Title: TCP/IP Performance Optimization over ADSL

Authors: Shivkumar Kalyanaraman, Dibyendu Shekhar, Kalyan Kidambi
Contact author: Shivkumar Kalyanaraman
Affiliation: Department of ECSE, Rensselaer Polytechnic Institute
Address: 110, 8th Street, Room JEC 6003, Tory NY 12180-3590, U.S.A.
Telephone: +1 (518) 276 8979
Fax: +1 (518) 276 2433

Abstract:
Asymmetric Digital Subscriber Line (ADSL) is an Internet access technology over copper wires which promises significant bandwidth compared to current modem or ISDN technology. However, the TCP/IP performance over ADSL is significantly diminished due to the effects of asymmetry. We introduce a new operational model called the "AMP model" which, with an understanding of TCP dynamics and buffer management techniques explains the performance effects seen. We apply this model to guide design improvements in buffer management (ack-regulation) and scheduling schemes to achieve performance improvements of an order of magnitude or more. The improvements observed are better than those proposed in earlier literature.

General conference topics: Cable-Based Delivery & Access Systems, Quality of Service.
Symposium: Global Internet 2000 (GI2000)

1. Introduction

Asymmetric Digital Subscriber Line (ADSL) is a technology promising significant bandwidth increases over existing copper local loops for Internet (TCP/IP) traffic. ADSL provides an asymmetric channel for data transmission: the forward channel ranges in speeds from 1.5 Mbps to 8 Mbps whereas the reverse channel ranges in speed from 64 kbps to 768 kbps. This bandwidth asymmetry has a profound impact on TCP performance because the TCP throughput is regulated (and limited) by the flow of acknowledgements (called "ack clock") [tcp-cong].

The impact of asymmetry on TCP/IP performance has been characterized by Lakshman et al [lakshman] using an index called ***normalized asymmetry ratio (k)*** which is the *ratio of raw bandwidths on both directions to the ratio of packet sizes in both directions*. When k is greater than one, the TCP throughput on the forward channel is restricted to a maximum of (**forward channel bandwidth) / k**. Other factors like bi-directional traffic (e.g.: downloading a web page while sending an email) or protocol overhead (e.g.: PPPoE, PPTP, L2TP, ATM etc) will increase k and further aggravate the asymmetry problem.

Balakrishnan et al [hari], Keshav [keshav] and the PILC group [pep] independently show that performance can be substantially increased by making two key changes: simply suppressing acknowledgements on the reverse channel (***ack-filtering***), and regenerating them after the reverse link has been traversed (***ack-reconstruction***). Balakrishnan et al also apply the same techniques to address types of asymmetry other than bandwidth asymmetry including asymmetry in delay, loss rates etc. We classify these techniques, as "***ack-regulation techniques***" since they are very different from mechanisms like Random Early Drop (RED) because the latter are designed to drop packets, not acks, and are used to signal congestion to TCP, not alleviate asymmetry problems. Our work focusses on generalizing these studies into a model called the ***AMP model***, and making specific design improvements (Smart Ack Dropper: SAD) for performance optimization based upon this model.

2. The AMP model

We extend the notion of normalized bandwidth ratio (k) [lakshman] to form a simple model called the AMP model. The model is described as follows:

- Assume that, over an observation period, the ***fraction of the reverse link allocated to ack traffic is saturated at A acks/second***, and
- ***Each ack generates M packets on the average (M = multiplicative factor),*** due to the effects of TCP dynamics and ack-regulation schemes working together, and
- The ***average size of packets is P bits/packet,***
- The ***forward link capacity allocated to packets corresponding to these acks is F bits/s,*** then

the ***forward link throughput over the observation period is limited to Min (F, A*M*P) bits/second***.

Observe that Lakshman's normalized bandwidth ratio "k" characterizes an absolute upper bound on the forward throughput based upon link bandwidths and packet vs. ack sizes. The AMP expression, on the other hand, is an *operational bound* achievable with the asymmetric channel augmented with ack-regulation schemes. In particular, if the ack-regulation components can achieve a maximum average multiplicative factor of M, then the AMP expression is tighter bound on the achievable forward throughput. Moreover, it also accounts for scheduling allocations to acks and corresponding packets and vice versa (in the definition of F and A), and can hence be useful in understanding effects of bi-directional traffic.

For example, if the reverse link speed is 64kbps, the forward link speed is 8Mbps, packet size is 1000 bytes, ack size = 40 bytes, TCP is in its slow start phase (generating two packets per ack) and no ack-regulation schemes are used. Then ack rate (A) would now be 200 acks/second , the multiplicative factor M=2, and P = 8000 bits, the maximum throughput limit on the forward link = Min (8 Mbps, 3.2 Mbps) = 3.2 Mbps. Note that *the number of TCP flows sharing the ADSL link is immaterial in this model* (it is captured in M). In particular, in this case if the TCP sources are all in congestion avoidance (and not in slow start), then M is closer to 1, and limit is even lower (about 1.6 Mbps)!

The AMP model is also useful in analyzing the ***bi-directional*** traffic case, especially when link-sharing schemes like CBQ [cbq] are deployed. Specifically, assume that in each direction, there are two classes (queues) served by CBQ: one for packets and one for acks. Further assume that packets on the forward link get a fraction f ($1 > f > 0$) of the forward link capacity (C_f), and on the reverse link get a fraction g ($1 > g > 0$) of the reverse link capacity (C_r). Acks, therefore get fractions 1-f and 1-g of capacity forward and reverse links, respectively. Assume that P_{ack} is the size of acks in bits/ack and *that the reverse channel (both packets and acks) is saturated at their respective scheduling shares.* The bounds on the maximum rates of packets and acks in both directions are given in the following table:

	Max Packet Rate (pkts/s)	**Max Ack Rate (acks/s)**
Reverse Channel	$G^* C_r /P$ (saturated)	$(1-g)^* C_r$ /Pack (saturated)
Forward Channel	**Min[(1-g) C_r M/Pack, f*C_f/P]**	Min[f^*C_f/P_{ack} , $g^* C_r /P$]

Observe that f (the link fraction allocated to packets on the forward link) should be chosen to be large, and can be as large as 99% for pure TCP/IP bi-directional traffic. The choice of g (the link fraction allocated to packets on the reverse link) is a tradeoff between performance and policy considerations. We will use this model to design ack regulation and scheduling policies.

3. The Smart-Ack Dropping (SAD) and Ack Regeneration Policy

The Smart-ack dropping technique (SAD) is a simple extension of concepts developed by Balakrishnan et al [hari] and Keshav [keshav], i.e. to suppress as many acks in the reverse channel as possible because acks are cumulative. Balakrishnan et al's "*ack-filtering*" technique involves checking the entire queue upon the arrival of a new ack to remove earlier acks for the same connection. The goal is partially to free some space in the queue for other data packets and acks, and partially to compress the ack information. Keshav [keshav] independently proposes an "*ack-collapsing*" technique where all acks are queued, but at the transmission opportunity, the latest ack is sent and all others are dropped.

SAD is perhaps closest to Keshav's scheme in concept, but uses minimal per-flow state to avoid enqueuing acks, which are going to be dropped anyway. This way, we require only a buffer of size N (acks) where N is the number of active flows (assuming separate packet buffers). In particular, SAD works as follows:

We assume a FIFO queue for acks (and optionally packets). The per-flow information (stored as a hash table) includes a *single bit, which indicates if an ack of that flow exists in the queue*, and the *latest ack number seen from that flow*. When an ack arrives and the per-flow bit is zero, then we enqueue the ack, set the per-flow bit and copy the ack number into the table. If the per-flow bit is already set upon arrival of a non-duplicate ack, then we update the per-flow ack number information to this value, and drop the ack. When an ack is dequeued for transmission, the latest value of ack number from the table is copied to the header (header checksum adjusted), and the bit in the table is cleared. Ack processing is O(1), since we search the hash table and not the FIFO queue. This scheme can be extended to account for duplicate acks and SACKs [techreport].

A detailed analysis of the scheme applying the AMP model is also described in [techreport]. Specifically, if TCP is in congestion avoidance, the throughput is almost constant at W/RTT. The impact of SAD is to use the reverse channel capacity of A acks/s to support a maximum forward rate of Min[m*A, W/RTT] packets/s. When TCP is in slow start, each ack reaching the
TCP source results in (m + 1) segments being sent where m is the number of acks suppressed by SAD. In fact, if the TCP sources remains in slow start phase, the AMP model predicts that SAD can compensate for arbitrary degrees of asymmetry (i.e. any finite k). However, since most TCPs reach congestion avoidance within a few RTTs, the multiplicative factor saturates at the value of (near-constant) m seen during the congestion avoidance.

SAD also reduces the probability of negative interactions with RTT estimation (which occurs in Balakrishnan et al's scheme) because acks are sent out in a timely manner, and queueing delay is bounded. The suppression of acks leads to burstiness in the forward direction. This burstiness can be alleviated by using an ***acknowledgement regeneration/reconstruction (AR)*** technique at the end of the reverse link as

suggested by Keshav [keshav] and Balakrishnan [hari]. The regeneration scheme would regenerate (and optionally smooth out) acks suppressed by SAD. In fact, the regenerator could regenerate more than one ack per MSS (which we quantify as the "***regeneration factor***", ***R***), all the way upto one ack for every byte acked. The regeneration factor, R directly affects the rate of TCP window increase especially during slow start.

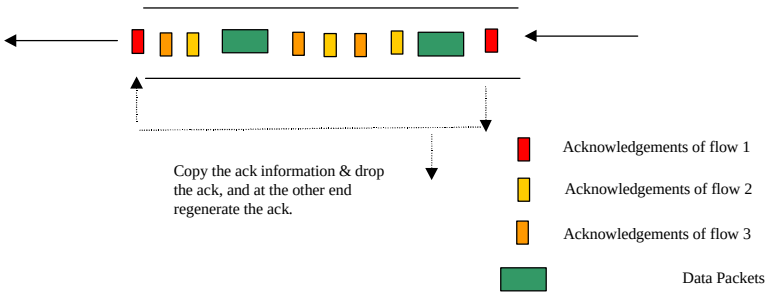

Figure 2:Illustration of SmartAckDropper(SAD)

4. Performance Evaluation

We have studied ack-regulation techniques have been studied using the "ns" simulator. The ADSL model and terminology used in this paper is described in Figure 1. The ADSL channel (which is our focus) is terminated on either end by devices, which are called the ATU-R (on the customer premises), and the ATM-C (at the central office). A larger suite of results may be found in [techreport]. The analytical explanation of the results and the limitations of the scheme are also discussed in that report.

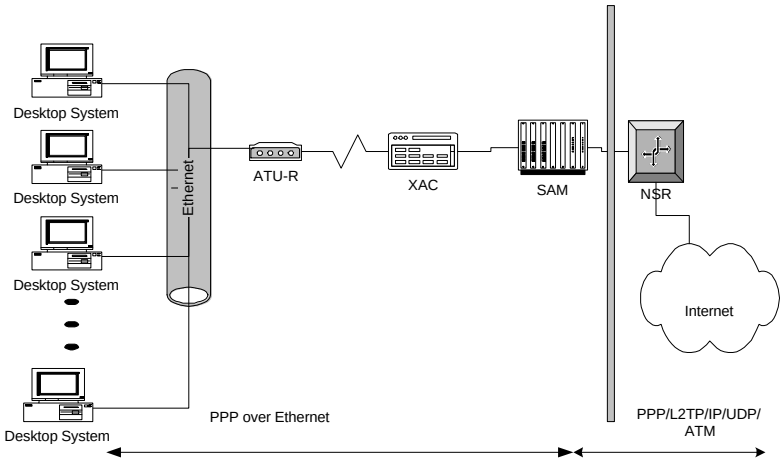

Figure 1: System Topology

4.1 Single TCP undirectional flow

With single TCP flow and without any ack regulation schemes, the simulation yielded the following results, which can be used as a baseline to evaluate the effectiveness of SAD (Table 1). As shown, the asymmetry problem grows increasingly worse as the upstream channel becomes more constrained to a worst case of 19% utilization.

Downstream/ Upstream Speed (Mbps/kbps)	**Throughput (Mbps)**	**Link Utilization**	**K**

8/64	1.52	19%	5[1]
8/128	3.01	37%	2.5
8/256	5.98	74.75%	1.25
8/640	7.46	93.25%	0.5
8/784	7.46	93.25%	0.41

Table 1: Uni-directional Single Flow Simulation

However, these problems can be overcome almost entirely by using SAD (Table 2). For instance, in the worst-case scenario (8 Mbps/64 kbps), ***throughput rose four-fold to 6.2 Mbps***.

Downstream link/ Upstream link (Mbps/kbps)	**Throughput (Mbps)**	**Link Utilization**	**K**
8/64	6.2	77.5%	5
8/128	6.79	84.88%	2.5
8/256	7.28	91%	1.25
8/640	7.45	93.13%	0.5
8/784	7.46	93.25%	0.41

Table 2: Uni-directional Single Flow Simulation with SmartAckDropper

Additional throughput can be gained using AR techniques (Table 3**).** A regeneration factor (R) of R generates R acks for every ack suppressed by SAD. With a regeneration factor of 4, a maximum throughput of 7.4 Mbps can be achieved even in the worst-case scenario.

Downstream/ Upstream link (Mbps/kbps)	**Throughput (Mbps)**	**Regeneration factor**
8/64	7.51	4
8/64	7.40	3
8/64	6.98	2
8/64	6.65	1

Table 3: SmartAckDropper and Ack Regenerator

Although this performance is the result of a trivial uni-directional flow, in reality one will find multiple flows, both uni-directional and bi-directional. We focus on the latter here for brevity. Please refer our detailed technical report for a full suite of simulations [techreport].

4.2 Bi-Directional Flows

The problems associated with bi-directional transfers are unique and require a slightly different approach. Here, link utilization on *both* channels is key performance indicator. Balakrishnan et al [hari] proposed that an acks-first scheduling policy be used. However, we find that if acks are given priority, the rate of packets on the reverse (low-speed) channel drops to unacceptably low values. Yet if packets are given priority, the effects of asymmetry lead to low packet throughput on the forward (high-speed) channel and beyond a point, this cannot be compensated by ack-regulation schemes (since the behavior is also dependent on TCP window dynamics).

Table 4 shows performance of a single FTP flow in each direction (baseline), with FIFO queuing at the upstream link. Here, the presence of bi-directional traffic reduces downstream throughput (and increases

[1] (8 Mbps/64 kbps)/(1000 bytes/40 bytes) = 5

asymmetry) by almost a factor of 10. Downstream throughput worst case is 18.8 kbps, compared to 1.5 Mbps in the uni-directional model (Table 1). Even in the case of 640 kbps upstream, downstream throughput is reduced to 7.4% available capacity. This drop can be attributed almost entirely to increased asymmetry and bidirectional issues, as there were no lost packets observed.

Downstream/ Upstream link (Mbps/kbps)	**Throughput (kbps)**	**Link Utilization**	**K**
8/64	18.8/58.8	0.235% / 91.875%	5
8/256	255.2/232	3.19%/90.63%	1.25
8/640	588/579.6	7.35%/90.56%	0.5

Table 4: Single Flow, Bi-directional Traffic.

The cleanest way to improve performance in such cases is to set up one queue for acks and one for packets in both directions, serviced with a link-sharing scheduler (CBQ: Class-Based Queuing) rather than a priority scheduler. On the ack queue on the reverse direction, both SAD and AR are applied. CBQ assigns fractions, F and G respectively, to the packet queues on the reverse and forward links respectively. Based on analysis in section 2, we set F large — as large as 99.7%. The setting of G is a policy/performance issue which we investigate here.

Table 6 shows performance using a combination of CBQ, SAD and AR for different values of G and degrees of asymmetry. Aggregate (packet + ack) throughput for both downstream and upstream directions is shown. Note that the performance in all cases has increased ***by a factor of 10-20*** compared to corresponding values in Table 4. Furthermore, as G increases from 0.1 to 0.9, the upstream aggregate throughput increases (and in some cases nearly doubles or triples) at the expense of downstream throughput (20-50% throughput loss), hence leading to a policy/performance tradeoff.

Downstream/Upstream link (Mbps/kbps)	**Aggregate Throughput (Mbps/kbps)**	**G**
8/64	3.39/34.4	0.1
8/64	3.73/44.4	0.5
8/64	6.43/54	0.9
Downstream/Upstream link	**Aggregate Throughput**	**G**
8/256	5.46/84.4	0.1
8/256	5.59/149.2	0.5
8/256	6.55/217.2	0.9
Downstream/Upstream link	**Aggregate Throughput**	**G**
8/640	6.7/333.6	0.1
8/640	6.65/336.8	0.5
8/640	7.02/542.8	0.9

Table 5: Effect of SAD+AR+CBQ

4.3 **ATM vs IP on the Local Loop**

ATM has been criticized for its “cell-tax” of 10-20% on transport. But we find that use of ack-regulation schemes makes the issue of ATM cell-tax moot when applied on the local loop, i.e., the cell-tax is not the dominant issue determining performance. While the cell tax increases the normalized asymmetry ratio (K), it can easily be compensated for by using the SAD+AR techniques discussed earlier. Table 7 shows that the percentage change in K, when ATM is used, is no more than 16%, which has been compensated by the SAD/AR scheme while achieving maximum levels of link utilization.

Downstrm / upstrm link (Mbps/kbps)	Packet size (bytes)	New packet size[2]	K w/o ATM	K w/ ATM	% change link k
8/64	1024	1174	4.88	5.64	0.16
8/128	1024	1174	2.42	2.8	0.16
8/256	1024	1174	1.21	1.4	0.16
8/640	1024	1174	0.47	0.54	0.15

Table 6: Normalized Asymmetry with ATM Fragmentation and Reassembly

Summary

We presented a simple AMP model and an improved ack-regulation scheme called SAD[3] to explain and improve the performance of TCP/IP over ADSL channels. We also propose the use of link-sharing schedulers with just two queues (ack and packet queues, with SAD implemented on the ack queues) at the ATU-R to effectively support bidirectional Internet traffic. The percentage gains in performance can range from 100%-2000% . However arbitrary degrees of asymmetries cannot be solved by ack regulation schemes because performance is ultimately dictated TCP dynamics as well.

References

[lakshman] T. V. Lakshman, U. Madhow, B. Suter, "Window-based Error Recovery and Flow control with a Slow Acknowledgement Channel: a Study of TCP/IP Performance", Proceedings of INFOCOM '97, April 1997.

[hari] Hari Balakrishnan, Venkata N. Padmanabhan, and Randy H. Katz. "The Effects of Asymmetry on TCP Performance," ACM Mobile Networks and Applications (MONET) Journal, to appear.

[cbq] Sally Floyd and Van Jacobson, "Link-sharing and Resource Management Models for Packet Networks," IEEE/ACM transactions on Networking, Vol 3 No. 4, August 1995.

[keshav] Keshav Srinivasan "Method and Apparatus for collapsing TCP Acks on Asymmetric Conditions" US patent number 5,793,768 ,Issued Aug. 11, 1998, Filed Aug. 13, 1996

[pep] G Montenegro, Jim Griner, John Border, Markku Kojo, "Performance Enhancing Proxies ", draft-ietf-pilc-pep-00.txt, July 1999.

[tcp-cong]M. Allman, V. Paxson and W. Stevens, TCP Congestion Control, RFC 2581, Proposed Standard, April 1999.

[techreport] D. Shekhar, S. Kalyanaraman, K. Kidambi, “TCP/IP over ADSL,” Technical report, Available from http://www.ecse.rpi.edu/Homepages/shivkuma

[2] IP/ATM/LLC overhead only

[3] Pulsecom has applied for a patent on the SAD scheme & algorithm