HAL
open science

A new look at bidirectional TCP connections over asymmetric links

Martin Heusse, Timothy X Brown, Thomas Schwengler, Andrzej Duda

To cite this version:

Martin Heusse, Timothy X Brown, Thomas Schwengler, Andrzej Duda. A new look at bidirectional TCP connections over asymmetric links. [Research Report] RR-LIG-002, LIG lab. 2010. hal-01073421

HAL Id: hal-01073421

https://inria.hal.science/hal-01073421

Submitted on 13 Aug 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Les rapports de recherche du LIG

A new look at bidirectional TCP connections over asymmetric links

Martin HEUSSE, Professor, U. Joseph Fourier, Grenoble, France
Timothy X BROWN, Associated Professor, U. of Colorado, Boulder, USA
Thomas SCHWENGLER, Instructor Adjunct, , U. of Colorado, Boulder, USA
Andrzej DUDA, Professor, Grenoble-INP, Grenoble, France

RR-LIG-002
mars 2010

http://rr.liglab.fr
ISSN 2105-0422

L I G

A new look at bidirectional TCP connections over asymmetric links

Martin Heusse, Timothy X Brown,
Thomas Schwengler, Andrzej Duda

Abstract

Many papers invoke the *ACK compression* phenomenon to explain the download performance drop in the presence of bidirectional traffic, whereas we argue that this symptom is merely a side effect of the core phenomenon that we analyze here. Interferences between upload and download is more and more frequent as it requires two commonplace ingredients: asymmetric links like most SOHO access links and long-lasting TCP uploads e.g. due to the presence of peer-to-peer traffic.

We find that interaction between uploads and downloads is simple to analyze in the rare case where the (up)link buffer size is given in bytes. Otherwise, in the more common case where buffer size is a given number of packets, there is greater adverse interactions. We provide guidance for sizing the link buffers to mitigate detrimental interference. We provide experimental results that illustrate the case with buffer sizes in bytes using an asymmetric WiMAX link.

1 Introduction

We study the interactions between upload and download traffic on asymmetric links. This phenomenon, which often causes drastic throughput reduction of the TCP downloads, is not well understood although this type of link is pervasive at the edge of the Internet. More precisely, in many cases, data transfers in each direction obtain similarly low throughputs, which on a typical ADSL line leads to filling only one tenth of the downlink capacity in the presence of uploads. This problem is rendered more acute by the evolution in network usage as users tend to upload more and more data, often continuously in the case of P2P software [10]. Moreover, this hit on download goodput strongly affects user perceived responsiveness.

Consider the example shown in Figure 1. The goodput over time of an asymmetric link with 2 Mbps download and 192kbps upload rate is shown. The download starts at time 10s and its throughput is shown in black. Between seconds 30 and 45 there is a single upload (dashed line). Between seconds 60 and 85 there is a pair of uploads (dashed lines). The buffer can hold 75,000B (which corresponds to 50 full size packets) in the downlink direction and uplink direction, which is a typical setting. The download can achieve nearly 2 Mbps when sent by itself. However, the presence of an upload causes the throughput to drop so that both upload and download achieve about 200kbps. Giving proper sizes to the buffers lessens the interaction significantly, as in figure 2.

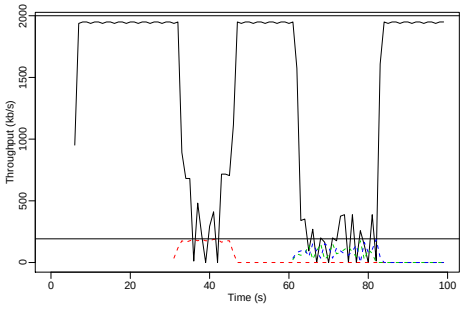


Figure 1: Interference between one download (solid line) and uploads (dashed lines: one then two) with buffers of size 75,000 B on both ends of the asymmetric link

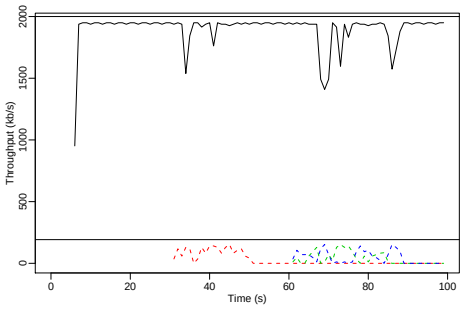


Figure 2: Interference between one download and one upload with buffers of size 75,000 B for the downlink and 6,000 B for the uplink

This example illustrates the key phenomena about simultaneous TCP downloads and uploads that we are investigating. First, uploads can have a dramatic negative effect on download performance. Second, how the buffer is managed (in packets or in bytes) is important. Third, the buffer size is the key factor in mitigating the negative effects. Fourth, bigger buffers are not always the solution.

Many papers invoke the *ACK compression* phenomenon to explain download performance drop in presence of bidirectional traffic [3, 4, 5, 8, 9, 12]. In a nutshell, ACK compression refers to the clustering of data packets and ACK packets of both connections, which allegedly compromises TCP connections self clocking by the acknowledgements. These papers focus more on proposing solutions to the problem than actually understanding it. For example, we could not find a paper that considers the influence of link buffer sizes on the magnitude of the problem.

In this paper, we study upload download interference and assert that:

1. *ACK compression* is not the cause of upload download interference, although it could explain why, in some cases, both the uplink and downlink may be not fully utilized [7];

2. Buffers in bytes are beneficial, especially on the uplink side, although not because of the intuitive reason that they naturally give more precedence

to ACKs (as ACKs may fit in a buffer which is otherwise full for data segments[1]). Rather, buffers in bytes are useful because they allow much smaller uplink buffers without hurting the upload when a download is present; this is in turn mandatory to reach a satisfactory utilization of the downlink.

We focus first on the case in which the asymmetric link buffer sizes are measured in bytes (Section 3). In this case, interferences between uploads on downloads are milder than usually thought. This case lends itself well to analysis, and the various parameters involved can be tuned to reduce adverse interferences predictably.

However, in many cases, buffers can keep only a limited number of packets regardless of their size and this generally leads to more marked interactions between uploads and downloads. We study this case by simulation in Section 4 and show that it is more challenging to find settings that would consistently give a satisfactory behavior.

To confirm our findings, we ran performance tests on one of the rare links with buffers limited in bytes, a WiMAX link. The results are given in Section 5.

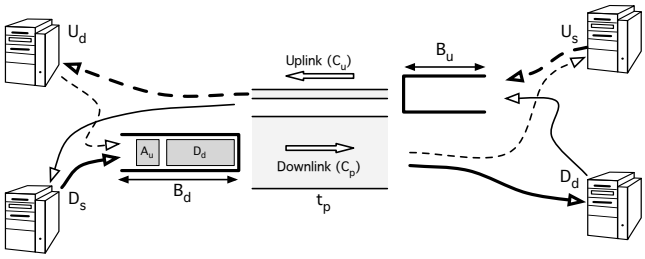


Figure 3: Two flows in opposite directions on the same link. The data packets (thick lines) in a buffer interact with the opposite direction's ACK packets (thin lines). Both flows experience the same delays over a round trip.

2 Interference between upload and download on an asymmetric link

We consider two TCP (half-)connections on an asymmetric link. The download ACK stream shares the upstream buffer with the upload data packets and *vice versa*. The TCP end points are connected to an asymmetric link (Fig. 3). The buffers at the heads of the asymmetric link are on different network layer entities than the TCP endpoints, so that the TCP connections continuously probe the link bandwidth and packets may be lost. (If the buffers were on the same entity, the behavior would not be interesting since the TCP connections would refrain from transmitting when the buffer is full.) The TCP connections may use the delayed ACK mechanism whereby an ACK is only sent for every other data packet that is received. We will show later how this factors into our analysis.

[1] As discussed in `http://www-nrg.ee.lbl.gov/floyd/REDaveraging.txt`, cited in RFC 2309.

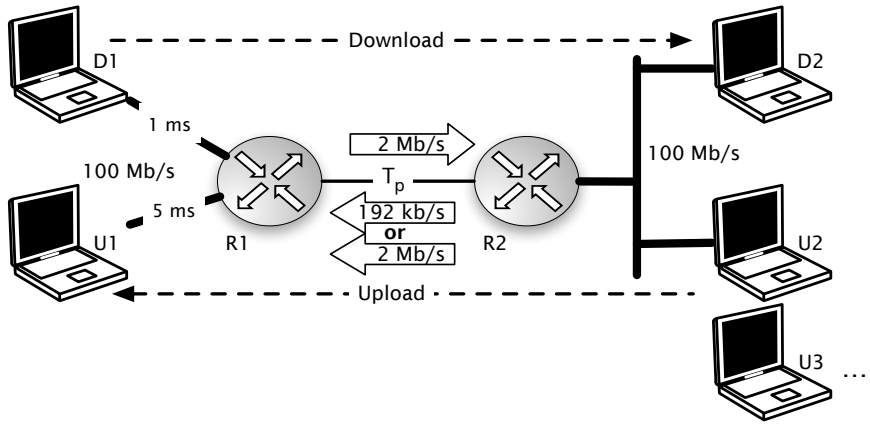


Figure 4: Simulation setup

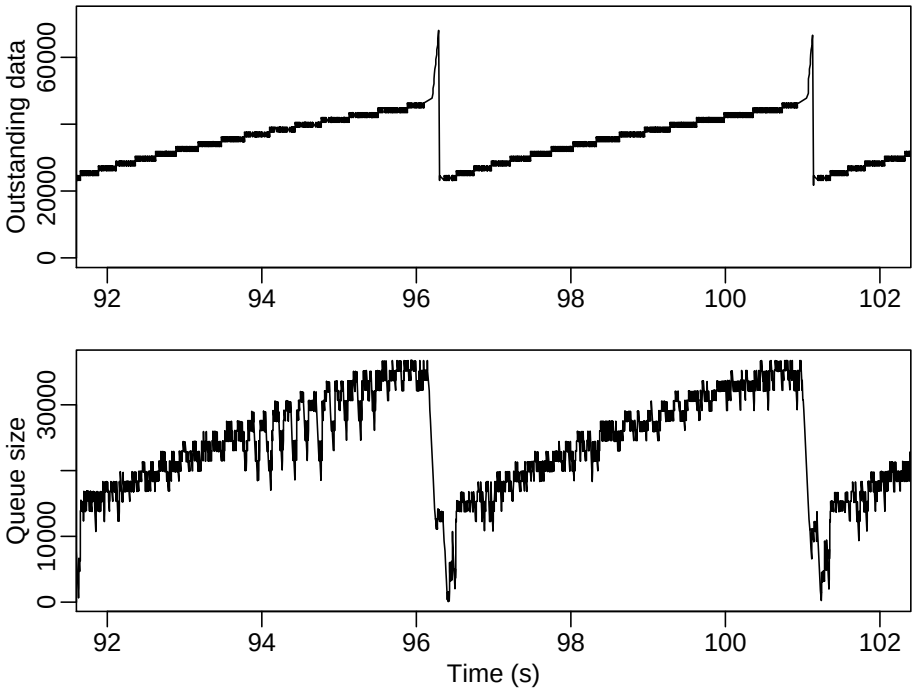
Considering a tagged data packet D_d from the download stream, if we leave aside the delayed ACK mechanism, reception of this segment will trigger the emission of an ACK packet A_d. This ACK will enter the upstream queue behind an upload segment D_u. When they leave the asymmetric link these packets will engender at least one new data packet D'_d and a new acknowledgement A'_u that will enter the downstream queue simultaneously. Thus, it is clear that both connections experience similar round trip times. Note that this holds if the end points of the connections are disjoint, as long as the RTT is dominated by the waiting time in the buffers at the head of the bottleneck (asymmetric) link and the propagation time T_{p}.

For a congestion window, $cwnd_x$, where x refers to uplink or downlink and RTT the round trip time, the connection throughputs are approximately $T_x = \frac{cwnd_x}{RTT}$. Thus, for instance if the 2 connections end up with similar average congestion windows *cwnd*, then they will obtain similar throughputs. We will see that this is what happens in most cases.

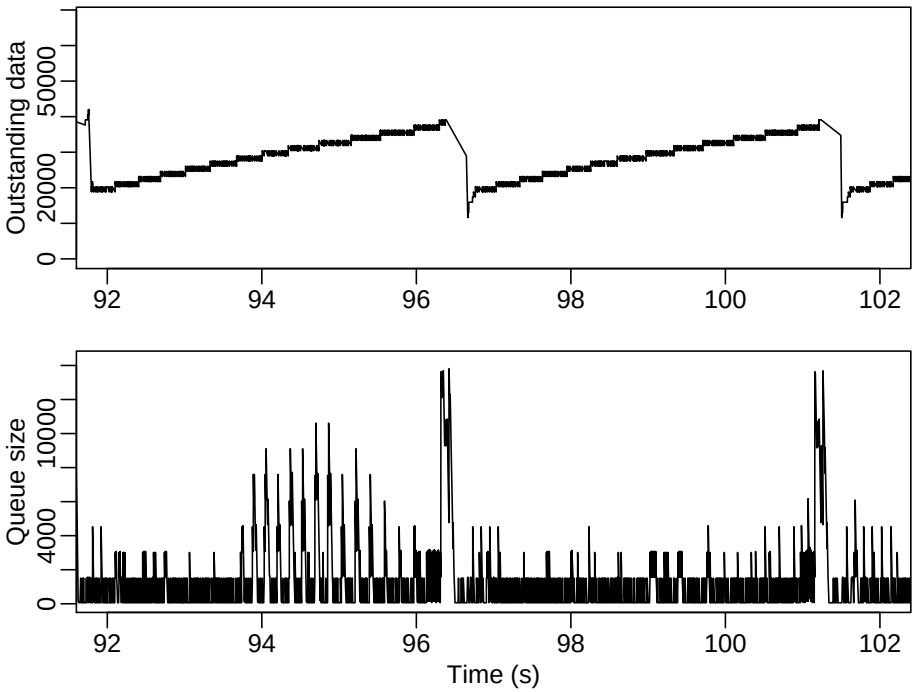
2.1 ACK compression is a side effect of upload/download interaction

Subsequently, because TCP congestion windows are inflated upon reception of an acknowledgement, data packets are produced back-to-back. When this process repeats, then ACK compression appears. Note that in this perspective, it does not matter if ACKs are compressed or not for uploads to impact downloads. Consequently, simply addressing this aspect by pacing TCP senders does not help. In fact, RFC 3449 explicitly rules out this type of solution.

On the other hand, prioritizing ACKs in the upstream buffers reduces the RTT of downloads, thus effectively allowing downloads to occupy the whole downlink bandwidth. Using any fair queueing mechanism at the uplink helps to solve this problem, as ACKs represent little volume compared to the upload data segments, so that they are effectively prioritized.



5.1: Buffer occupancy at R1



5.2: Buffer occupancy at R2

Figure 5: Buffer occupancies at both ends of a *symmetric link* with different buffer size (37500 and 15000 bytes)

2.2 Pendulum of the outstanding data

Two TCP connections exhibit a hieratic behavior when they share a link in opposite directions. The amount of data in the buffers at both ends of the link often shifts quickly from one side to the other. In essence this oscillation reflects the fact that the amount of data in flight from both connections is slow-varying, so if one queue builds up, the other dips. One example appears in figure 5. This is obtained using the setup in Figure 4; the link capacities are **symmetrical** (2 Mb/s) with different buffer sizes at both ends of the link: 37500 and 15000 bytes.

Two types of oscillations are visible: first quick oscillations between $t \approx 94$ s and $t \approx 96$ s. The best explanation for these is ACK compression. We note that it does not always emerge: there is nothing similar between 98 s and 100 s for instance. Our point is that ACK compression is a reality but it's not the most important cause of upload download interference. Let us look at the other wide oscillation starting at $t = 96$ s. At this time, the buffer in R1 overflows. This causes the download to stop sending packets for a while. Then all upload outstanding data goes into the buffer in R2 which immediately overflows. In the remainder of this section, we model this pendulum of outstanding data.

We consider the downlink and uplink capacities C_d and C_u, with $C_\mathrm{d} = k \times C_\mathrm{u}$ (and generally $k \geq 1$). Moreover, let d_d and d_u the number of data bytes standing in the downlink and uplink buffers. We neglect the amount of data in flight on the link and we consider that the ACK size is negligible. Then, as noted in section 2,

$$RTT = \frac{8\ d_\mathrm{d}}{C_\mathrm{d}} + \frac{8\ d_\mathrm{u}}{C_\mathrm{u}}. \tag{1}$$

Also, d_d and d_u are directly linked to the congestion windows of the upload and download. The data in flight for each TCP connections is the sum of the data segments waiting in the bottleneck buffer and the ACKs waiting in the reverse direction. For example, every time a download ACK enters the uplink buffer, it experiences a delay of $\frac{8\ d_\mathrm{u}}{C_\mathrm{u}}$. So the set of ACKs in this buffer stand for $\frac{d_\mathrm{u} C_\mathrm{d}}{C_\mathrm{u}}$ data bytes.

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

(and similarly for the upload). Plugging 2 into 1 yields simply

$$RTT = \frac{8\ cwnd_\mathrm{d}}{C_\mathrm{d}} \tag{3}$$

and conversely we have

$$RTT = \frac{8\ cwnd_\mathrm{u}}{C_\mathrm{u}} \tag{4}$$

which implies either that:

1. $d_\mathrm{d} = 0$ and then $RTT = \frac{8\ cwnd_\mathrm{u}}{C_\mathrm{u}}$; or,
2. $d_\mathrm{u} = 0$ and then $RTT = \frac{8\ cwnd_\mathrm{d}}{C_\mathrm{d}}$; or,
3. $d_\mathrm{u} \neq 0$ **and** $d_\mathrm{d} \neq 0$ only if $\frac{cwnd_\mathrm{u}}{C_\mathrm{u}} = \frac{cwnd_\mathrm{d}}{C_\mathrm{d}}$.

In figure 5, the system is in case 2 most of the time, and goes into case 1 only when the buffer in R1 overflows and subsequently empties because the download stops for a while. Then we go to case 1 and the buffer in R2 immediately overflows.

Let us consider what decides of the case in which the system operates. We consider for example that $d_{\mathrm{d}} = 0$ (case 1). Then, the download does not occupy the downstream link continuously (or just barely), so that its packets are not well interspersed in the uplink buffer. So $cwnd_{\mathrm{d}} \leqslant d_{\mathrm{u}} \frac{C_{\mathrm{d}}}{C_{\mathrm{u}}}$.

Multiplying both sides of this inequality by RTT and using Eq. 4, we get

$$\frac{cwnd_{\mathrm{d}}}{cwnd_{\mathrm{u}}} \leqslant \frac{8\ d_{\mathrm{u}} C_{\mathrm{d}}}{RTT\ C_{\mathrm{u}}^2}$$

and, as $RTT = \frac{8\ d_{\mathrm{u}}}{C_{\mathrm{u}}}$, we have:

$$d_{\mathrm{d}} = 0 \Rightarrow \frac{cwnd_{\mathrm{d}}}{C_{\mathrm{d}}} \leqslant \frac{cwnd_{\mathrm{u}}}{C_{\mathrm{u}}} \tag{5}$$

This results is not unexpected: queueing happens on the side of the connection that is more intensely filling its buffer. Similarly,

$$d_{\mathrm{u}} = 0 \Rightarrow \frac{cwnd_{\mathrm{d}}}{C_{\mathrm{d}}} \geqslant \frac{cwnd_{\mathrm{u}}}{C_{\mathrm{u}}}.$$

One important consequence of this remark is that, unless the *cwnd*s are kept proportional at all time, one of the links is not fully utilized. Unfortunately, this is not achievable with TCP on an asymmetric link, as the growth of the congestion windows is the same in both directions since it depends only on the RTT. So it cannot be maintained equal to the ratio of the capacities. Then it is impossible to maintain both (half-) links permanently active. Note that we reach this conclusion without invoking ACK compression.

2.3 Buffer size for unidirectional traffic

The minimal size for the link buffers is a good starting point so we recall the rule of thumb to set them. This is not directly linked to upload/download interference, but we remark that, in general, buffers should be of different sizes on both ends of an asymmetric link.

With only unidirectional traffic, the link buffers should be large enough to keep the link busy when there is only a single active TCP connection on the link. The usual rule of thumb for buffer size stipulates that it should be

$$B_{\mathrm{x}} = \frac{2 \times t_{\mathrm{p}} \times C_{\mathrm{x}}}{8\ MTU} \tag{6}$$

packets where C_{x} is the bottleneck link capacity [11] and t_{p}, the end-to-end propagation delay. Note that smaller buffer sizes give satisfactory results when a large number of connections are active concurrently [2], but this is obviously not a generally valid hypothesis for access links.

3 Asymmetric link with Buffers measured in bytes

We seek to obtain a simple expression for the throughput ratio r between upload and download. To this end, we estimate the congestion windows of both connections.

For an upload, first, two factors may limit $cwnd_{\mathrm{u}}$ growth:

1. TCP buffer size W at sender or receiver. If W is smaller than the buffer size plus the amount of data in flight further down, the sender is paced by the rate of acknowledgment reception and there is always a full TCP buffer worth of data in flight;

2. Uplink buffer size: the amount of sender data in flight follows a sawtooth curve that culminates when the buffer overflows. $cwnd_{\mathrm{u}}$ oscillates between $\frac{B_{\mathrm{u}}^{*}+P_{\mathrm{u}}}{2}$ and $B_{\mathrm{u}}^{*}+P_{\mathrm{u}}$, where B_{u}^{*} is the uplink buffer space left by the download ACKs and P_{u} is the amount of data in flight on the link, $P_{\mathrm{u}} = 2\, C_{\mathrm{u}}\, t_{\mathrm{p}}$. Then if ρ is the ratio of ACK size to the segment size, ($\rho \approx \frac{1}{30}$),

$$B_{\mathrm{u}}^{*} = B_{\mathrm{u}} - \rho\ \overline{cwnd}_{\mathrm{d}}/2 \tag{7}$$

   if ACKs are delayed, and $B_{\mathrm{u}}^{*} = B_{\mathrm{u}} - \rho\ \overline{cwnd}_{\mathrm{d}}$ otherwise. Note that we do not take into account that, in principle, some ACKs may be standing in the downlink queue at the moment the uplink buffer overflows. This is a direct consequence of the remark in section 2.2.

Similarly, the throughput of downloads is limited by the downlink buffer size. In this case, we simply neglect the presence of ACKs in the queue, as ACKs are at the same time a lot smaller than data packets and they are less numerous as we still expect the download to have a higher throughput than the upload.

Moreover, we suppose that the downlink buffer (in R1) is larger than the uplink buffer, which is in fact mandatory for reasonable performance. Then the sawtooth oscillations of the upload $cwnd_{\mathrm{u}}$ will be much faster than on the download side. Thus, it is fair to consider that as soon as $cwnd_{\mathrm{d}}$ grows bigger than what the system can store, that is the downlink buffer size added to the amount of data in flight P_d, then the buffer at R1 overflows.

So we have, if $B_d^{*} = B_d$:

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

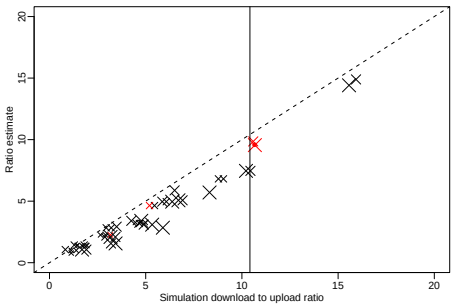
So the ratio r between upload and download throughput is then simply:

$$r \approx \frac{\overline{cwnd}_{\mathrm{d}}}{RTT_{\mathrm{d}}} \frac{RTT_{\mathrm{u}}}{\overline{cwnd}_{\mathrm{u}}} \approx \frac{\overline{cwnd}_{\mathrm{d}}}{\overline{cwnd}_{\mathrm{u}}} \tag{8}$$

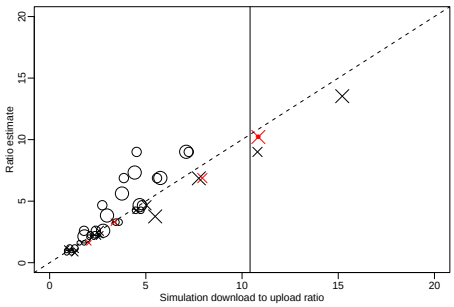
Where RTT_x are the RTTs for upload or download. The approximation that $RTT_u \approx RTT_d$ will be valid if they are both dominated by the queueing delays, or if the upload and download endpoints are at a similar "distance" (We do not consider interactions between TCP flows of various RTT). In a nutshell, we show that reasonable buffer sizes, in particular on the uplink side, greatly reduces adverse interference between uploads and downloads. Even more strikingly, the throughput ratio r does not depend on k, it is only marginally linked to C_{u} and C_{d} when computing P_{u} and P_d!

3.1 Simulations

We use the Qualnet network simulator [1] with the network topology in Figure 4. The downlink capacity of the bottleneck asymmetric link is 2 Mb/s downlink and 192 kb/s uplink (from R2 to R1). The asymmetric (bottleneck) link propagation delay is $t_{\mathrm{p}} = 20\,\mathrm{ms}$ or $t_{\mathrm{p}} = 100\,\mathrm{ms}$. On the left side, which can be seen as the "Internet", the upload and download endpoints are placed respectively 5 ms and 1 ms "away" from the R1-R2 link, 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. The TCP variant is New Reno with selective acknowledgments.

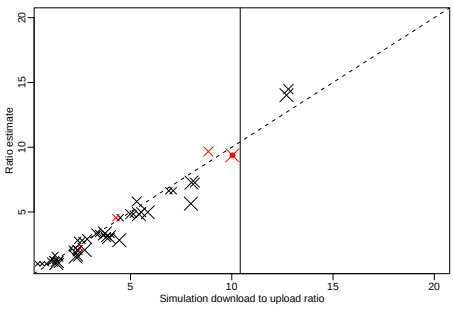


6.1: 20 ms propagation delay

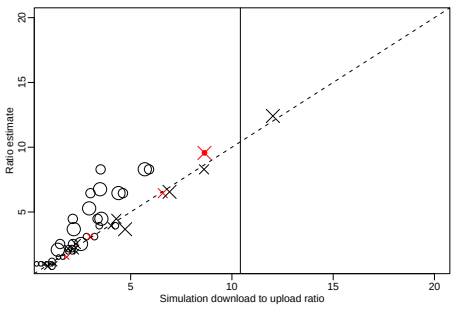


6.2: 100 ms propagation delay

Figure 6: Ratio estimate $\frac{\overline{cwnd}_{\mathrm{d}}}{\overline{cwnd}_{\mathrm{u}}}$ vs. the download to upload throughput ratio obtained by simulation. The vertical line is the downlink to uplink capacity ratio k. Overlying points are moved to form small clusters; circles are points for which the downlink buffer is smaller than the bandwidth delay product. Point sizes are proportional to the TCP buffer size. Red points correspond to link buffer sizes that follow the rule of thumb for a RTT of 300 ms.



7.1: 20 ms propagation delay



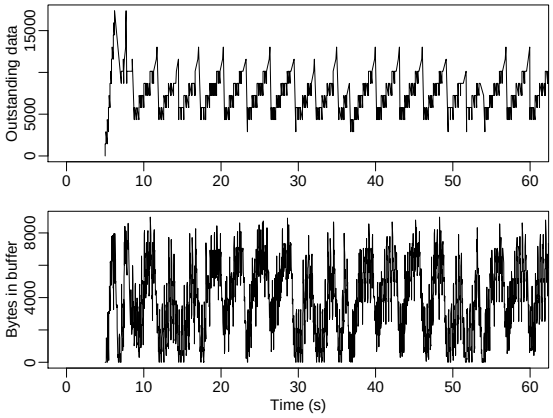
7.2: 100 ms propagation delay

Figure 7: Ratio estimate $\frac{\overline{cwnd}_{\mathrm{d}}}{\overline{cwnd}_{\mathrm{u}}}$ vs. the download to upload throughput ratio obtained by simulation, 10 concurrent upload connections.

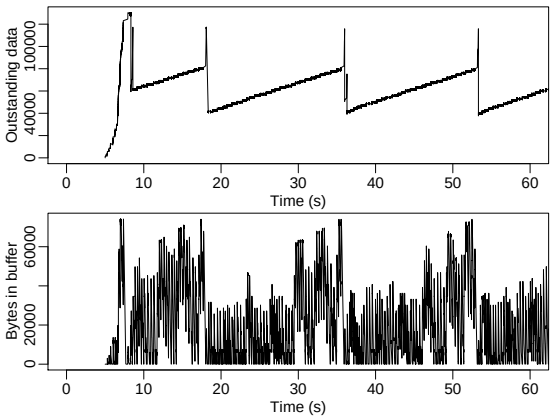
In Fig. 6, to validate our estimate of r, we plot $\frac{\overline{cwnd}_{\mathrm{d}}}{\overline{cwnd}_{\mathrm{u}}}$ against the obtained download to upload throughput ratios for TCP buffer size in (16384, 32768, 65536, 131072); uplink buffer size in (6000, 9000, 15000, 30000) and downlink

buffer size in (22500, 37500, 75000), all in bytes. The dots clearly follow our estimate, so that it is effectively the link buffer sizes that govern the upload to download throughput ratio, provided that TCP buffers are large enough (larger points on the figure) and the downlink buffer is large enough (the circles are offset to the left).

The desired ratio of approximately 10 between upload and download is obtained for TCP buffers of at least 64 kB and link buffers of 9000 B (uplink side) and 75000 B (downlink side). Note that this corresponds to the rule-of-thumb buffer size for approximately 300ms RTT, which is large enough in most cases [6].



8.1: Upload



8.2: Download

Figure 8: Outstanding data (*cwnd*) of upload, download and buffer occupancy at R1 and R2.

To illustrate the system behavior, we take a closer look at the case when both the uplink and downlink are maintained busy. We focus on the point marked by a dot in Figure 6. Figure 8 plots the congestion windows and buffer occupancies at the start of the simulation. Faster oscillations for the download are evident, as well as the dip in the downlink buffer occupancy after it overflows. To validate that when the uplink buffer overflows, the downlink buffer is mostly empty, we

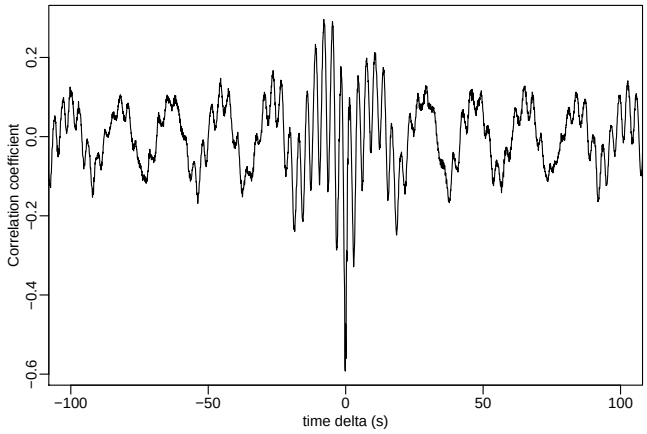


Figure 9: Convolution of the buffer occupancy time series.

plot the convolution of the buffer occupancy time series in Fig. 9. The negative correlation is patent.

We also would like to point out that the analysis in Section 3 can be extended to the case of multiple uploads, which is relevant in the case of P2P traffic for instance. Fig. 7 corresponds to this case, with 10 upload connections instead of one. The results are very similar to what we obtain with a single upload.

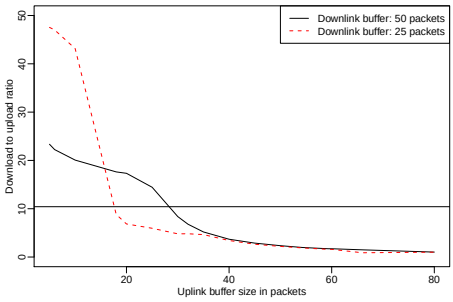
4 Uplink buffer measured in packets

In most operating systems (FreeBSD, Linux...) and for e.g. Cisco routers as far as the authors can tell, buffers can only hold a limited number of packets regardless of their size. So having buffers measured in packets is the usual case. Unfortunately this is not as simple to model as for buffer size in bytes.

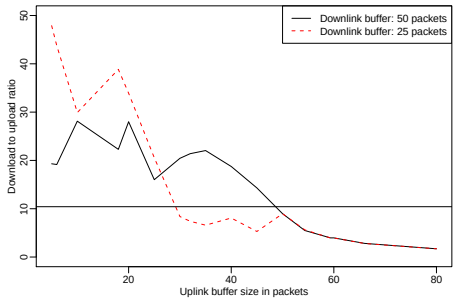
In particular, let's look at the minimum size for the uplink buffer. When there is one upload and one download, for each data packet sent uplink, n ACKs corresponding to k download data segments may concomitantly enter the uplink buffer. Consequently, when the upload TCP sender inflates its congestion window, its data packet compete with numerous ACKs. Thus using a buffer of e.g. 6 packets (9000 B) as in the previous section leads to a high loss rate that would cause the upload to stall.

In Fig. 10, we display the throughput ratio between upload and download for various uplink buffer size. Clearly there is a range of buffer sizes that would give satisfactory results, but it depends on the number of ACKs sent per TCP segment. With delayed ACK, in Fig. 10.1, one acknowledgment is sent upon receipt of every other segment; whereas there is one ACK per segment otherwise (Fig. 10.2). Without delayed ACK, the upload to download ratio shows strong variations for adjacent uplink buffer sizes. This variability is not due to the simulation duration, but rather to the fact that in this region uploads are starved, and their behavior is hard to predict or capture in this kind of adverse conditions.

Empirically, the uplink buffer should be at least half the size of the downlink buffer, or uploads will starve. This has a relatively straightforward explana-



10.1: With delayed ACK



10.2: No delayed ACK

Figure 10: Buffer measured in packets, TCP buffer size 64kB. Each point is 2000s of simulation; the horizontal line is k.

tion: when the upload transmission rate approaches the capacity, then download ACKs will queue in the uplink buffer. They will cause packet drops if the uplink buffer is not large enough to hold them all. For a smaller uplink buffer size, the download will not be able to occupy the entire available downlink bandwidth, as the ratio of the $\overline{cwnd}_x$ will be close to 1.

Using large buffers on both ends of ADSL lines is a common practice [5] and this causes a notable drop of the download throughput in the presence of uploads. In other words, most ADSL link operate to the right of Fig. 10; to the point where concurrent uploads and downloads obtain similar throughputs.

5 Experimentation with a WiMAX link

To confirm experimentally the observation of Section 3, we used one of the rare network links that use buffers limited in bytes: an 802.16d fixed WiMAX system (FDD base station, half duplex subscriber units). We checked that the buffer is measured in bytes by verifying that queueing delay in a full buffer does not change when loading the link with short or long packets.

On this system, we set QoS profiles to limit the maximum information rate (MIR) to 2 Mb/s downlink and 192 kb/s uplink. With these settings, a single TCP download goodput is 1.9 Mb/s, and 0.16 Mb/s for a single upload (goodput ratio of 12). If one download and one upload are present concurrently, then the download reduces to 1.04 Mb/s, while the upload gets 0.15 Mb/s. We note the download rate drops by about a half when an upload is present. However, the downlink is still able to maintain a throughput ratio of 7. Our simulation results suggest that with buffers measured in packets, the goodput ratio would drop to close to 1.

This is on of the rare cases where a much higher ratio is maintained by the use of buffers limited in bytes, although this fact is most probably not the cause of this unusual design choice.

6 Conclusion

In this paper, we showed that TCP performance on asymmetric links depends strongly on the buffer sizes on both sides of the link, as well as on how the buffer size is counted—in number of bytes or packets. Also, we consider that interferences between upload and download are not due to ACK compression.

We conclude that using a buffer limit calculated in bytes allows to greatly reduce adverse interference between uploads and downloads. This is not usual practice, though. On ADSL links, using buffers of adequate sizes counted in ATM cells would most probably be quite straightforward and sufficient. We also recommend to use buffers of relatively small size, especially on the uplink side, which is again not usual.

In the case of buffers that have a capacity set to a given number of packets, the best solution is to use fair queueing mechanisms, or set the buffer to a reasonable size to avoid either uploads starvation or drastic download goodput lessening. Also, simply restricting TCP buffer size for the uploads can limit uplink buffer occupancy, although this is a limited solution when several uploads coexist.

References

[1] Qualnet 4.5. Scalable Networks.

[2] G. Appenzeller, I. Keslassy, and N. McKeown. Sizing router buffers. In *SIGCOMM '04: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 281–292, New York, NY, USA, 2004. ACM.

[3] H. Balakrishnan, R. H. Katz, and V. N. Padmanbhan. The effects of asymmetry on TCP performance. *Mob. Netw. Appl.*, 4(3):219–241, 1999.

[4] H. Balakrishnan, V. Padmanabhan, G. Fairhurst, and M. Sooriyabandara. RFC 3449 TCP performance implications of network path asymmetry. `http://www.rfc-editor.org/rfc/rfc3449.txt`, Dec. 2002.

[5] J. D. Brouer. Optimization of TCP/IP traffic across shared ADSL. Master's thesis, University of Copenhagen, 2005.

[6] B. Huffaker, M. Fomenkov, D. J. Plummer, D. Moore, and kc claffy. Distance metrics in the internet. In *in proc. of the IEEE International Telecommunications Symposium*, 2002.

[7] L. Kalampoukas, A. Varma, and K. Ramakrishnan. Two-way TCP traffic over rate controlled channels: effects and analysis. *Networking, IEEE/ACM Transactions on*, 6(6):729–743, Dec 1998.

[8] L. Kalampoukas, A. 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.

[9] F. Louati, C. Barakat, and W. 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.

[10] H. Schulze and K. Mochalski. Internet study 2007. `http://www.ipoque.com/userfiles/file/internet_study_2007.pdf`, 2007.

[11] C. Villamizar and C. Song. High performance TCP in ANSNET. *SIGCOMM Comput. Commun. Rev.*, 24(5):45–60, 1994.

[12] L. Zhang, S. Shenker, and D. 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.