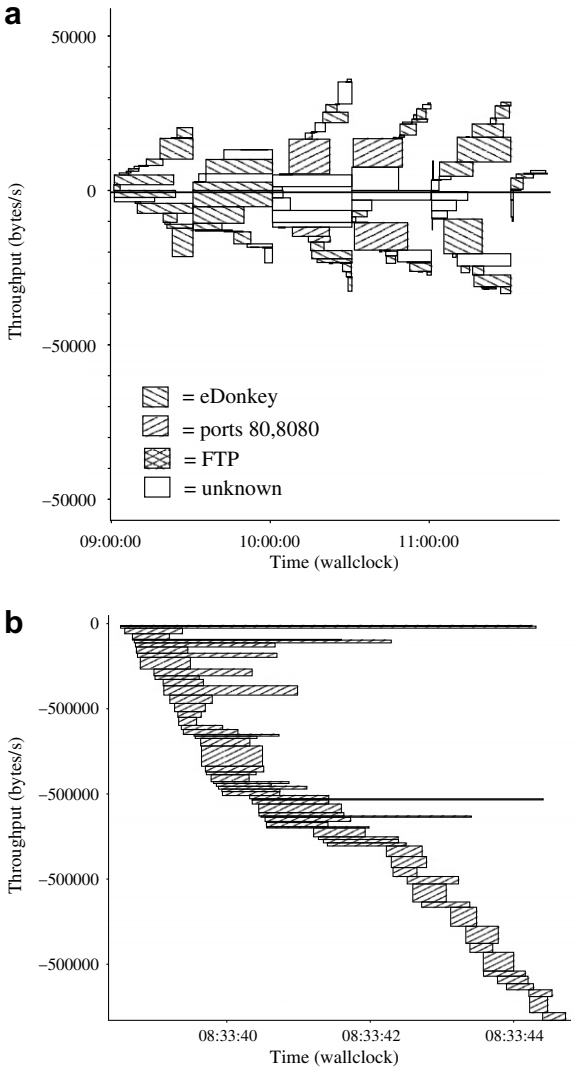

Fig. 9. Activity of a single client. (a) Three-hour piece of an activity plot. (b) Close up of the client's connections originating most likely from Web browsing that cause the peak download rates.

[4] We estimate the saturation of an access link by comparing the throughput of a BTP with the access link capacity estimated using PPrate.

[5] Note that we only show the scale for *y*-axis but not cumulative values because the absolute vertical position of the upper boundary of a particular rectangle does not necessarily signify the instantaneous throughput. In other words, these plots are different from the area plots in Fig. 8, as they do not represent the *cumulative* throughput.

throughput activity persisting throughout the day. We can visually identify from the plot a typical size of an eDonkey connection by looking at the areas of the rectangles. This size is approximately 9 Mbyte, which is the typical chunk size for eDonkey transfers, i.e. each file larger than that is divided into 9 Mbyte chunks. If we compare the areas of the rectangles representing eDonkey connections to the rectangles of port 80/8080 and unknown traffic, we see that those connections are often of the same size. Moreover, many of these port 80/8080 and unknown transfers are upstream. Thus, as a second remark, there is probably a significant amount of eDonkey traffic disguised as Web traffic or that is not using the standard port numbers (6881–6889). Just before the activity represented in Fig. 9a, the client had different activities. The client produced occasional fast downloads originating from port 80/8080 that caused peak rates for this client. We zoomed in to one of these periods, which took place at around 8h30. The activity diagram is in Fig. 9b. Note that the scale of the *y*-axis is more than ten times larger in this plot than in the plot of Fig. 9a (500 vs. 50 Kbytes/s steps). We see that it contains several simultaneous connections and when one ends there is another one that starts. The largest connections carry around 100 Kbytes. Thus, it seems likely to be a download of a Web page using parallel and persistent (HTTP/1.1) connections. We looked at the root cause analysis results for this client that showed that the clear majority of the bytes (most of the time more than 85%) uploaded and downloaded by this client are rate limited by the application on top. However, some port 80/8080 traffic such as the occasional fast downloads like the one enlarged in Fig. 9b were rate limited by the network. These results are illustrative of the trends for different applications that we saw in Fig. 8: most eDonkey traffic is rate limited by application while "true" Web traffic tends to be network limited and the major cause for access link saturation.

9. Conclusions

Knowledge about factors that limit TCP throughput is useful for the development and the optimization of (i) the applications operating on top of TCP and (ii) the TCP protocol itself. It also helps network operators for troubleshooting purposes.

In this paper, we showed how to infer these rate limitation causes for each TCP connection from packet traces collected at a single measurement point. We described a set of algorithms that we designed and implemented in a TCP root cause analysis toolkit. This toolkit adopts a two phase "divide-and-conquer" approach: The first phase filters out those periods of a connection that are rate limited by the application. The second phase, which is the focus of this paper, analyzes the remaining bulk transfer periods (BTP) for TCP and IP level limitation causes. We also illustrated how this toolkit could be applied in a case study on the performance analysis of ADSL clients.

Our root cause analysis toolkit is quite mature and has been extensively validated and evaluated. What remains to be done is to extend the work to include short connections.

Currently, the toolkit is used for off-line analysis of stored packet traces. Another interesting thread of work would be to try to redesign the algorithms in such a way that they could be run on-line, i.e. while observing the traffic and without the need to store the traffic in trace files. Such modifications might need to trade off some of the accuracy for the sake of performance.

Acknowledgement

This work has been partly supported by France Telecom, Project Number CRE-46126878.

References

[1] Nist net home page: <http://www-x.antd.nist.gov/nistnet/>.
[2] Official mirror sites for the fedora core linux distribution: <http://fedora.redhat.com/download/mirrors.html>.
[3] Rshaper: <http://freshmeat.net/projects/rshaper/>.
[4] C. Barakat, E. Altman, Performance of short TCP transfers, in: Proceedings of the Networking, 2000.
[5] E. Brosh, G. Lubetzky-Sharon, Y. Shavitt, Spatial-temporal analysis of passive TCP measurements, in: Proceedings of IEEE Infocom'05, 2005.
[6] W. Chen, Y. Huang, B.F. Ribeiro, K. Suh, H. Zhang, E. de Souza e Silva, J.F. Kurose, D.F. Towsley, Exploiting the IPID field to infer network path and end-system characteristics, in: Proceedings of Passive and Active Network Measurement (PAM), 2005.
[7] T. En-Najjary, G. Urvoy-Keller, Pprate: A passive capacity estimation tool, in: Proceedings of IEEE/IFIP Workshop on End-to-End Monitoring Techniques and Services. URL <http://www.eurecom.fr/btroup/BPublished/pprate.pdf>.
[8] N. Hu, L.E. Li, Z.M. Mao, P. Steenkiste, J. Wang, Locating internet bottlenecks: algorithms, measurements, and implications, in: Proceedings of ACM SIGCOMM 2004 Conference, 2004.
[9] S. Jaiswal, Measurements in the middle: Inferring end–end path properties and characteristics of TCP connections through passive measurements, Ph.D. Thesis, University of Massachusetts, Amherst, 2005.
[10] 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: Proceedings of IEEE Infocom'03, 2003.
[11] K. Lakshminarayanan, V.N. Padmanabhan, J. Padhye, Bandwidth estimation in broadband access networks, in: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement, ACM Press, New York, NY, USA, 2004.
[12] M. Mathis, J. Heffner, R. Reddy, Web100: extended TCP instrumentation for research, education and diagnosis, Comput. Commun. Rev. 33 (3) (2003) 69–79. URL <http://www.psc.edu/mathis/papers/mathis03web100.pdf>.
[13] R.S. Prasad, M. Murray, C. Dovrolis, K. Claffy, Bandwidth estimation: metrics, measurement techniques, and tools, IEEE Network 17 (6) (2003) 27–35.
[14] M. Siekkinen, Root cause analysis of TCP throughput: Methodology, techniques, and applications, Ph.D. Thesis, Institut Eurécom/Université de Nice-Sophia Antipolis, Sophia Antipolis, France (Oct. 2006). URL <http://www.ifi.uio.no/siekkine/pub.html>.
[15] M. Siekkinen, D. Collange, G. Urvoy-Keller, E.W. Biersack, Performance limitations of ADSL users: A case study, in: Proceedings of the Eighth Passive and Active Measurement Conference (PAM), 2007.
[16] M. Siekkinen, G. Urvoy-Keller, E. Biersack, T. En-Najjary, Root cause analysis for long-lived TCP connections, in: Proceedings of CoNEXT, 2005. URL <http://www.eurecom.fr/btroup/BPublished/siekkinen05conext.pdf>.
[17] M. Siekkinen, G. Urvoy-Keller, E.W. Biersack, On the interaction between internet applications and TCP, in: Proceedings of the 20th International Teletraffic Congress (ITC20), 2007.
[18] R. Teixeira, J. Rexford, A measurement framework for pin-pointing routing changes, in: NetT'04: Proceedings of the ACM SIGCOMM workshop on Network troubleshooting, 2004.
[19] B. Veal, K. Li, D. Lowenthal, New methods for passive estimation of TCP round-trip times, in: Proceedings of Passive and Active Measurements (PAM), 2005. URL <http://www.pam2005.org/PDF/34310124.pdf>.

[20] Y. Zhang, L. Breslau, V. Paxson, S. Shenker, On the characteristics and origins of internet flow rates, in: Proceedings of ACM SIGCOMM 2002 Conference, Pittsburgh, PA, USA, 2002.
[21] Y. Zhang, L. Qiu, Speeding up short data transfers: Theory, architectural support, and simulation results, in: The 10th International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV 2000), 2000.

Matti Siekkinen (siekkine@ifi.uio.no) obtained his MSc in computer science from the Helsinki University of Technology, Finland, in 2003. He completed his Ph.D. in 2006 from Universite de Nice – Sophia Antipolis, France, after three years of research work at Institut Eurecom. He is currently at University of Oslo, Norway, as a postdoctoral researcher. His research interests are network architectures, network measurements and monitoring, and data management.

Guillaume Urvoy-Keller received the Engineer Degre from Institut National des Télécommunications, France, in 1995, the Ph.D. degree in Computer Science from University Paris 6 in 1999 and the "Habilitation à Diriger des Recherches" from University of Nice Sophia Antipolis in 2006. Since 2000, he is an Assistant Professor at Institut Eurecom. His research interests include traffic analysis, peer-to-peer and quality of service.

Ernst Biersack received his M.S. and Ph.D. degrees in Computer Science from the Technische Universität München, Munich, Germany and his habilitation from the University of Nice, France. Since March 1992 he has been a Professor in Telecommunications at Institut Eurecom, in Sophia Antipolis, France. His current research is on Peer-to-Peer Systems, Network Tomography, and LAS Schedling in Edge Routers. Among other awards, he has received (together with J. Nonnenmacher and D. Towsley) the 1999 W.R. Bennet Award of the IEEE for the best paper published in the year of 1998 in the ACM/IEEE Transactions on Networking.

Denis Collange is a Research Engineer in Orange Labs (France Telecom R&D). He graduated from Telecom Paris. His research interests include performance evaluation of network protocols, traffic modeling and performance troubleshooting.