Two-way TCP Connections: Old Problem, New Insight

Martin Heusse*, Sears A Merritt†, Timothy X Brown†, Andrzej Duda*
* Grenoble Institute of Technology, CNRS Grenoble Informatics Laboratory UMR 5217, France
[martin.heusse, andrzej.duda]@imag.fr
†University of Colorado at Boulder, USA
[sears.merritt, timxb]@colorado.edu

ABSTRACT

Many papers explain the drop of download performance when two TCP connections in opposite directions share a common bottleneck link by *ACK compression*, the phenomenon in which download ACKs arrive in bursts so that TCP self clocking breaks. Efficient mechanisms to cope with the performance problem exist and we do not consider proposing yet another solution. We rather thoroughly analyze the interactions between connections and show that actually ACK compression only arises in a perfectly symmetrical setup and it has little impact on performance. We provide a different explanation of the interactions—*data pendulum*, a core phenomenon that we analyze in this paper. In the *data pendulum* effect, data and ACK segments alternately fill only one of the link buffers (on the upload or download side) at a time, but almost never both of them. We analyze the effect in the case in which buffers are structured as arrays of bytes and derive an expression for the ratio between the download and upload throughput. Simulation results and measurements confirm our analysis and show how appropriate buffer sizing alleviates performance degradation. We also consider the case of buffers structured as arrays of packets and show that it amplifies the effects of *data pendulum*.

Categories and Subject Descriptors: C.2.2 [**Computer-communication networks**]: Network Protocols; C.2.5 [**Computer-communication networks**]: Local and Wide-Area Networks
General Terms: Performance
Keywords: TCP, asymmetric links

1. INTRODUCTION

We consider the problem of interactions between two TCP connections that generate two-way traffic in opposite directions over a common bottleneck link. Several authors analyzed the TCP dynamics in this configuration [1, 2, 3, 4] and others extended the investigations to an asymmetric link [5, 6, 7, 8, 9]. They observed how data transfers often suffer from poor performance: for instance, a download connection on a typical ADSL line may only fill one tenth of the downlink capacity in the presence of uploads. The performance problem has recently become fairly important with widespread deployment of access networks such as ADSL and increasing upload P2P traffic. It is also critical for cable modems as well as for satellite and wireless links. We consider the case of a *moderate* asymmetry, where the capacity in the uplink direction is always sufficient to accommodate the download ACKs [10].

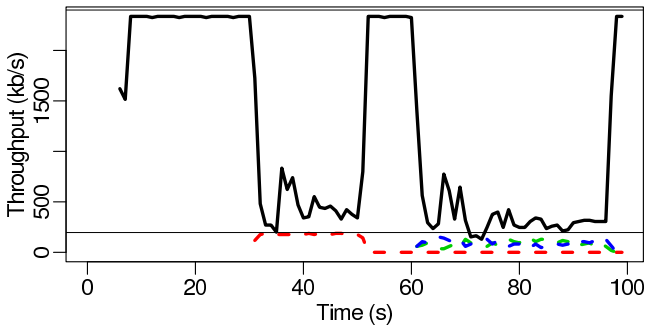


Figure 1: TCP throughput on an asymmetric link with buffers of 30,000 B on both ends. Download on a 2.4 Mbps link (solid line) and uploads on a 196 kbps link (dashed lines: one upload from 30s to 50s, then two uploads from 60s to 95s).

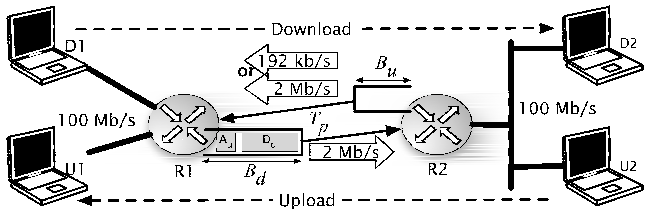
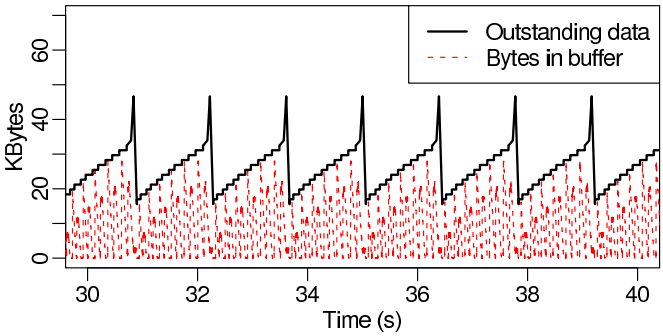


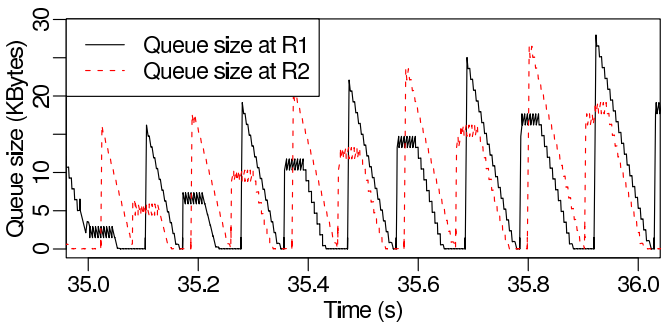
Figure 2: Bottleneck link with download and upload TCP connections.

To illustrate the importance of the performance problem, let us consider the following example. Figure 1 presents the evolution of the TCP throughput over an asymmetric link with 2.4 Mbps download and 196 kbps upload transmission rates: a download (solid line) starts at instant 10s, a single upload (dashed line) is active between 30s and 50s, and two uploads (dashed lines) send traffic between 60s and 95s. The buffers in both directions are set to a typical value of 30,000B corresponding to 20 full size segments. We can observe that the download without any upload can achieve nearly 2.4 Mbps. However, whenever an upload is present, the download throughput drops and oscillates around the upload throughput close to 196 kbps.

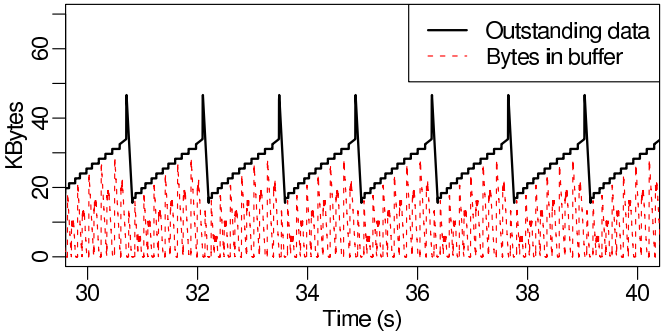
Efficient techniques for fixing the performance problem exist and this paper is not about proposing yet another solution. For instance, RFC 3449 recommends several up-



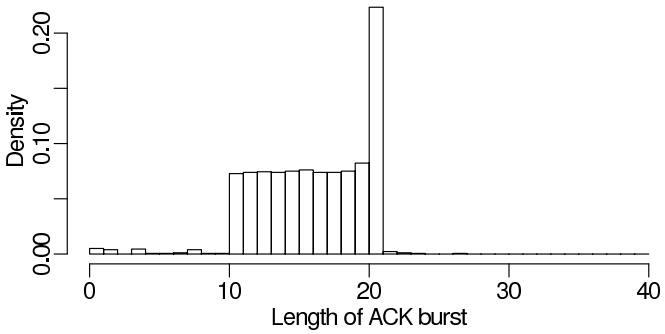
3.1: Outstanding data of D1 and queue size at R1



3.2: Zoom in on the queue size from Figures 3.1 and 3.3, in phase oscillations



3.3: Outstanding data of U2 and queue size at R2



3.4: Histogram of the ACK burst length, symmetric case, strong ACK compression

Figure 3: Outstanding data and queue size for a *symmetric* setup, equal buffer sizes (30000 bytes), 1 ms delay between both U1 and R1 as well as between D1 and R1.

stream link scheduling disciplines (Weighted Fair Queueing, Class Based Queueing) that all effectively isolate the download ACKs from data segments [7]. Louati et al. proposed other more sophisticated mechanisms for scheduling the upload link [8] and Brouer used the Linux Hierarchical Token Bucket to optimize TCP performance [9].

However, we do not yet thoroughly understand why the performance problem arises and in what conditions. Conventional wisdom explains its cause by the *ACK compression* phenomenon that affects TCP self clocking by clustering data segments and ACKs of both connections [1, 2, 3, 4]. The authors defined ACK compression as the arrival of download ACKs in bursts after being queued behind data segments on the uplink. In the presence of two-way TCP traffic, the receiver of download data segments generates ACKs at the downlink departure rate. Many ACKs may be generated during one segment transmission on the uplink and because of FIFO scheduling they will all reach the sender in a burst, which in turn causes bursty transmissions at the sender.

In this paper, we revisit the performance problem and provide a different explanation of its roots. Contrary to the common knowledge, we show that ACK compression is in fact a rare phenomenon and has little impact on performance. We find out instead that TCP connections in opposite directions exhibit a *data pendulum* effect in which data and ACK segments alternately fill only one of the link buffers (upload or download), but almost never both of them. Consequently, only one link can be fully utilized at any instant, which is the actual root of the interactions. Fundamentally, the TCP connections fill only one buffer, because one of them always induces more queueing delay than the other, so that all data or ACKs end up in a single buffer. Another way of looking at this behavior is to consider that both connections attempt to track the bandwidth-delay product of the system. The product depends on the state of one connection that the other one tries to follow and vice versa, which results in unintended interactions [10].

We analyze the influence of buffer sizing on the data pendulum effect: when the size of buffers is not correctly set, the data pendulum effect results in significant performance degradation. In particular, we note that bigger buffers are not always the right solution. Another important issue is the structure of buffers—it turns out that a buffer taking form of an array of bytes is beneficial compared to a buffer managed as an array of packets.

The next section (Section 2) details the case of a symmetric link in which ACK compression may occur, although it does not have any consequence on the connection performance. Then, we introduce the data pendulum effect that appears in the same set up—we analyze its roots and model the consequences (Section 3). Sections 4 and 5 confront our findings with simulation results and measurements.

2. INTERFERENCE BETWEEN TCP CONNECTIONS IN OPPOSITE DIRECTIONS

To analyze the interactions between upload and download TCP connections, we consider the setup presented in Figure 2. We first use simulation to isolate the data pendu-

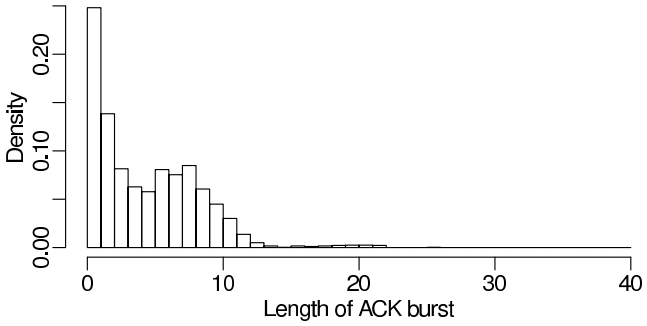


Figure 4: Histogram of the ACK burst length, symmetric link, symmetric delays, delayed ACKs. No ACK compression.

lum effect in a highly controlled environment and then provide some measurement results. TCP endpoints connect to routers that transmit over a link with propagation time T_{p}. We examine two cases: *symmetric* 2.4 Mbps/2.4 Mbps and *asymmetric* 2.4 Mbps/196 kbps one. We consider two independent TCP connections with data traffic in one direction, so there is no piggybacking of ACKs onto data segments. If not stated otherwise, the TCP variant is New Reno with selective acknowledgements. We use the Qualnet network simulator [11].

The reception of a download data segment triggers the emission of an ACK segment that enters the upstream queue behind an upload data segment. The reception of an upload data segment and the ACK triggers in turn the emission of a new download data segment and a new ACK that simultaneously enter the downstream queue. Thus, it is clear that both connections experience similar round trip times (RTT). Note that, if the TCP source and destination are disjoint, this holds as long as RTT is dominated by the sum of propagation time T_{p} and the waiting time in the buffers at both ends of the bottleneck link. The connection throughputs are approximately $X_x = \frac{cwnd_x}{RTT}$, where $cwnd_x$ is the congestion window of a given connection (x refers to the uplink or downlink). We denote by $r = \frac{X_d}{X_u}$ the ratio between the download and upload throughputs.

2.1 Elusiveness of ACK compression

We start with an analysis of the dynamic behavior of TCP connections in the setup of Figure 2. Let us consider a perfect *symmetric* case with 2.4 Mbps bit rates in both directions, the same buffer sizes at both ends of the link (30000 bytes), $T_{\mathrm{p}} = 0$, and 1 ms delay between the endpoints and the routers.

Figure 3 presents the evolution in time of two variables: the outstanding data at each TCP sender (the total number of unacknowledged bytes, which reflects the TCP congestion window) and the queue size at each router (the total number of bytes in a buffer). For both connections, we can observe a typical increase of the outstanding data in the congestion avoidance phase until a drop after a fast recovery phase due to buffer saturation. Note that both directions are synchronized—their phases of increase and drop occur almost at the same time.

The queue size is subject to quick oscillations and if we zoom in, we can see that the oscillations are in phase (cf. Figure 3.2): when one buffer grows, the other one grows too. Similarly, buffers shrink at the same time. This is typical of ACK compression—in fact, if the buffers grew and shrank out of phase, then the ACKs of the data segments freed from one buffer would mix with data segments accumulating in the buffer in the opposite direction, which is contradictory with the definition of ACK compression. The phenomenon has a negative impact on performance, because at times, queues are empty and the link is idle. This is the case of "in phase" TCP oscillations studied before [2].

Figure 3.4 presents a histogram of the ACK burst length—how many ACK segments appear between data segments in a buffer. We can see that ACKs arrive in large bursts of 10 to 20 segments, which provides a quantitative evidence for the presence of ACK compression in the perfect symmetric case. It mostly disappears by simply activating the delayed ACK mechanism in which an ACK is generated for every other segment received in sequence. The corresponding histogram of the ACK burst length appears in Figure 4—in spite of this difference between the histograms, the evolution of the TCP connections remain perfectly similar to the one in Figure 3.

This example shows that although we can observe the ACK compression effect in a particular setup, a small change in parameters makes it vanish without any change in the connection dynamics. In the subsection below, we consider a case without ACK compression in which another effect emerges, the data pendulum.

2.2 Data pendulum

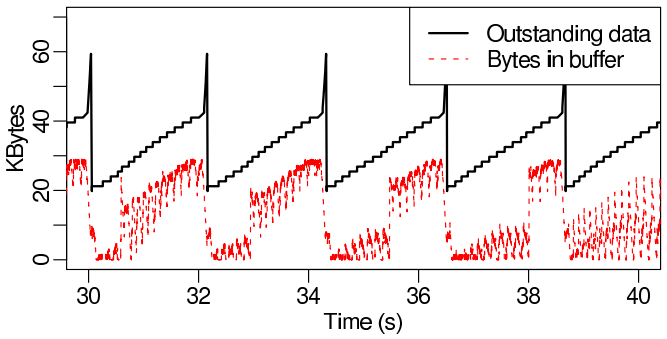
Figure 5 shows the corresponding diagrams to Figure 3 for a symmetric link, but with slightly different delays for both connections. The connections are not synchronized any more and the queue sizes exhibit opposite behavior on a longer time scale: when the queue size at R1 drops around $t = 32$ s, the queue size at R2 grows. Immediately before $t = 33$ s, the situation is reversed: the queue at R2 drops and that of R1 fills up. This slowly varying behavior with a period of around 2s in our case is distinctive of data pendulum: when one queue holds some segments during a significant period of time with respect to RTT, the other queue is empty and vice versa. The result is that when one buffer overflows, the other is empty.

Fast oscillations presented in Figure 5.2 superimpose on the slowly varying pattern due to data pendulum. This is caused by small bursts of ACKs and segments, although the shape of the histogram in Figure 5.4 is much different from Figure 3.4: we can observe that ACKs are very well interspersed between data segments. It is a regular behavior of TCP to generate short term bursts of data segments.

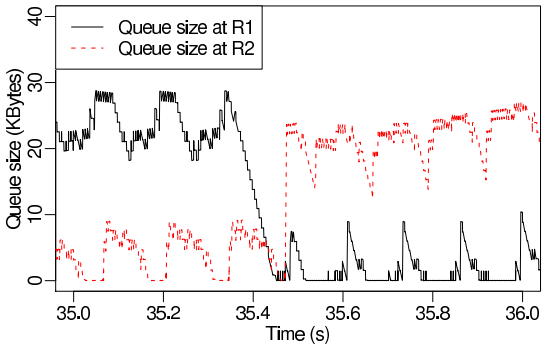
The quick oscillations in Figures 5.1 and 5.3 have little impact on the effective utilization of the links: as long as there are some packets in its queue, a link remains active and well utilized. On the contrary, an empty buffer is a good indicator of possible low link utilization, as it requires a perfect pacing of packets to reach the buffer at a rate that closely matches link capacity with no extra segments in the buffer. This is effectively unlikely: TCP self clocking always requires some data in the bottleneck buffer, so small bursts caused, for instance by the delayed ACK mechanism, are ironed out.

2.3 Asymmetric link

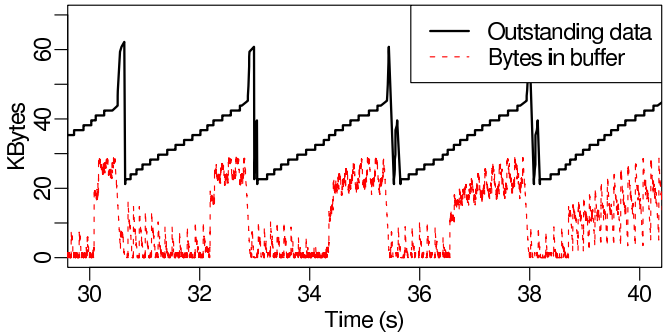
We now consider an *asymmetric* case with 2.4 Mb/s downlink and 196 kb/s uplink, 30000 B downlink buffer and 9000 B



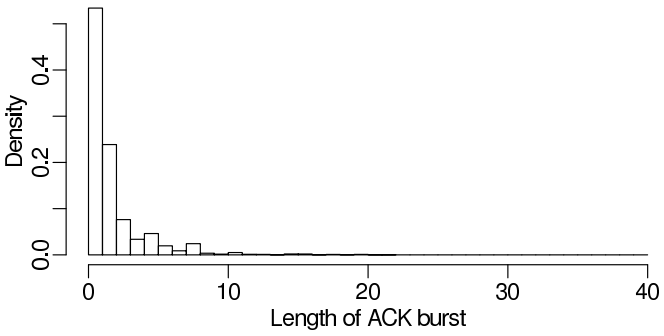
5.1: Outstanding data of D1 and queue size at R1



5.2: Queue size at R1 and R2



5.3: Outstanding data of U2 and queue size at R2



5.4: Histogram of the ACK burst length, no ACK compression

Figure 5: Outstanding data and queue size at both ends of a *symmetric* link, 1ms delay between D1 and R1, 5ms between U1 and R1.

uplink one. Figures 6 and 7 present the outstanding data, queue size, and the histogram of the ACK burst length. The queueing behavior is more complex than for the previous symmetric case, but we can still observe the pendulum effect—after R1 overflows, the queue at R1 remains low while that of R2 grows, which often causes it to overflow. The downlink is used at roughly 50% of the capacity, which corresponds to a rather low level of interference between connections.

Finally, Figures 8 and 9 presents the corresponding diagrams for our introductory case shown in Figure 1 (asymmetric link capacities, equal buffer size of 30000 B) that exhibits significant harmful interactions because of the large buffer size (30000 B) at the head of the uplink. In this case, the uplink buffer is always busy, whereas the downlink one is often empty. All queueing takes place in the uplink buffer and the downlink is very much underutilized.

The histograms of the ACK burst length for the asymmetric cases appear in Figures 7 and 9. Comparing to the histogram in Figure 5.4, we can observe a slightly increased length of ACK bursts, although the general shape of the histograms is still much different from the pure symmetric case, which indicates the absence of ACK compression. There is a slight increase in the length of ACK bursts in the asymmetric case with different buffer sizes (cf. Figure 7), but it is not surprising: when both links are mostly well utilized, there are much less data segments in the uplink buffer to intermingle with ACKs. So, longer ACK bursts are in this case a sign of a low level of interactions between the upload and the download.

3. ANALYSIS OF THE DATA PENDULUM

We want to analyze the dynamics of the data pendulum effect in a general case of possibly asymmetric link capacities C_d and C_u with $C_\mathrm{d} = k \times C_\mathrm{u}$ and generally $k \geq 1$. To simplify expressions, we assume that the capacities are in Mbytes/s. We consider buffers in bytes and denote by d_d and d_u the number of data bytes continuously present in the downlink and uplink buffers, respectively, over one RTT. We neglect the amount of data in flight on the link ($T_\mathrm{p} = 0$) and we consider at first that the ACK size is negligible compared to the size of data segments. Then,

$$RTT = \frac{d_\mathrm{d}}{C_\mathrm{d}} + \frac{d_\mathrm{u}}{C_\mathrm{u}}. \tag{1}$$

Consider the download congestion window $cwnd_\mathrm{d}$: it is an interval of sequence numbers representing d_d bytes waiting in the downlink buffer and the amount of data sent over the downlink during the delay required for each download ACK to cross the upload buffer. This delay is $\frac{d_\mathrm{u}}{C_\mathrm{u}}$. So, we have the following relations:

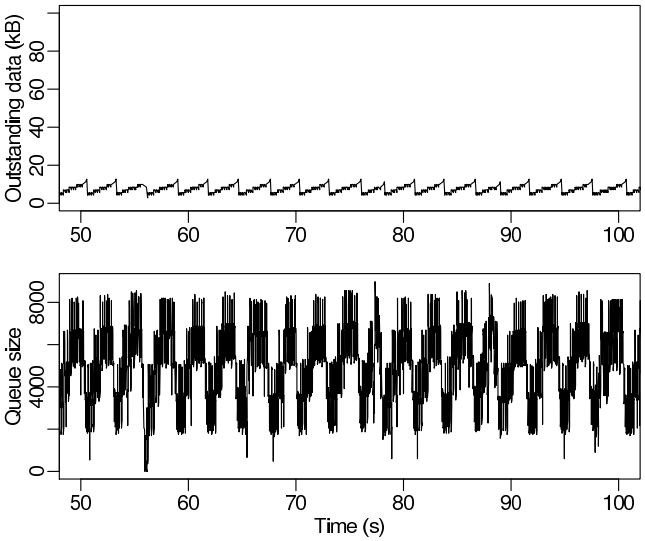
$$cwnd_\mathrm{d} = d_\mathrm{d} + \frac{d_\mathrm{u} C_\mathrm{d}}{C_\mathrm{u}} \tag{2}$$

and similarly,

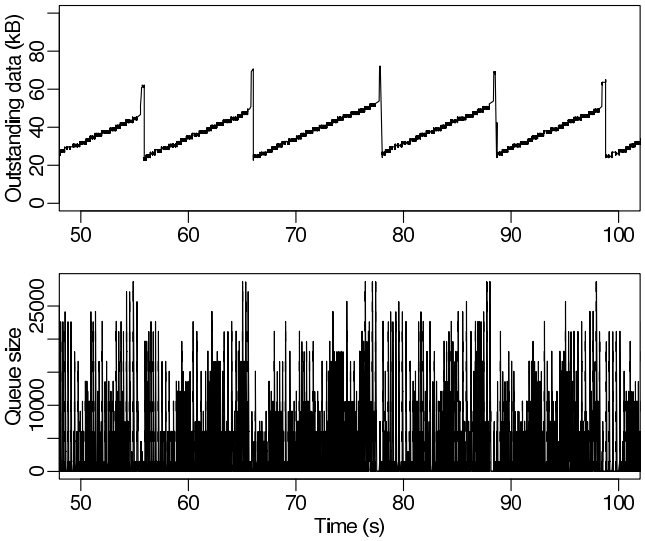
$$cwnd_\mathrm{u} = d_\mathrm{u} + \frac{d_\mathrm{d} C_\mathrm{u}}{C_\mathrm{d}}. \tag{3}$$

Using Eq. 2 in Eq. 1 simply yields that either

$$RTT = \frac{cwnd_\mathrm{d}}{C_\mathrm{d}} \tag{4}$$



6.1: Upload (R2)



6.2: Download (R1)

Figure 6: Outstanding data (*cwnd*) and queue size at R1 and R2, asymmetric link, different buffer sizes (9000 and 30000 bytes).

or

$$RTT = \frac{cwnd_{\mathrm{u}}}{C_{\mathrm{u}}}. \quad (5)$$

This implies three exclusive cases:

1. if $d_{\mathrm{d}} = 0$, then the queueing takes place only in the uplink buffer and:

$$RTT = \frac{cwnd_{\mathrm{u}}}{C_{\mathrm{u}}}, X_d = \frac{cwnd_d}{cwnd_u} C_{\mathrm{u}}, X_u = C_{\mathrm{u}};$$

2. if $d_{\mathrm{u}} = 0$, then packets are mostly in the downlink buffer,

$$RTT = \frac{cwnd_{\mathrm{d}}}{C_{\mathrm{d}}}, X_d = C_{\mathrm{d}}, X_u = \frac{cwnd_u}{cwnd_d} C_{\mathrm{d}};$$

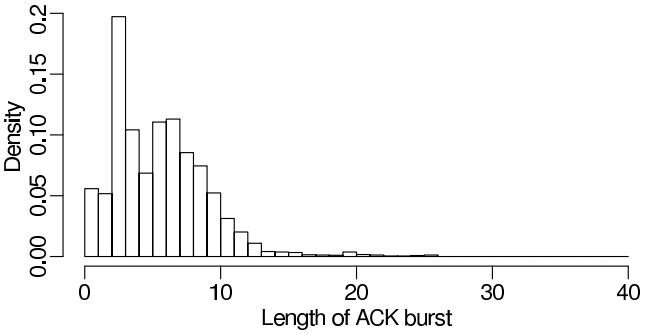


Figure 7: Histogram of the ACK burst length, asymmetric link, different buffer sizes.

3. Finally, the case of $d_{\mathrm{u}} \neq 0$ and $d_{\mathrm{d}} \neq 0$ only happens if

$$\frac{cwnd_{\mathrm{u}}}{C_{\mathrm{u}}} = \frac{cwnd_{\mathrm{d}}}{C_{\mathrm{d}}},$$

thus

$$X_d = C_{\mathrm{d}} \text{ and } X_u = C_{\mathrm{u}}.$$

Note that to have the last case, the ratio of the congestion windows should be exactly equal to the ratio of link capacities. Unfortunately, TCP senders cannot maintain this relation over time.

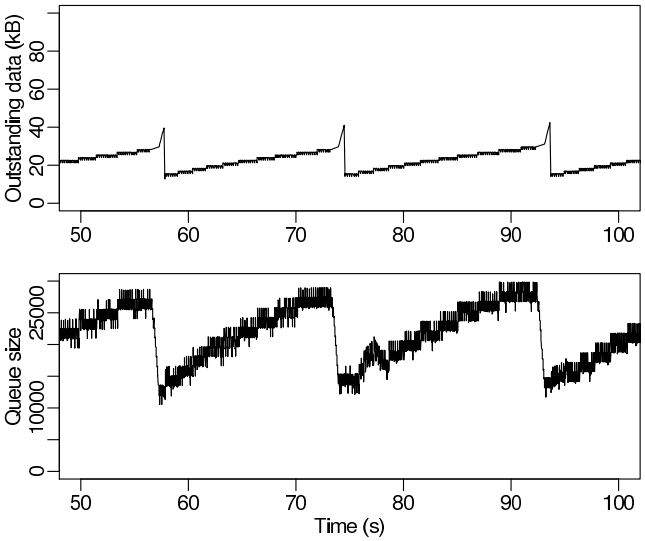
In Figure 3, the system is in case 3, which can only arise in a perfectly symmetric setup. Figure 5 presents the dynamics of TCP connections when they alternate between case 1 and 2: when the buffer at R1 overflows and becomes empty, the system is in case 1, then the buffer at R2 overflows and the system goes into case 2.

Relations 1–2 explain the *data pendulum* effect—there is an alternation between cases 1 and 2, because it is highly unlikely that the congestion windows at TCP senders satisfy the relation required for case 3. The pendulum represents the situation in which the download buffer is filled while the other one is empty and it changes to the opposite case with the upload buffer filled and the download one empty.

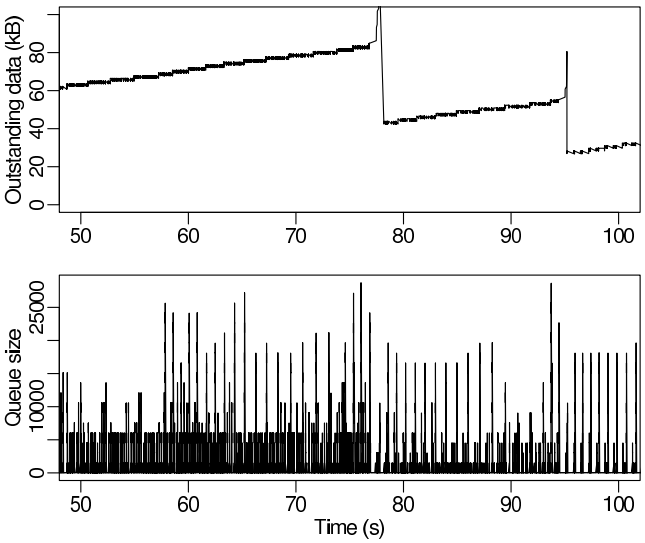
Let us find for which conditions the link is in a given case. We have seen that in case 3 $d_{\mathrm{u}} \neq 0$ and $d_{\mathrm{d}} \neq 0$ if $\frac{cwnd_{\mathrm{u}}}{C_{\mathrm{u}}} = \frac{cwnd_{\mathrm{d}}}{C_{\mathrm{d}}}$. We can observe that if $\frac{cwnd_{\mathrm{u}}}{C_{\mathrm{u}}} < \frac{cwnd_{\mathrm{d}}}{C_{\mathrm{d}}}$, the upload TCP sender takes less time to transfer its window than the download sender. Thus, packets fill the downstream buffer, which corresponds to $d_{\mathrm{u}} = 0$ (case 2). For $\frac{cwnd_{\mathrm{u}}}{C_{\mathrm{u}}} > \frac{cwnd_{\mathrm{d}}}{C_{\mathrm{d}}}$, the situation is reversed; so it corresponds to $d_{\mathrm{d}} = 0$ (case 1).

The results are expected from the observation of the data pendulum effect: queueing happens on the side of the connection that is more intensively filling its buffer with respect to the available capacity. If ever the buffers have sizes of the same order of magnitude, then both connections end up with similar congestion windows. On an asymmetric link, this results in the constantly non-empty uplink buffer as in Figure 8.1.

An important consequence of these results is that unless the senders maintain the proper ratio between their congestion windows *cwnd*, one of the links is not fully used. Unfortunately, two TCP senders cannot keep their congestion windows proportional over an asymmetric link, because the growth of the congestion windows is the same in both



8.1: Upload (R2)



8.2: Download (R1)

Figure 8: Outstanding data (*cwnd*) and queue size at R1 and R2. Equal buffer size (30,000B) on both sides. The downlink buffer remains empty for significant periods of time, whereas the uplink buffer never empties. The case corresponds to the throughput in Figure 1.

directions since it only depends on the RTT. So it cannot be maintained equal to the ratio of the link capacities. Thus, it is impossible to keep both (half-) links continuously active.

3.1 Throughput ratio under the pendulum effect

Based on the relations explaining the data pendulum effect, we want to obtain simple expressions for the connection throughput. In fact, if this phenomenon accurately explains the dynamics of the system, it means that only one buffer may hold a significant number of bytes for some significant

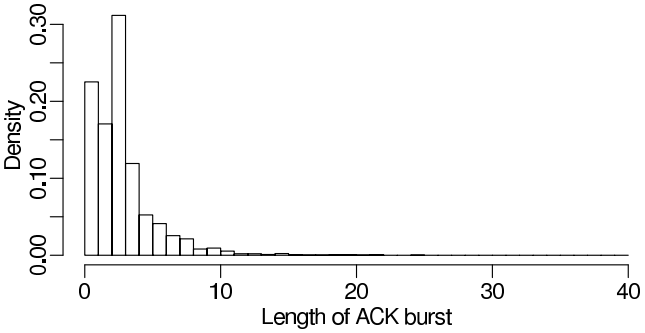


Figure 9: Histogram of the ACK burst length, asymmetric link, equal buffer sizes.

period of time. This allows us to model the system behavior, predict its performance, and find out the relevant parameters.

We focus on a single measure that characterizes the download and upload throughput—their ratio $r = \frac{X_d}{X_u}$. As the performance problem more often impacts the download, if the throughput ratio shows that the upload and the download get the same throughput, there is no doubt that the download is strongly hit independently of whether or not the upload gets a decent link utilization. The throughput ratio does not capture the fact that both links could be theoretically empty, but this would mean that the TCP senders do not manage to fill any buffer in the system, which contradicts their most basic target (as long as delays are reasonable).

So, we will consider ratio r as a function of uplink and downlink buffer sizes (B_u and B_d) as well as the amount of data in flight ($P_\mathrm{u} = 2C_\mathrm{u}T_\mathrm{p}$ and $P_\mathrm{d} = 2C_\mathrm{d}T_\mathrm{p}$) under the pendulum effect. The uplink buffer size limits the growth of $cwnd_\mathrm{u}$: the amount of data in flight follows a sawtooth curve that culminates when the buffer overflows, so $cwnd_\mathrm{u}$ oscillates between $(B^*_\mathrm{u}+P_\mathrm{u})/2$ and $B^*_\mathrm{u}+P_\mathrm{u}$, where B^*_u is the uplink buffer space left by download ACKs. Similarly, the downlink buffer size limits the throughput of the download connection, so the mean values of congestion windows are as follows:

$$\overline{cwnd}_\mathbf{x} = \begin{cases} B_\mathrm{TCP}, & \text{if } B^*_\mathrm{x} + P_\mathrm{x} > B_\mathrm{TCP} \\ \frac{3}{4}(B^*_\mathrm{x} + P_\mathrm{x}), & \text{otherwise;} \end{cases} \quad (6)$$

where B_TCP is the minimum of the TCP buffer sizes at the sender and the receiver.

This simple expression results from taking into account the data pendulum effect, because it only considers the data in the buffer in one direction and in flight on the link, but not in the buffer in the opposite direction.

If ρ denotes the ratio of the ACK size to the data segment size, ($\rho \approx \frac{1}{30}$ for ACKs with timestamps), we obtain the following expression for B^*_u in the case of delayed ACKs:

$$B^*_\mathrm{u} = B_\mathrm{u} - \rho\ \overline{cwnd}_\mathrm{d}/2 \quad (7)$$

and

$$B^*_\mathrm{u} = B_\mathrm{u} - \rho\ \overline{cwnd}_\mathrm{d} \quad (8)$$

otherwise. Note that, following the pendulum effect, we do not take into account that in principle, some ACKs may be standing in the downlink queue at the instant the uplink buffer overflows.

Similarly, the download buffer space left by the uplink ACKs in the case of delayed ACKs is:

$$B_{\mathrm{d}}^{*} = B_{\mathrm{d}} - \rho\ \overline{cwnd}_{\mathrm{u}}/2 \tag{9}$$

and

$$B_{\mathrm{d}}^{*} = B_{\mathrm{d}} - \rho\ \overline{cwnd}_{\mathrm{u}} \tag{10}$$

otherwise.

We can solve the system of equations Eq. 6 for $\overline{cwnd}_{\mathrm{d}}$ and $\overline{cwnd}_{\mathrm{u}}$, and use them to obtain the ratio r between the upload and download throughput:

$$r \approx \frac{\overline{cwnd}_{\mathrm{d}}}{RTT_{\mathrm{d}}} \frac{RTT_{\mathrm{u}}}{\overline{cwnd}_{\mathrm{u}}} \approx \frac{\overline{cwnd}_{\mathrm{d}}}{\overline{cwnd}_{\mathrm{u}}} = \frac{\gamma - 3/4\rho}{1 - 3/4\rho\gamma} \approx \frac{\gamma}{1 - 3/4\rho\gamma}, \tag{11}$$

where RTT_x are the RTTs for upload or download and $\gamma = \frac{B_{\mathrm{d}}+P_{\mathrm{d}}}{B_{\mathrm{u}}+P_{\mathrm{u}}}$. The approximation $RTT_u \approx RTT_d$ is valid as long as they are both dominated by the queueing delays or if the upload and download endpoints are at a similar "distance" (we do not consider here the interactions between TCP connections with different RTTs). We can see that the throughput ratio is at least equal to ratio $\frac{B_{\mathrm{d}}+P_{\mathrm{d}}}{B_{\mathrm{u}}+P_{\mathrm{u}}}$, so that it is important to use buffers with proper sizes to reduce upload to download interference. The reciprocal relation (γ as a function of r) may be used for buffer dimensioning: $\gamma = \frac{r}{1+\frac{3}{4}\rho r}$.

To summarize, the data pendulum effect determines how TCP connections occupy the buffers and we can use it to dimension the buffers properly. Nevertheless, we need to confront the insights gained from the analysis above with more simulation results as well as with measurements. Also, the analysis makes the assumption that buffers are structured as arrays of bytes, so we need to consider a more common case of buffers structured as arrays of packets.

4. SIMULATIONS

We still consider the setup from Figure 2: the downlink capacity of 2.4 Mb/s and 196 kb/s uplink (from R2 to R1). The link propagation delay is $T_{\mathrm{p}} = 20$ ms or $T_{\mathrm{p}} = 100$ ms. On the left side seen as the "Internet", the upload and download endpoints have delays of 5 ms and 1 ms "away" from the R1-R2 link, respectively, so that packets of the two connections do not follow a perfect cycle from one round trip to the next. Each simulation run represents 10 minutes of traffic.

To validate our estimate of r (Eq. 11), we compare the analytical value $\frac{\overline{cwnd}_{\mathrm{d}}}{\overline{cwnd}_{\mathrm{u}}}$ with the ratio obtained by simulation against uplink buffer sizes for various downlink buffer sizes in Figure 10. Although our model is relatively simple and does not capture all the complexity of TCP behavior, the simulation values follow our estimate fairly well especially for 100ms delay and for a not too small uplink buffer. Recall that in general, contending TCP connections exhibit high variations in throughput (cf. Figure 1).

We can extend the analysis in Section 3.1 to the case of multiple uploads, which is relevant in the case of P2P traffic. Simulations with 10 uploads exhibit the same behavior (not shown here as there is no significant difference).

4.1 Uplink buffer structured in packets

In most (if not all) operating systems, buffers can only hold a limited number of packets regardless of their size. Unfortunately, they are not as simple to model as buffers structured as an array of bytes.

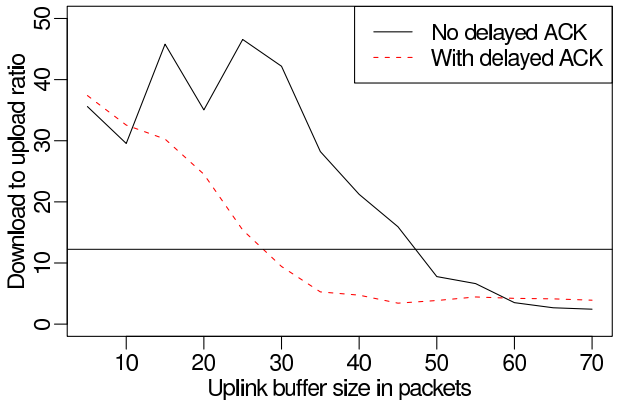


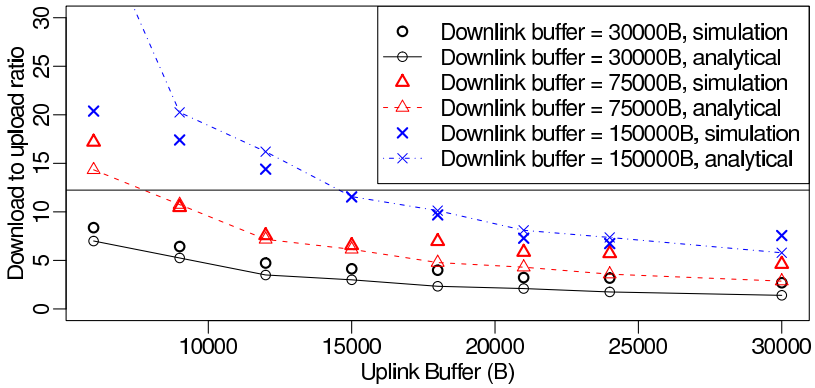
Figure 11: Asymmetric link, downlink buffer structured as an array of 50 packets. TCP buffer size 256KB. Each point represents 1000s of simulation, horizontal line is k. Performance strongly depends on delayed ACKs and interference between connections is more important than in the case of the buffers in bytes.

Figure 11 shows the simulated throughput ratio between upload and download for various uplink buffer sizes. Clearly, there is a range of buffer sizes that result in good performance (i.e. when the ratio k between the asymmetric link capacities is around 12), although it depends on the use of delayed ACKs for the download: every time a data segment enters the uplink buffer, it may compete with k or $k/2$ download ACKs. Two cases emerge: either the uplink buffer is large enough for the flow of ACKs so the upload will impact the download. Or the uplink buffer is too small so the upload suffers from frequent losses, its congestion window does not grow much, and there is little impact on the download.

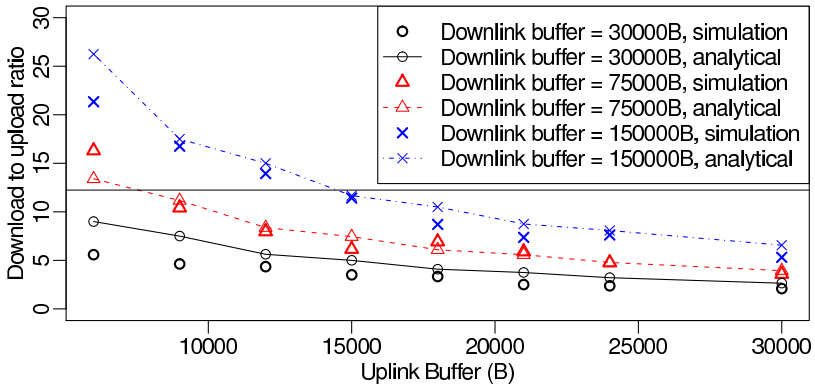
On the right side of the figure, interference between connections is similar to the case of buffers in bytes. On the left side, the upload suffers much more from the presence of many ACKs than in the case of buffers in bytes: ACKs quickly fill the buffer slots, which results in data segment losses detrimental to the upload. In this case, there are not enough slots for ACKs in the upload buffer, so that both ACKs and data segments suffer from losses, which severely impacts the upload. Consequently, it can obtain significant throughput only when there is enough room in the buffer for most of ACKs, that is for 25 slots with delayed ACKs or 50 otherwise.

Note that one might think that a buffer in bytes helps because it tends to prioritize the (smaller) ACKs. However, this would only hold for a buffer near saturation, whereas in regular conditions TCP causes rare buffer overflows only at the peaks of the sawtooth oscillations of the congestion window.

Unfortunately, using large buffers on both ends of ADSL lines is a common practice [9] that causes a notable drop of download throughput in the presence of uploads. In other words, most ADSL links operate at the right side of Figure 11—in the situation in which downloads are far from saturating the access link in presence of the uplink traffic. Moreover, large uplink buffers result in unacceptable latency, as it takes seconds to transmit 50 1500 bytes packets on a 196 kb/s link.



10.1: 20 ms propagation delay



10.2: 100 ms propagation delay

Figure 10: Comparison of the analytical value $\frac{\overline{cwnd_d}}{\overline{cwnd_u}}$ with the ratio obtained by simulation against uplink buffer sizes for various downlink buffer sizes. Good fit between modeling and simulation results.

5. MEASUREMENTS

We ran tests on Cisco routers (2821 and 2811 running IOS 12.4(24)T2) and FreeBSD 7.2 hosts: a series of ten experiments with the downlink buffer of size 50 packets and the varying size of the uplink buffer: from 5 to 50 packets. A 100 Mb/s Ethernet connects R1 and R2 with traffic shapers restricting the capacity to respectively 2.4Mb/s and 196kbps on R1 and R2. Hosts have the same TCP configuration and the default FreeBSD 7.2 kernel settings except for the net.inet.tcp.inflight.enable option that is disabled (it would lead to a "Vegas-like" TCP). We thus use the New Reno variant of TCP with SACKs and delayed ACKs.

Figure 12 presents the throughput ratio. Interestingly, there is little interaction between connections for short buffers, whereas for larger buffers the behavior is similar to what we obtain by simulation. The main difference between the simulations and the measurements is the use of traffic shapers based on token buckets. In the measurements, we observe that they allow us to obtain fairly good uplink performance even for relatively short buffers. For larger uplink buffers, the uplink is saturated and the traffic shaper is constantly busy, so that it behaves similarly to a bandwidth constrained link.

Finally, we measure the dynamic behavior of TCP in the previous setup and compare *cwnd* with the instantaneous bandwidth-delay product (we derive the delay from *RTT* estimated by TCP). Figure 13.1 shows the case in which

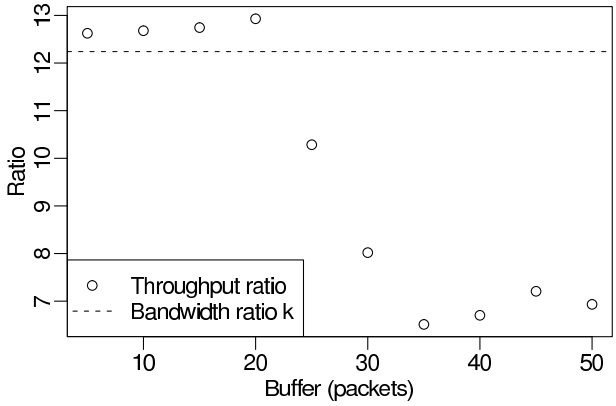
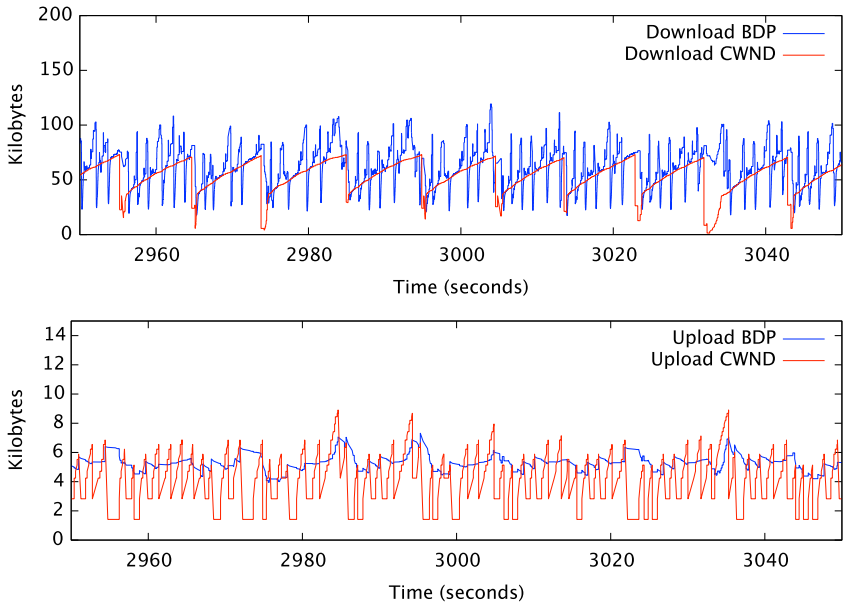
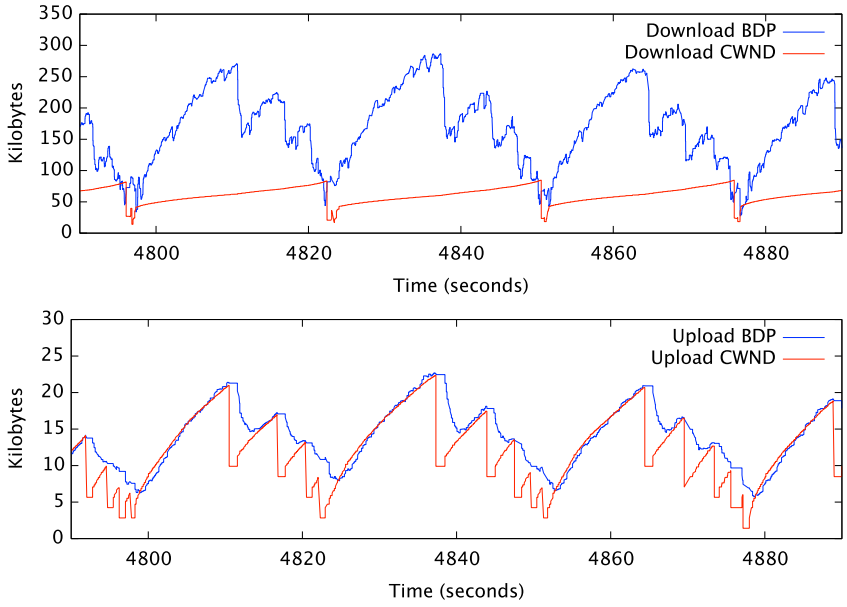


Figure 12: Measured throughput ratio vs. uplink buffer size.

cwnd follows on the average the bandwidth delay product, which results in good throughput ratio. On the contrary, in Figure 13.2, the download connection does not sufficiently fill the buffer and the resulting ratio is too low. Figure 13 clearly illustrates how, depending on buffer sizing, the TCP senders manage or fail to track the delay product that the connections are subject to.



13.1: Uplink buffer of 15 packets



13.2: Uplink buffer of 40 packets

Figure 13: Buffers structured as arrays of packets, measurement results. When the uplink buffer is too large (40 packets), the download connection fails to follow the fluctuations of the bandwidth-delay product.

6. CONCLUSION

In this paper, we have shown that the *data pendulum* effect is the primary cause of performance problems in the interaction of two-way TCP connections. Previous work considered ACK compression as the root of the problems, however, it has only a minor influence on the connection performance in general. Otherwise, degraded performance comes from the fact that the link can be kept busy only in one direction at a time.

Our analysis shows that small upload buffers (if possible, in bytes) greatly reduce harmful interference between uploads and downloads. In the case of buffers structured as an array of packets, the best solution is to use fair queueing mechanisms or to set the buffer to a reasonable size to avoid either upload starvation or drastic drop in download goodput. Also, simply reducing the TCP buffer size for uploads can result in a smaller uplink queue, although this solution is limited when several uploads coexist.

Acknowledgments

This work was partially supported by ANR (*Agence Nationale de la Recherche*) contract ELAN ANR-08-VERS-008.

7. REFERENCES

[1] Scott Shenker, Lixia Zhang, and David D. Clark. Some Observations on the Dynamics of a Congestion Control Algorithm. *SIGCOMM Comput. Commun. Rev.*, 20(5):30–39, 1990.

[2] Lixia Zhang, Scott Shenker, and David D. Clark. Observations on the Dynamics of a Congestion Control Algorithm: the Effects of Two-Way Traffic. *SIGCOMM Comput. Commun. Rev.*, 21(4):133–147, 1991.

[3] Rick Wilder, K. K. Ramakrishnan, and Allison Mankin. Dynamics of Congestion Control and Avoidance of Two-Way Traffic in an OSI Testbed. *SIGCOMM Comput. Commun. Rev.*, 21(2):43–58, 1991.

[4] L. Kalampoukas, A. Varma, and K.K. Ramakrishnan. Two-Way TCP Traffic over Rate Controlled Channels: Effects and Analysis. *IEEE/ACM Transactions on Networking*, 6(6):729–743, Dec 1998.

[5] Lampros Kalampoukas, Anujan Varma, and K. K. Ramakrishnan. Improving TCP Throughput over Two-Way Asymmetric Links: Analysis and Solutions. *SIGMETRICS Perform. Eval. Rev.*, 26(1):78–89, 1998.

[6] Hari Balakrishnan, Randy H. Katz, and Venkata N. Padmanbhan. The Effects of Asymmetry on TCP Performance. *Mob. Netw. Appl.*, 4(3):219–241, 1999.

[7] H. Balakrishnan, V. Padmanabhan, G. Fairhurst, and M. Sooriyabandara. RFC 3449 TCP Performance Implications of Network Path Asymmetry. `http://tools.ietf.org/html/rfc3449`, December 2002.

[8] Fatma Louati, Chadi Barakat, and Walid Dabbous. Handling Two-Way TCP Traffic in Asymmetric Networks. In *7th IEEE International Conference on High Speed Networks and Multimedia Communications HSNMC'04*, number 3079 in LNCS, pages 233–243. Springer-Verlag, 2004.

[9] Jesper Dangaard Brouer. Optimization of TCP/IP Traffic Across Shared ADSL. Master's thesis, University of Copenhagen, 2005.

[10] Denis Collange. Performance model of opposite TCP connections on asymmetric capacities. In *10th Symposium on Performance Evaluation of Computer and Telecommunications Systems (SPECTS 2007)*, San Diego (USA-CA), july 2007.

[11] Qualnet 5.0.2. Scalable Networks.