Network Working Group
Request for Comments: 3449
BCP: 69
Category: Best Current Practice

H. Balakrishnan
MIT LCS
V. N. Padmanabhan
Microsoft Research
G. Fairhurst
M. Sooriyabandara
University of Aberdeen, U.K.

December 2002

TCP Performance Implications of Network Path Asymmetry

Status of this Memo

Copyright Notice

Abstract

This document describes TCP performance problems that arise because of asymmetric effects. These problems arise in several access networks, including bandwidth-asymmetric networks and packet radio subnetworks, for different underlying reasons. However, the end result on TCP performance is the same in both cases: performance often degrades significantly because of imperfection and variability in the ACK feedback from the receiver to the sender.

The document details several mitigations to these effects, which have either been proposed or evaluated in the literature, or are currently deployed in networks. These solutions use a combination of local link-layer techniques, subnetwork, and end-to-end mechanisms, consisting of: (i) techniques to manage the channel used for the upstream bottleneck link carrying the ACKs, typically using header compression or reducing the frequency of TCP ACKs, (ii) techniques to handle this reduced ACK frequency to retain the TCP sender's acknowledgment-triggered self-clocking and (iii) techniques to schedule the data and ACK packets in the reverse direction to improve performance in the presence of two-way traffic. Each technique is described, together with known issues, and recommendations for use. A summary of the recommendations is provided at the end of the document.

Table of Contents

1. Conventions used in this Document

FORWARD DIRECTION: The dominant direction of data transfer over an asymmetric network path. It corresponds to the direction with better characteristics in terms of capacity, latency, error rate, etc. Data transfer in the forward direction is called "forward transfer". Packets travelling in the forward direction follow the forward path through the IP network.

REVERSE DIRECTION: The direction in which acknowledgments of a forward TCP transfer flow. Data transfer could also happen in this direction (and is termed "reverse transfer"), but it is typically less voluminous than that in the forward direction. The reverse direction typically exhibits worse characteristics than the forward direction. Packets travelling in the reverse direction follow the reverse path through the IP network.

UPSTREAM LINK: The specific bottleneck link that normally has much less capability than the corresponding downstream link. Congestion is not confined to this link alone, and may also occur at any point along the forward and reverse directions (e.g., due to sharing with other traffic flows).

DOWNSTREAM LINK: A link on the forward path, corresponding to the upstream link.

ACK: A cumulative TCP acknowledgment [RFC791]. In this document, this term refers to a TCP segment that carries a cumulative acknowledgement (ACK), but no data.

DELAYED ACK FACTOR, d: The number of TCP data segments acknowledged by a TCP ACK. The minimum value of d is 1, since at most one ACK should be sent for each data packet [RFC1122, RFC2581].

STRETCH ACK: Stretch ACKs are acknowledgements that cover more than 2 segments of previously unacknowledged data (d>2) [RFC2581]. Stretch ACKs can occur by design (although this is not standard), due to implementation bugs [All97b, RFC2525], or due to ACK loss [RFC2760].

NORMALIZED BANDWIDTH RATIO, k: The ratio of the raw bandwidth (capacity) of the forward direction to the return direction, divided by the ratio of the packet sizes used in the two directions [LMS97].

SOFTSTATE: Per-flow state established in a network device that is used by the protocol [Cla88]. The state expires after a period of time (i.e., is not required to be explicitly deleted when a session

expires), and is continuously refreshed while a flow continues (i.e., lost state may be reconstructed without needing to exchange additional control messages).

2. Motivation

Asymmetric characteristics are exhibited by several network technologies, including cable data networks, (e.g., DOCSIS cable TV networks [DS00, DS01]), direct broadcast satellite (e.g., an IP service using Digital Video Broadcast, DVB, [EN97] with an interactive return channel), Very Small Aperture satellite Terminals (VSAT), Asymmetric Digital Subscriber Line (ADSL) [ITU02, ANS01], and several packet radio networks. These networks are increasingly being deployed as high-speed Internet access networks, and it is therefore highly desirable to achieve good TCP performance. However, the asymmetry of the network paths often makes this challenging. Examples of some networks that exhibit asymmetry are provided in the Appendix.

Asymmetry may manifest itself as a difference in transmit and receive capacity, an imbalance in the packet loss rate, or differences between the transmit and receive paths [RFC3077]. For example, when capacity is asymmetric, such that there is reduced capacity on reverse path used by TCP ACKs, slow or infrequent ACK feedback degrades TCP performance in the forward direction. Similarly, asymmetry in the underlying Medium Access Control (MAC) and Physical (PHY) protocols could make it expensive to transmit TCP ACKs (disproportionately to their size), even when capacity is symmetric.

2.1 Asymmetry due to Differences in Transmit and Receive Capacity

Network paths may be asymmetric because the upstream and downstream links operate at different rates and/or are implemented using different technologies.

The asymmetry in capacity may be substantially increased when best effort IP flows carrying TCP ACKs share the available upstream capacity with other traffic flows, e.g., telephony, especially flows that have reserved upstream capacity. This includes service guarantees at the IP layer (e.g., the Guaranteed Service [RFC2212]) or at the subnet layer (e.g., support of Voice over IP [ITU01] using the Unsolicited Grant service in DOCSIS [DS01], or CBR virtual connections in ATM over ADSL [ITU02, ANS01]).

When multiple upstream links exist the asymmetry may be reduced by dividing upstream traffic between a number of available upstream links.

2.2 Asymmetry due to Shared Media in the Reverse Direction

In networks employing centralized multiple access control, asymmetry may be a fundamental consequence of the hub-and-spokes architecture of the network (i.e., a single base node communicating with multiple downstream nodes). The central node often incurs less transmission overhead and does not incur latency in scheduling its own downstream transmissions. In contrast, upstream transmission is subject to additional overhead and latency (e.g., due to guard times between transmission bursts, and contention intervals). This can produce significant network path asymmetry.

Upstream capacity may be further limited by the requirement that each node must first request per-packet bandwidth using a contention MAC protocol (e.g., DOCSIS 1.0 MAC restricts each node to sending at most a single packet in each upstream time-division interval [DS00]). A satellite network employing dynamic Bandwidth on Demand (BoD), also consumes MAC resources for each packet sent (e.g., [EN00]). In these schemes, the available uplink capacity is a function of the MAC algorithm. The MAC and PHY schemes also introduce overhead per upstream transmission which could be so significant that transmitting short packets (including TCP ACKs) becomes as costly as transmitting MTU-sized data packets.

2.3 The General Problem

Despite the technological differences between capacity-dependent and MAC-dependent asymmetries, both kinds of network path suffer reduced TCP performance for the same fundamental reason: the imperfection and variability of ACK feedback. This document discusses the problem in detail and describes several techniques that may reduce or eliminate the constraints.

3. How does Asymmetry Degrade TCP Performance?

This section describes the implications of network path asymmetry on TCP performance. The reader is referred to [BPK99, Bal98, Pad98, FSS01, Sam99] for more details and experimental results.

3.1 Asymmetric Capacity

The problems that degrade unidirectional transfer performance when the forward and return paths have very different capacities depend on the characteristics of the upstream link. Two types of situations arise for unidirectional traffic over such network paths: when the upstream bottleneck link has sufficient queuing to prevent packet (ACK) losses, and when the upstream bottleneck link has a small buffer. Each is considered in turn.

If the upstream bottleneck link has deep queues, so that this does not drop ACKs in the reverse direction, then performance is a strong function of the normalized bandwidth ratio, k. For example, for a 10 Mbps downstream link and a 50 Kbps upstream link, the raw capacity ratio is 200. With 1000-byte data packets and 40-byte ACKs, the ratio of the packet sizes is 25. This implies that k is 200/25 = 8. Thus, if the receiver acknowledges more frequently than one ACK every 8 (k) data packets, the upstream link will become saturated before the downstream link, limiting the throughput in the forward direction. Note that, the achieved TCP throughput is determined by the minimum of the receiver advertised window or TCP congestion window, cwnd [RFC2581].

If ACKs are not dropped (at the upstream bottleneck link) and k > 1 or k > 0.5 when delayed ACKs are used [RFC1122], TCP ACK-clocking breaks down. Consider two data packets transmitted by the sender in quick succession. En route to the receiver, these packets get spaced apart according to the capacity of the smallest bottleneck link in the forward direction. The principle of ACK clocking is that the ACKs generated in response to receiving these data packets reflects this temporal spacing all the way back to the sender, enabling it to transmit new data packets that maintain the same spacing [Jac88]. ACK clocking with delayed ACKs, reflects the spacing between data packets that actually trigger ACKs. However, the limited upstream capacity and queuing at the upstream bottleneck router alters the inter-ACK spacing of the reverse path, and hence that observed at the sender. When ACKs arrive at the upstream bottleneck link at a faster rate than the link can support, they get queued behind one another. The spacing between them when they emerge from the link is dilated with respect to their original spacing, and is a function of the upstream bottleneck capacity. Thus the TCP sender clocks out new data packets at a slower rate than if there had been no queuing of ACKs. The performance of the connection is no longer dependent on the downstream bottleneck link alone; instead, it is throttled by the rate of arriving ACKs. As a side effect, the sender's rate of cwnd growth also slows down.

A second side effect arises when the upstream bottleneck link on the reverse path is saturated. The saturated link causes persistent queuing of packets, leading to an increasing path Round Trip Time (RTT) [RFC2998] observed by all end hosts using the bottleneck link. This can impact the protocol control loops, and may also trigger false time out (underestimation of the path RTT by the sending host).

A different situation arises when the upstream bottleneck link has a relatively small amount of buffer space to accommodate ACKs. As the transmission window grows, this queue fills, and ACKs are dropped. If the receiver were to acknowledge every packet, only one of every k

ACKs would get through to the sender, and the remaining (k-1) are dropped due to buffer overflow at the upstream link buffer (here k is the normalized bandwidth ratio as before). In this case, the reverse bottleneck link capacity and slow ACK arrival rate are not directly responsible for any degraded performance. However, the infrequency of ACKs leads to three reasons for degraded performance:

1. The sender transmits data in large bursts of packets, limited only by the available cwnd. If the sender receives only one ACK in k, it transmits data in bursts of k (or more) packets because each ACK shifts the sliding window by at least k (acknowledged) data packets (TCP data segments). This increases the likelihood of data packet loss along the forward path especially when k is large, because routers do not handle large bursts of packets well.

2. Current TCP sender implementations increase their cwnd by counting the number of ACKs they receive and not by how much data is actually acknowledged by each ACK. The later approach, also known as byte counting (section 4.7), is a standard implementation option for cwnd increase during the congestion avoidance period [RFC2581]. Thus fewer ACKs imply a slower rate of growth of the cwnd, which degrades performance over long-delay connections.

3. The sender TCP's Fast Retransmission and Fast Recovery algorithms [RFC2581] are less effective when ACKs are lost. The sender may possibly not receive the threshold number of duplicate ACKs even if the receiver transmits more than the DupACK threshold (> 3 DupACKs) [RFC2581]. Furthermore, the sender may possibly not receive enough duplicate ACKs to adequately inflate its cwnd during Fast Recovery.

3.2 MAC Protocol Interactions

The interaction of TCP with MAC protocols may degrade end-to-end performance. Variable round-trip delays and ACK queuing are the main symptoms of this problem.

One example is the impact on terrestrial wireless networks [Bal98]. A high per-packet overhead may arise from the need for communicating link nodes to first synchronise (e.g., via a Ready To Send / Clear to Send (RTS/CTS) protocol) before communication and the significant turn-around time for the wireless channel. This overhead is variable, since the RTS/CTS exchange may need to back-off exponentially when the remote node is busy (e.g., engaged in a conversation with a different node). This leads to large and variable communication latencies in packet-radio networks.

An asymmetric workload (more downstream than upstream traffic) may cause ACKs to be queued in some wireless nodes (especially in the end host modems), exacerbating the variable latency. Queuing may also occur in other shared media, e.g., cable modem uplinks, BoD access systems often employed on shared satellite channels.

Variable latency and ACK queuing reduces the smoothness of the TCP data flow. In particular, ACK traffic can interfere with the flow of data packets, increasing the traffic load of the system.

TCP measures the path RTT, and from this calculates a smoothed RTT estimate (srtt) and a linear deviation, rttvar. These are used to estimate a path retransmission timeout (RTO) [RFC2988], set to srtt + 4*rttvar. For most wired TCP connections, the srtt remains constant or has a low linear deviation. The RTO therefore tracks the path RTT, and the TCP sender will respond promptly when multiple losses occur in a window. In contrast, some wireless networks exhibit a high variability in RTT, causing the RTO to significantly increase (e.g., on the order of 10 seconds). Paths traversing multiple wireless hops are especially vulnerable to this effect, because this increases the probability that the intermediate nodes may already be engaged in conversation with other nodes. The overhead in most MAC schemes is a function of both the number and size of packets. However, the MAC contention problem is a significant function of the number of packets (e.g., ACKs) transmitted rather than their size. In other words, there is a significant cost to transmitting a packet regardless of packet size.

Experiments conducted on the Ricochet packet radio network in 1996 and 1997 demonstrated the impact of radio turnarounds and the corresponding increased RTT variability, resulting in degraded TCP performance. It was not uncommon for TCP connections to experience timeouts of 9 - 12 seconds, with the result that many connections were idle for a significant fraction of their lifetime (e.g., sometimes 35% of the total transfer time). This leads to under-utilization of the available capacity. These effects may also occur in other wireless subnetworks.

3.3 Bidirectional Traffic

Bidirectional traffic arises when there are simultaneous TCP transfers in the forward and reverse directions over an asymmetric network path, e.g., a user who sends an e-mail message in the reverse direction while simultaneously receiving a web page in the forward direction. To simplify the discussion, only one TCP connection in each direction is considered. In many practical cases, several simultaneous connections need to share the available capacity, increasing the level of congestion.

Bidirectional traffic makes the effects discussed in section 3.1 more pronounced, because part of the upstream link bandwidth is consumed by the reverse transfer. This effectively increases the degree of bandwidth asymmetry. Other effects also arise due to the interaction between data packets of the reverse transfer and ACKs of the forward transfer. Suppose at the time the forward TCP connection is initiated, the reverse TCP connection has already saturated the bottleneck upstream link with data packets. There is then a high probability that many ACKs of the new forward TCP connection will encounter a full upstream link buffer and hence get dropped. Even after these initial problems, ACKs of the forward connection could get queued behind large data packets of the reverse connection. The larger data packets may have correspondingly long transmission times (e.g., it takes about 280 ms to transmit a 1 Kbyte data packet over a 28.8 kbps line). This causes the forward transfer to stall for long periods of time. It is only at times when the reverse connection loses packets (due to a buffer overflow at an intermediate router) and slows down, that the forward connection gets the opportunity to make rapid progress and build up its cwnd.

When ACKs are queued behind other traffic for appreciable periods of time, the burst nature of TCP traffic and self-synchronizing effects can result in an effect known as ACK Compression [ZSC91], which reduces the throughput of TCP. It occurs when a series of ACKs, in one direction are queued behind a burst of other packets (e.g., data packets traveling in the same direction) and become compressed in time. This results in an intense burst of data packets in the other direction, in response to the burst of compressed ACKs arriving at the server. This phenomenon has been investigated in detail for bidirectional traffic, and recent analytical work [LMS97] has predicted ACK Compression may also result from bi-directional transmission with asymmetry, and was observed in practical asymmetric satellite subnetworks [FSS01]. In the case of extreme asymmetry (k>>1), the inter-ACK spacing can increase due to queuing (section 3.1), resulting in ACK dilation.

In summary, sharing of the upstream bottleneck link by multiple flows (e.g., IP flows to the same end host, or flows to a number of end hosts sharing a common upstream link) increases the level of ACK Congestion. The presence of bidirectional traffic exacerbates the constraints introduced by bandwidth asymmetry because of the adverse interaction between (large) data packets of a reverse direction connection and the ACKs of a forward direction connection.

3.4 Loss in Asymmetric Network Paths

Loss may occur in either the forward or reverse direction. For data transfer in the forward direction this results respectively in loss of data packets and ACK packets. Loss of ACKs is less significant than loss of data packets, because it generally results in stretch ACKs [CR98, FSS01].

In the case of long delay paths, a slow upstream link [RFC3150] can lead to another complication when the end host uses TCP large windows [RFC1323] to maximize throughput in the forward direction. Loss of data packets on the forward path, due to congestion, or link loss, common for some wireless links, will generate a large number of back-to-back duplicate ACKs (or TCP SACK packets [RFC2018]), for each correctly received data packet following a loss. The TCP sender employs Fast Retransmission and Recovery [RFC2581] to recover from the loss, but even if this is successful, the ACK to the retransmitted data segment may be significantly delayed by other duplicate ACKs still queued at the upstream link buffer. This can ultimately lead to a timeout [RFC2988] and a premature end to the TCP Slow Start [RFC2581]. This results in poor forward path throughput. Section 5.3 describes some mitigations to counter this.

4. Improving TCP Performance using Host Mitigations

There are two key issues that need to be addressed to improve TCP performance over asymmetric networks. The first is to manage the capacity of the upstream bottleneck link, used by ACKs and possibly other traffic. A number of techniques exist which work by reducing the number of ACKs that flow in the reverse direction. This has the side effect of potentially destroying the desirable self-clocking property of the TCP sender where transmission of new data packets is triggered by incoming ACKs. Thus, the second issue is to avoid any adverse impact of infrequent ACKs.

Each of these issues can be handled by local link-layer solutions and/or by end-to-end techniques. This section discusses end-to-end modifications. Some techniques require TCP receiver changes (sections 4.1 4.4, 4.5), some require TCP sender changes (sections 4.6, 4.7), and a pair requires changes to both the TCP sender and receiver (sections 4.2, 4.3). One technique requires a sender modification at the receiving host (section 4.8). The techniques may be used independently, however some sets of techniques are complementary, e.g., pacing (section 4.6) and byte counting (section 4.7) which have been bundled into a single TCP Sender Adaptation scheme [BPK99].

It is normally envisaged that these changes would occur in the end hosts using the asymmetric path, however they could, and have, been used in a middle-box or Protocol Enhancing Proxy (PEP) [RFC3135] employing split TCP. This document does not discuss the issues concerning PEPs. Section 4 describes several techniques, which do not require end-to-end changes.

4.1 Modified Delayed ACKs

There are two standard methods that can be used by TCP receivers to generate acknowledgments. The method outlined in [RFC793] generates an ACK for each incoming data segment (i.e., d=1). [RFC1122] states that hosts should use "delayed acknowledgments". Using this algorithm, an ACK is generated for at least every second full-sized segment (d=2), or if a second full-sized segment does not arrive within a given timeout (which must not exceed 500 ms [RFC1122], and is typically less than 200 ms). Relaxing the latter constraint (i.e., allowing d>2) may generate Stretch ACKs [RFC2760]. This provides a possible mitigation, which reduces the rate at which ACKs are returned by the receiver. An implementer should only deviate from this requirement after careful consideration of the implications [RFC2581].

Reducing the number of ACKs per received data segment has a number of undesirable effects including:

(i) Increased path RTT
(ii) Increased time for TCP to open the cwnd
(iii) Increased TCP sender burst size, since cwnd opens in larger steps

In addition, a TCP receiver is often unable to determine an optimum setting for a large d, since it will normally be unaware of the details of the properties of the links that form the path in the reverse direction.

RECOMMENDATION: A TCP receiver must use the standard TCP algorithm for sending ACKs as specified in [RFC2581]. That is, it may delay sending an ACK after it receives a data segment [RFC1122]. When ACKs are delayed, the receiver must generate an ACK within 500 ms and the ACK should be generated for at least every second full sized segment (MSS) of received data [RFC2581]. This will result in an ACK delay factor (d) that does not exceed a value of 2. Changing the algorithm would require a host modification to the TCP receiver and awareness by the receiving host that it is using a connection with an asymmetric path. Such a change has many drawbacks in the general case and is currently not recommended for use within the Internet.

4.2 Use of Large MSS

A TCP sender that uses a large Maximum Segment Size (MSS) reduces the number of ACKs generated per transmitted byte of data.

Although individual subnetworks may support a large MTU, the majority of current Internet links employ an MTU of approx 1500 bytes (that of Ethernet). By setting the Don't Fragment (DF) bit in the IP header, Path MTU (PMTU) discovery [RFC1191] may be used to determine the maximum packet size (and hence MSS) a sender can use on a given network path without being subjected to IP fragmentation, and provides a way to automatically select a suitable MSS for a specific path. This also guarantees that routers will not perform IP fragmentation of normal data packets.

By electing not to use PMTU Discovery, an end host may choose to use IP fragmentation by routers along the path in the forward direction [RFC793]. This allows an MSS larger than smallest MTU along the path. However, this increases the unit of error recovery (TCP segment) above the unit of transmission (IP packet). This is not recommended, since it can increase the number of retransmitted packets following loss of a single IP packet, leading to reduced efficiency, and potentially aggravating network congestion [Ken87]. Choosing an MSS larger than the forward path minimum MTU also permits the sender to transmit more initial packets (a burst of IP fragments for each TCP segment) when a session starts or following RTO expiry, increasing the aggressiveness of the sender compared to standard TCP [RFC2581]. This can adversely impact other standard TCP sessions that share a network path.

RECOMMENDATION:

A larger forward path MTU is desirable for paths with bandwidth asymmetry. Network providers may use a large MTU on links in the forward direction. TCP end hosts using Path MTU discovery may be able to take advantage of a large MTU by automatically selecting an appropriate larger MSS, without requiring modification. The use of Path MTU discovery [RFC1191] is therefore recommended.

Increasing the unit of error recovery and congestion control (MSS) above the unit of transmission and congestion loss (the IP packet) by using a larger end host MSS and IP fragmentation in routers is not recommended.

4.3 ACK Congestion Control

ACK Congestion Control (ACC) is an experimental technique that operates end to end. ACC extends congestion control to ACKs, since they may make non-negligible demands on resources (e.g., packet buffers, and MAC transmission overhead) at an upstream bottleneck link. It has two parts: (a) a network mechanism indicating to the receiver that the ACK path is congested, and (b) the receiver's response to such an indication.

A router feeding an upstream bottleneck link may detect incipient congestion, e.g., using an algorithm based on RED (Random Early Detection) [FJ93]. This may track the average queue size over a time window in the recent past. If the average exceeds a threshold, the router may select a packet at random. If the packet IP header has the Explicit Congestion Notification Capable Transport (ECT) bit set, the router may mark the packet, i.e., sets an Explicit Congestion Notification (ECN) [RFC3168] bit(s) in the IP header, otherwise the packet is normally dropped. The ECN notification received by the end host is reflected back to the sending TCP end host, to trigger congestion avoidance [RFC3168]. Note that routers implementing RED with ECN, do not eliminate packet loss, and may drop a packet (even when the ECT bit is set). It is also possible to use an algorithm other than RED to decide when to set the ECN bit.

ACC extends ECN so that both TCP data packets and ACKs set the ECT bit and are thus candidates for being marked with an ECN bit. Therefore, upon receiving an ACK with the ECN bit set [RFC3168], a TCP receiver reduces the rate at which it sends ACKs. It maintains a dynamically varying delayed-ACK factor, d, and sends one ACK for every d data packets received. When it receives a packet with the ECN bit set, it increases d multiplicatively, thereby multiplicatively decreasing the frequency of ACKs. For each subsequent RTT (e.g., determined using the TCP RTTM option [RFC1323]) during which it does not receive an ECN, it linearly decreases the factor d, increasing the frequency of ACKs. Thus, the receiver mimics the standard congestion control behavior of TCP senders in the manner in which it sends ACKs.

The maximum value of d is determined by the TCP sender window size, which could be conveyed to the receiver in a new (experimental) TCP option. The receiver should send at least one ACK (preferably more) for each window of data from the sender (i.e., d < (cwnd/mss)) to prevent the sender from stalling until the receiver's delayed ACK timer triggers an ACK to be sent.

RECOMMENDATION: ACK Congestion Control (ACC) is an experimental technique that requires TCP sender and receiver modifications. There is currently little experience of using such techniques in the Internet. Future versions of TCP may evolve to include this or similar techniques. These are the subject of ongoing research. ACC is not recommended for use within the Internet in its current form.

4.4 Window Prediction Mechanism

The Window Prediction Mechanism (WPM) is a TCP receiver side mechanism [CLP98] that uses a dynamic ACK delay factor (varying d) resembling the ACC scheme (section 4.3). The TCP receiver reconstructs the congestion control behavior of the TCP sender by predicting a cwnd value. This value is used along with the allowed window to adjust the receiver's value of d. WPM accommodates for unnecessary retransmissions resulting from losses due to link errors.

RECOMMENDATION: Window Prediction Mechanism (WPM) is an experimental TCP receiver side modification. There is currently little experience of using such techniques in the Internet. Future versions of TCP may evolve to include this or similar techniques. These are the subjects of ongoing research. WPM is not recommended for use within the Internet in its current form.

4.5 Acknowledgement based on Cwnd Estimation.

Acknowledgement based on Cwnd Estimation (ACE) [MJW00] attempts to measure the cwnd at the TCP receiver and maintain a varying ACK delay factor (d). The cwnd is estimated by counting the number of packets received during a path RTT. The technique may improve accuracy of prediction of a suitable cwnd.

RECOMMENDATION: Acknowledgement based on Cwnd Estimation (ACE) is an experimental TCP receiver side modification. There is currently little experience of using such techniques in the Internet. Future versions of TCP may evolve to include this or similar techniques. These are the subject of ongoing research. ACE is not recommended for use within the Internet in its current form.

4.6 TCP Sender Pacing

Reducing the frequency of ACKs may alleviate congestion of the upstream bottleneck link, but can lead to increased size of TCP sender bursts (section 4.1). This may slow the growth of cwnd, and is undesirable when used over shared network paths since it may significantly increase the maximum number of packets in the bottleneck link buffer, potentially resulting in an increase in network congestion. This may also lead to ACK Compression [ZSC91].

TCP Pacing [AST00], generally referred to as TCP Sender pacing, employs an adapted TCP sender to alleviating transmission burstiness. A bound is placed on the maximum number of packets the TCP sender can transmit back-to-back (at local line rate), even if the window(s) allow the transmission of more data. If necessary, more bursts of data packets are scheduled for later points in time computed based on the transmission rate of the TCP connection. The transmission rate may be estimated from the ratio cwnd/srtt. Thus, large bursts of data packets get broken up into smaller bursts spread over time.

A subnetwork may also provide pacing (e.g., Generic Traffic Shaping (GTS)), but implies a significant increase in the per-packet processing overhead and buffer requirement at the router where shaping is performed (section 5.3.3).

RECOMMENDATIONS: TCP Sender Pacing requires a change to implementation of the TCP sender. It may be beneficial in the Internet and will significantly reduce the burst size of packets transmitted by a host. This successfully mitigates the impact of receiving Stretch ACKs. TCP Sender Pacing implies increased processing cost per packet, and requires a prediction algorithm to suggest a suitable transmission rate. There are hence performance trade-offs between end host cost and network performance. Specification of efficient algorithms remains an area of ongoing research. Use of TCP Sender Pacing is not expected to introduce new problems. It is an experimental mitigation for TCP hosts that may control the burstiness of transmission (e.g., resulting from Type 1 techniques, section 5.1.2), however it is not currently widely deployed. It is not recommended for use within the Internet in its current form.

4.7 TCP Byte Counting

The TCP sender can avoid slowing growth of cwnd by taking into account the volume of data acknowledged by each ACK, rather than opening the cwnd based on the number of received ACKs. So, if an ACK acknowledges d data packets (or TCP data segments), the cwnd would grow as if d separate ACKs had been received. This is called TCP Byte Counting [RFC2581, RFC2760]. (One could treat the single ACK as being equivalent to d/2, instead of d ACKs, to mimic the effect of the TCP delayed ACK algorithm.) This policy works because cwnd growth is only tied to the available capacity in the forward direction, so the number of ACKs is immaterial.

This may mitigate the impact of asymmetry when used in combination with other techniques (e.g., a combination of TCP Pacing (section4.6), and ACC (section 4.3) associated with a duplicate ACK threshold at the receiver.)

The main issue is that TCP byte counting may generate undesirable long bursts of TCP packets at the sender host line rate. An implementation must also consider that data packets in the forward direction and ACKs in the reverse direction may both travel over network paths that perform some amount of packet reordering. Reordering of IP packets is currently common, and may arise from various causes [BPS00].

RECOMMENDATION: TCP Byte Counting requires a small TCP sender modification. In its simplest form, it can generate large bursts of TCP data packets, particularly when Stretch ACKs are received. Unlimited byte counting is therefore not allowed [RFC2581] for use within the Internet.

It is therefore strongly recommended [RFC2581, RFC2760] that any byte counting scheme should include a method to mitigate the potentially large bursts of TCP data packets the algorithm can cause (e.g., TCP Sender Pacing (section 4.6), ABC [abc-ID]). If the burst size or sending rate of the TCP sender can be controlled then the scheme may be beneficial when Stretch ACKs are received. Determining safe algorithms remain an area of ongoing research. Further experimentation will then be required to assess the success of these safeguards, before they can be recommended for use in the Internet.

4.8 Backpressure

Backpressure is a technique to enhance the performance of bidirectional traffic for end hosts directly connected to the upstream bottleneck link [KVR98]. A limit is set on how many data packets of upstream transfers can be enqueued at the upstream bottleneck link. In other words, the bottleneck link queue exerts 'backpressure' on the TCP (sender) layer. This requires a modified implementation, compared to that currently deployed in many TCP stacks. Backpressure ensures that ACKs of downstream connections do not get starved at the upstream bottleneck, thereby improving performance of the downstream connections. Similar generic schemes that may be implemented in hosts/routers are discussed in section 5.4.

Backpressure can be unfair to a reverse direction connection and make its throughput highly sensitive to the dynamics of the forward connection(s).

RECOMMENDATION: Backpressure requires an experimental modification to the sender protocol stack of a host directly connected to an upstream bottleneck link. Use of backpressure is an implementation issue, rather than a network protocol issue. Where backpressure is implemented, the optimizations described in this section could be

desirable and can benefit bidirectional traffic for hosts. Specification of safe algorithms for providing backpressure is still a subject of ongoing research. The technique is not recommended for use within the Internet in its current form.

5. Improving TCP performance using Transparent Modifications

Various link and network layer techniques have been suggested to mitigate the effect of an upstream bottleneck link. These techniques may provide benefit without modification to either the TCP sender or receiver, or may alternately be used in conjunction with one or more of the schemes identified in section 4. In this document, these techniques are known as "transparent" [RFC3135], because at the transport layer, the TCP sender and receiver are not necessarily aware of their existence. This does not imply that they do not modify the pattern and timing of packets as observed at the network layer. The techniques are classified here into three types based on the point at which they are introduced.

Most techniques require the individual TCP connections passing over the bottleneck link(s) to be separately identified and imply that some per-flow state is maintained for active TCP connections. A link scheduler may also be employed (section 5.4). The techniques (with one exception, ACK Decimation (section 5.2.2) require:

(i) Visibility of an unencrypted IP and TCP packet header (e.g., no use of IPSec with payload encryption [RFC2406]).
(ii) Knowledge of IP/TCP options and ability to inspect packets with tunnel encapsulations (e.g., [RFC2784]) or to suspend processing of packets with unknown formats.
(iii) Ability to demultiplex flows (by using address/protocol/port number, or an explicit flow-id).

[RFC3135] describes a class of network device that provides more than forwarding of packets, and which is known as a Protocol Enhancing Proxy (PEP). A large spectrum of PEP devices exists, ranging from simple devices (e.g., ACK filtering) to more sophisticated devices (e.g., stateful devices that split a TCP connection into two separate parts). The techniques described in section 5 of this document belong to the simpler type, and do not inspect or modify any TCP or UDP payload data. They also do not modify port numbers or link addresses. Many of the risks associated with more complex PEPs do not exist for these schemes. Further information about the operation and the risks associated with using PEPs are described in [RFC3135].

5.1 TYPE 0: Header Compression

A client may reduce the volume of bits used to send a single ACK by using compression [RFC3150, RFC3135]. Most modern dial-up modems support ITU-T V.42 bulk compression. In contrast to bulk compression, header compression is known to be very effective at reducing the number of bits sent on the upstream link [RFC1144]. This relies on the observation that most TCP packet headers vary only in a few bit positions between successive packets in a flow, and that the variations can often be predicted.

5.1.1 TCP Header Compression

TCP header compression [RFC1144] (sometimes known as V-J compression) is a Proposed Standard describing use over low capacity links running SLIP or PPP [RFC3150]. It greatly reduces the size of ACKs on the reverse link when losses are infrequent (a situation that ensures that the state of the compressor and decompressor are synchronized). However, this alone does not address all of the asymmetry issues:

(i) In some (e.g., wireless) subnetworks there is a significant per-packet MAC overhead that is independent of packet size (section 3.2).
(ii) A reduction in the size of ACKs does not prevent adverse interaction with large upstream data packets in the presence of bidirectional traffic (section 3.3).
(iii) TCP header compression cannot be used with packets that have IP or TCP options (including IPSec [RFC2402, RFC2406], TCP RTTM [RFC1323], TCP SACK [RFC2018], etc.).
(iv) The performance of header compression described by RFC1144 is significantly degraded when compressed packets are lost. An improvement, which can still incur significant penalty on long network paths is described in [RFC2507]. This suggests it should only be used on links (or paths) that experience a low level of packet loss [RFC3150].
(v) The normal implementation of Header Compression inhibits compression when IP is used to support tunneling (e.g., L2TP, GRE [RFC2794], IP-in-IP). The tunnel encapsulation complicates locating the appropriate packet headers. Although GRE allows Header Compression on the inner (tunneled) IP header [RFC2784], this is not recommended, since loss of a packet (e.g., due to router congestion along the tunnel path) will result in discard of all packets for one RTT [RFC1144].

RECOMMENDATION: TCP Header Compression is a transparent modification performed at both ends of the upstream bottleneck link. It offers no benefit for flows employing IPSec [RFC2402, RFC2406], or when additional protocol headers are present (e.g., IP or TCP options,

and/or tunnel encapsulation headers). The scheme is widely implemented and deployed and used over Internet links. It is recommended to improve TCP performance for paths that have a low-to-medium bandwidth asymmetry (e.g., k<10).

In the form described in [RFC1144], TCP performance is degraded when used over links (or paths) that may exhibit appreciable rates of packet loss [RFC3150]. It may also not provide significant improvement for upstream links with bidirectional traffic. It is therefore not desirable for paths that have a high bandwidth asymmetry (e.g., k>10).

5.1.2 Alternate Robust Header Compression Algorithms

TCP header compression [RFC1144] and IP header compression [RFC2507] do not perform well when subject to packet loss. Further, they do not compress packets with TCP option fields (e.g., SACK [RFC2018] and Timestamp (RTTM) [RFC1323]). However, recent work on more robust schemes suggest that a new generation of compression algorithms may be developed which are much more robust. The IETF ROHC working group has specified compression techniques for UDP-based traffic [RFC3095] and is examining a number of schemes that may provide improve TCP header compression. These could be beneficial for asymmetric network paths.

RECOMMENDATION: Robust header compression is a transparent modification that may be performed at both ends of an upstream bottleneck link. This class of techniques may also be suited to Internet paths that suffer low levels of re-ordering. The techniques benefit paths with a low-to-medium bandwidth asymmetry (e.g., k>10) and may be robust to packet loss.

Selection of suitable compression algorithms remains an area of ongoing research. It is possible that schemes may be derived which support IPSec authentication, but not IPSec payload encryption. Such schemes do not alone provide significant improvement in asymmetric networks with a high asymmetry and/or bidirectional traffic.

5.2 TYPE 1: Reverse Link Bandwidth Management

Techniques beyond Type 0 header compression are required to address the performance problems caused by appreciable asymmetry (k>>1). One set of techniques is implemented only at one point on the reverse direction path, within the router/host connected to the upstream bottleneck link. These use flow class or per-flow queues at the upstream link interface to manage the queue of packets waiting for transmission on the bottleneck upstream link.

This type of technique bounds the upstream link buffer queue size, and employs an algorithm to remove (discard) excess ACKs from each queue. This relies on the cumulative nature of ACKs (section 4.1). Two approaches are described which employ this type of mitigation.

5.2.1 ACK Filtering

ACK Filtering (AF) [DMT96, BPK99] (also known as ACK Suppression [SF98, Sam99, FSS01]) is a TCP-aware link-layer technique that reduces the number of ACKs sent on the upstream link. This technique has been deployed in specific production networks (e.g., asymmetric satellite networks [ASB96]). The challenge is to ensure that the sender does not stall waiting for ACKs, which may happen if ACKs are indiscriminately removed.

When an ACK from the receiver is about to be enqueued at a upstream bottleneck link interface, the router or the end host link layer (if the host is directly connected to the upstream bottleneck link) checks the transmit queue(s) for older ACKs belonging to the same TCP connection. If ACKs are found, some (or all of them) are removed from the queue, reducing the number of ACKs.

Some ACKs also have other functions in TCP [RFC1144], and should not be deleted to ensure normal operation. AF should therefore not delete an ACK that has any data or TCP flags set (SYN, RST, URG, and FIN). In addition, it should avoid deleting a series of 3 duplicate ACKs that indicate the need for Fast Retransmission [RFC2581] or ACKs with the Selective ACK option (SACK)[RFC2018] from the queue to avoid causing problems to TCP's data-driven loss recovery mechanisms. Appropriate treatment is also needed to preserve correct operation of ECN feedback (carried in the TCP header) [RFC3168].

A range of policies to filter ACKs may be used. These may be either deterministic or random (similar to a random-drop gateway, but should take into consideration the semantics of the items in the queue). Algorithms have also been suggested to ensure a minimum ACK rate to guarantee the TCP sender window is updated [Sam99, FSS01], and to limit the number of data packets (TCP segments) acknowledged by a Stretch ACK. Per-flow state needs to be maintained only for connections with at least one packet in the queue (similar to FRED [LM97]). This state is soft [Cla88], and if necessary, can easily be reconstructed from the contents of the queue.

The undesirable effect of delayed DupACKs (section 3.4) can be reduced by deleting duplicate ACKs above a threshold value [MJW00, CLP98] allowing Fast Retransmission, but avoiding early TCP timeouts, which may otherwise result from excessive queuing of DupACKs.

Future schemes may include more advanced rules allowing removal of selected SACKs [RFC2018]. Such a scheme could prevent the upstream link queue from becoming filled by back-to-back ACKs with SACK blocks. Since a SACK packet is much larger than an ACK, it would otherwise add significantly to the path delay in the reverse direction. Selection of suitable algorithms remains an ongoing area of research.

RECOMMENDATION: ACK Filtering requires a modification to the upstream link interface. The scheme has been deployed in some networks where the extra processing overhead (per ACK) may be compensated for by avoiding the need to modify TCP. ACK Filtering can generate Stretch ACKs resulting in large bursts of TCP data packets. Therefore on its own, it is not recommended for use in the general Internet.

ACK Filtering when used in combination with a scheme to mitigate the effect of Stretch ACKs (i.e., control TCP sender burst size) is recommended for paths with appreciable asymmetry (k>1) and/or with bidirectional traffic. Suitable algorithms to support IPSec authentication, SACK, and ECN remain areas of ongoing research.

5.2.2 ACK Decimation

ACK Decimation is based on standard router mechanisms. By using an appropriate configuration of (small) per-flow queues and a chosen dropping policy (e.g., Weighted Fair Queuing, WFQ) at the upstream bottleneck link, a similar effect to AF (section 5.2.1) may be obtained, but with less control of the actual packets which are dropped.

In this scheme, the router/host at the bottleneck upstream link maintains per-flow queues and services them fairly (or with priorities) by queuing and scheduling of ACKs and data packets in the reverse direction. A small queue threshold is maintained to drop excessive ACKs from the tail of each queue, in order to reduce ACK Congestion. The inability to identify special ACK packets (c.f., AF) introduces some major drawbacks to this approach, such as the possibility of losing DupACKs, FIN/ACK, RST packets, or packets carrying ECN information [RFC3168]. Loss of these packets does not significantly impact network congestion, but does adversely impact the performance of the TCP session observing the loss.

A WFQ scheduler may assign a higher priority to interactive traffic (providing it has a mechanism to identify such traffic) and provide a fair share of the remaining capacity to the bulk traffic. In the presence of bidirectional traffic, and with a suitable scheduling policy, this may ensure fairer sharing for ACK and data packets. An increased forward transmission rate is achieved over asymmetric links

by an increased ACK Decimation rate, leading to generation of Stretch ACKs. As in AF, TCP sender burst size increases when Stretch ACKs are received unless other techniques are used in combination with this technique.

This technique has been deployed in specific networks (e.g., a network with high bandwidth asymmetry supporting high-speed data services to in-transit mobile hosts [Seg00]). Although not optimal, it offered a potential mitigation applicable when the TCP header is difficult to identify or not visible to the link layer (e.g., due to IPSec encryption).

RECOMMENDATION: ACK Decimation uses standard router mechanisms at the upstream link interface to constrain the rate at which ACKs are fed to the upstream link. The technique is beneficial with paths having appreciable asymmetry (k>1). It is however suboptimal, in that it may lead to inefficient TCP error recovery (and hence in some cases degraded TCP performance), and provides only crude control of link behavior. It is therefore recommended that where possible, ACK Filtering should be used in preference to ACK Decimation.

When ACK Decimation is used on paths with an appreciable asymmetry (k>1) (or with bidirectional traffic) it increases the burst size of the TCP sender, use of a scheme to mitigate the effect of Stretch ACKs or control burstiness is therefore strongly recommended.

5.3 TYPE 2: Handling Infrequent ACKs

TYPE 2 mitigations perform TYPE 1 upstream link bandwidth management, but also employ a second active element which mitigates the effect of the reduced ACK rate and burstiness of ACK transmission. This is desirable when end hosts use standard TCP sender implementations (e.g., those not implementing the techniques in sections 4.6, 4.7).

Consider a path where a TYPE 1 scheme forwards a Stretch ACK covering d TCP packets (i.e., where the acknowledgement number is d*MSS larger than the last ACK received by the TCP sender). When the TCP sender receives this ACK, it can send a burst of d (or d+1) TCP data packets. The sender is also constrained by the current cwnd. Received ACKs also serve to increase cwnd (by at most one MSS).

A TYPE 2 scheme mitigates the impact of the reduced ACK frequency resulting when a TYPE 1 scheme is used. This is achieved by interspersing additional ACKs before each received Stretch ACK. The additional ACKs, together with the original ACK, provide the TCP sender with sufficient ACKs to allow the TCP cwnd to open in the same way as if each of the original ACKs sent by the TCP receiver had been forwarded by the reverse path. In addition, by attempting to restore

the spacing between ACKs, such a scheme can also restore the TCP self-clocking behavior, and reduce the TCP sender burst size. Such schemes need to ensure conservative behavior (i.e., should not introduce more ACKs than were originally sent) and reduce the probability of ACK Compression [ZSC91].

The action is performed at two points on the return path: the upstream link interface (where excess ACKs are removed), and a point further along the reverse path (after the bottleneck upstream link(s)), where replacement ACKs are inserted. This attempts to reconstruct the ACK stream sent by the TCP receiver when used in combination with AF (section 5.2.1), or ACK Decimation (section 5.2.2).

TYPE 2 mitigations may be performed locally at the receive interface directly following the upstream bottleneck link, or may alternatively be applied at any point further along the reverse path (this is not necessarily on the forward path, since asymmetric routing may employ different forward and reverse internet paths). Since the techniques may generate multiple ACKs upon reception of each individual Stretch ACK, it is strongly recommended that the expander implements a scheme to prevent exploitation as a "packet amplifier" in a Denial-of-Service (DoS) attack (e.g., to verify the originator of the ACK). Identification of the sender could be accomplished by appropriately configured packet filters and/or by tunnel authentication procedures (e.g., [RFC2402, RFC2406]). A limit on the number of reconstructed ACKs that may be generated from a single packet may also be desirable.

5.3.1 ACK Reconstruction

ACK Reconstruction (AR) [BPK99] is used in conjunction with AF (section 5.2.1). AR deploys a soft-state [Cla88] agent called an ACK Reconstructor on the reverse path following the upstream bottleneck link. The soft-state can be regenerated if lost, based on received ACKs. When a Stretch ACK is received, AR introduces additional ACKs by filling gaps in the ACK sequence. Some potential Denial-of-Service vulnerabilities may arise (section 6) and need to be addressed by appropriate security techniques.

The Reconstructor determines the number of additional ACKs, by estimating the number of filtered ACKs. This uses implicit information present in the received ACK stream by observing the ACK sequence number of each received ACK. An example implementation could set an ACK threshold, ackthresh, to twice the MSS (this assumes the chosen MSS is known by the link). The factor of two corresponds

to standard TCP delayed-ACK policy (d=2). Thus, if successive ACKs arrive separated by delta, the Reconstructor regenerates a maximum of ((delta/ackthresh) - 2) ACKs.

To reduce the TCP sender burst size and allow the cwnd to increase at a rate governed by the downstream link, the reconstructed ACKs must be sent at a consistent rate (i.e., temporal spacing between reconstructed ACKs). One method is for the Reconstructor to measure the arrival rate of ACKs using an exponentially weighted moving average estimator. This rate depends on the output rate from the upstream link and on the presence of other traffic sharing the link. The output of the estimator indicates the average temporal spacing for the ACKs (and the average rate at which ACKs would reach the TCP sender if there were no further losses or delays). This may be used by the Reconstructor to set the temporal spacing of reconstructed ACKs. The scheme may also be used in combination with TCP sender adaptation (e.g., a combination of the techniques in sections 4.6 and 4.7).

The trade-off in AR is between obtaining less TCP sender burstiness, and a better rate of cwnd increase, with a reduction in RTT variation, versus a modest increase in the path RTT. The technique cannot perform reconstruction on connections using IPSec (AH [RFC2402] or ESP [RFC2406]), since it is unable to generate appropriate security information. It also cannot regenerate other packet header information (e.g., the exact pattern of bits carried in the IP packet ECN field [RFC3168] or the TCP RTTM option [RFC1323]).

An ACK Reconstructor operates correctly (i.e., generates no spurious ACKs and preserves the end-to-end semantics of TCP), providing:

(i) the TCP receiver uses ACK Delay (d=2) [RFC2581]
(ii) the Reconstructor receives only in-order ACKs
(iii) all ACKs are routed via the Reconstructor
(iv) the Reconstructor correctly determines the TCP MSS used by the session
(v) the packets do not carry additional header information (e.g., TCP RTTM option [RFC1323], IPSec using AH [RFC2402]or ESP [RFC2406]).

RECOMMENDATION: ACK Reconstruction is an experimental transparent modification performed on the reverse path following the upstream bottleneck link. It is designed to be used in conjunction with a TYPE 1 mitigation. It reduces the burst size of TCP transmission in the forward direction, which may otherwise increase when TYPE 1 schemes are used alone. It requires modification of equipment after the upstream link (including maintaining per-flow soft state). The scheme introduces implicit assumptions about the network path and has

potential Denial-of-Service vulnerabilities (i.e., acting as a packet amplifier); these need to be better understood and addressed by appropriate security techniques.

Selection of appropriate algorithms to pace the ACK traffic remains an open research issue. There is also currently little experience of the implications of using such techniques in the Internet, and therefore it is recommended that this technique should not be used within the Internet in its current form.

5.3.2 ACK Compaction and Companding

ACK Compaction and ACK Companding [SAM99, FSS01] are techniques that operate at a point on the reverse path following the constrained ACK bottleneck. Like AR (section 5.3.1), ACK Compaction and ACK Companding are both used in conjunction with an AF technique (section 5.2.1) and regenerate filtered ACKs, restoring the ACK stream. However, they differ from AR in that they use a modified AF (known as a compactor or compressor), in which explicit information is added to all Stretch ACKs generated by the AF. This is used to explicitly synchronize the reconstruction operation (referred to here as expansion).

The modified AF combines two modifications: First, when the compressor deletes an ACK from the upstream bottleneck link queue, it appends explicit information (a prefix) to the remaining ACK (this ACK is marked to ensure it is not subsequently deleted). The additional information contains details the conditions under which ACKs were previously filtered. A variety of information may be encoded in the prefix. This includes the number of ACKs deleted by the AF and the average number of bytes acknowledged. This may subsequently be used by an expander at the remote end of the tunnel. Further timing information may also be added to control the pacing of the regenerated ACKs [FSS01]. The temporal spacing of the filtered ACKs may also be encoded.

To encode the prefix requires the subsequent expander to recognize a modified ACK header. This would normally limit the expander to link-local operation (at the receive interface of the upstream bottleneck link). If remote expansion is needed further along the reverse path, a tunnel may be used to pass the modified ACKs to the remote expander. The tunnel introduces extra overhead, however networks with asymmetric capacity and symmetric routing frequently already employ such tunnels (e.g., in a UDLR network [RFC3077], the expander may be co-located with the feed router).

ACK expansion uses a stateless algorithm to expand the ACK (i.e., each received packet is processed independently of previously received packets). It uses the prefix information together with the acknowledgment field in the received ACK, to produce an equivalent number of ACKs to those previously deleted by the compactor. These ACKs are forwarded to the original destination (i.e., the TCP sender), preserving normal TCP ACK clocking. In this way, ACK Compaction, unlike AR, is not reliant on specific ACK policies, nor must it see all ACKs associated with the reverse path (e.g., it may be compatible with schemes such as DAASS [RFC2760]).

Some potential Denial-of-Service vulnerabilities may arise (section 6) and need to be addressed by appropriate security techniques. The technique cannot perform reconstruction on connections using IPSec, since they are unable to regenerate appropriate security information. It is possible to explicitly encode IPSec security information from suppressed packets, allowing operation with IPSec AH, however this remains an open research issue, and implies an additional overhead per ACK.

RECOMMENDATION: ACK Compaction and Companding are experimental transparent modifications performed on the reverse path following the upstream bottleneck link. They are designed to be used in conjunction with a modified TYPE 1 mitigation and reduce the burst size of TCP transmission in the forward direction, which may otherwise increase when TYPE 1 schemes are used alone.

The technique is desirable, but requires modification of equipment after the upstream bottleneck link (including processing of a modified ACK header). Selection of appropriate algorithms to pace the ACK traffic also remains an open research issue. Some potential Denial-of-Service vulnerabilities may arise with any device that may act as a packet amplifier. These need to be addressed by appropriate security techniques. There is little experience of using the scheme over Internet paths. This scheme is a subject of ongoing research and is not recommended for use within the Internet in its current form.

5.3.3 Mitigating TCP packet bursts generated by Infrequent ACKs

The bursts of data packets generated when a Type 1 scheme is used on the reverse direction path may be mitigated by introducing a router supporting Generic Traffic Shaping (GTS) on the forward path [Seg00]. GTS is a standard router mechanism implemented in many deployed routers. This technique does not eliminate the bursts of data generated by the TCP sender, but attempts to smooth out the bursts by employing scheduling and queuing techniques, producing traffic which resembles that when TCP Pacing is used (section 4.6). These

techniques require maintaining per-flow soft-state in the router, and increase per-packet processing overhead. Some additional buffer capacity is needed to queue packets being shaped.

To perform GTS, the router needs to select appropriate traffic shaping parameters, which require knowledge of the network policy, connection behavior and/or downstream bottleneck characteristics. GTS may also be used to enforce other network policies and promote fairness between competing TCP connections (and also UDP and multicast flows). It also reduces the probability of ACK Compression [ZSC91].

The smoothing of packet bursts reduces the impact of the TCP transmission bursts on routers and hosts following the point at which GTS is performed. It is therefore desirable to perform GTS near to the sending host, or at least at a point before the first forward path bottleneck router.

RECOMMENDATIONS: Generic Traffic Shaping (GTS) is a transparent technique employed at a router on the forward path. The algorithms to implement GTS are available in widely deployed routers and may be used on an Internet link, but do imply significant additional per-packet processing cost.

Configuration of a GTS is a policy decision of a network service provider. When appropriately configured the technique will reduce size of TCP data packet bursts, mitigating the effects of Type 1 techniques. GTS is recommended for use in the Internet in conjunction with type 1 techniques such as ACK Filtering (section 5.2.1) and ACK Decimation (section 5.2.2).

5.4 TYPE 3: Upstream Link Scheduling

Many of the above schemes imply using per flow queues (or per connection queues in the case of TCP) at the upstream bottleneck link. Per-flow queuing (e.g., FQ, CBQ) offers benefit when used on any slow link (where the time to transmit a packet forms an appreciable part of the path RTT) [RFC3150]. Type 3 schemes offer additional benefit when used with one of the above techniques.

5.4.1 Per-Flow queuing at the Upstream Bottleneck Link

When bidirectional traffic exists in a bandwidth asymmetric network competing ACK and packet data flows along the return path may degrade the performance of both upstream and downstream flows [KVR98]. Therefore, it is highly desirable to use a queuing strategy combined with a scheduling mechanism at the upstream link. This has also been called priority-based multiplexing [RFC3135].

On a slow upstream link, appreciable jitter may be introduced by sending large data packets ahead of ACKs [RFC3150]. A simple scheme may be implemented using per-flow queuing with a fair scheduler (e.g., round robin service to all flows, or priority scheduling). A modified scheduler [KVR98] could place a limit on the number of ACKs a host is allowed to transmit upstream before transmitting a data packet (assuming at least one data packet is waiting in the upstream link queue). This guarantees at least a certain minimum share of the capacity to flows in the reverse direction, while enabling flows in the forward direction to improve TCP throughput.

Bulk (payload) compression, a small MTU, link level transparent fragmentation [RFC1991, RFC2686] or link level suspend/resume capability (where higher priority frames may pre-empt transmission of lower priority frames) may be used to mitigate the impact (jitter) of bidirectional traffic on low speed links [RFC3150]. More advanced schemes (e.g., WFQ) may also be used to improve the performance of transfers with multiple ACK streams such as http [Seg00].

RECOMMENDATION: Per-flow queuing is a transparent modification performed at the upstream bottleneck link. Per-flow (or per-class) scheduling does not impact the congestion behavior of the Internet, and may be used on any Internet link. The scheme has particular benefits for slow links. It is widely implemented and widely deployed on links operating at less than 2 Mbps. This is recommended as a mitigation on its own or in combination with one of the other described techniques.

5.4.2 ACKs-first Scheduling

ACKs-first Scheduling is an experimental technique to improve performance of bidirectional transfers. In this case data packets and ACKs compete for resources at the upstream bottleneck link [RFC3150]. A single First-In First-Out, FIFO, queue for both data packets and ACKs could impact the performance of forward transfers. For example, if the upstream bottleneck link is a 28.8 kbps dialup line, the transmission of a 1 Kbyte sized data packet would take about 280 ms. So even if just two such data packets get queued ahead of ACKs (not an uncommon occurrence since data packets are sent out in pairs during slow start), they would shut out ACKs for well over half a second. If more than two data packets are queued up ahead of an ACK, the ACKs would be delayed by even more [RFC3150].

A possible approach to alleviating this is to schedule data and ACKs differently from FIFO. One algorithm, in particular, is ACKs-first scheduling, which accords a higher priority to ACKs over data packets. The motivation for such scheduling is that it minimizes the idle time for the forward connection by minimizing the time that ACKs

spend queued behind data packets at the upstream link. At the same time, with Type 0 techniques such as header compression [RFC1144], the transmission time of ACKs becomes small enough that the impact on subsequent data packets is minimal. (Subnetworks in which the per-packet overhead of the upstream link is large, e.g., packet radio subnetworks, are an exception, section 3.2.) This scheduling scheme does not require the upstream bottleneck router/host to explicitly identify or maintain state for individual TCP connections.

ACKs-first scheduling does not help avoid a delay due to a data packet in transmission. Link fragmentation or suspend/resume may be beneficial in this case.

RECOMMENDATION: ACKs-first scheduling is an experimental transparent modification performed at the upstream bottleneck link. If it is used without a mechanism (such as ACK Congestion Control (ACC), section 4.3) to regulate the volume of ACKs, it could lead to starvation of data packets. This is a performance penalty experienced by end hosts using the link and does not modify Internet congestion behavior. Experiments indicate that ACKs-first scheduling in combination with ACC is promising. However, there is little experience of using the technique in the wider Internet. Further development of the technique remains an open research issue, and therefore the scheme is not currently recommended for use within the Internet.

6. Security Considerations

The recommendations contained in this document do not impact the integrity of TCP, introduce new security implications to the TCP protocol, or applications using TCP.

Some security considerations in the context of this document arise from the implications of using IPSec by the end hosts or routers operating along the return path. Use of IPSec prevents, or complicates, some of the mitigations. For example:

(i) When IPSec ESP [RFC2406] is used to encrypt the IP payload, the TCP header can neither be read nor modified by intermediate entities. This rules out header compression, ACK Filtering, ACK Reconstruction, and the ACK Compaction.

(ii) The TCP header information may be visible, when some forms of network layer security are used. For example, using IPSec AH [RFC2402], the TCP header may be read, but not modified, by intermediaries. This may in future allow extensions to support ACK Filtering, but rules out the generation of new

packets by intermediaries (e.g., ACK Reconstruction). The enhanced header compression scheme discussed in [RFC2507] would also work with IPSec AH.

There are potential Denial-of-Service (DoS) implications when using Type 2 schemes. Unless additional security mechanisms are used, a Reconstructor/expander could be exploited as a packet amplifier. A third party may inject unauthorized Stretch ACKs into the reverse path, triggering the generation of additional ACKs. These ACKs would consume capacity on the return path and processing resources at the systems along the path, including the destination host. This provides a potential platform for a DoS attack. The usual precautions must be taken to verify the correct tunnel end point, and to ensure that applications cannot falsely inject packets that expand to generate unwanted traffic. Imposing a rate limit and bound on the delayed ACK factor(d) would also lessen the impact of any undetected exploitation.

7. Summary

This document considers several TCP performance constraints that arise from asymmetry in the properties of the forward and reverse paths across an IP network. Such performance constraints arise, e.g., as a result of both bandwidth (capacity) asymmetry, asymmetric shared media in the reverse direction, and interactions with Media Access Control (MAC) protocols. Asymmetric capacity may cause TCP Acknowledgments (ACKs) to be lost or become inordinately delayed (e.g., when a bottleneck link is shared between many flows, or when there is bidirectional traffic). This effect may be exacerbated with media-access delays (e.g., in certain multi-hop radio subnetworks, satellite Bandwidth on Demand access). Asymmetry, and particular high asymmetry, raises a set of TCP performance issues.

A set of techniques providing performance improvement is surveyed. These include techniques to alleviate ACK Congestion and techniques that enable a TCP sender to cope with infrequent ACKs without destroying TCP self-clocking. These techniques include both end-to-end, local link-layer, and subnetwork schemes. Many of these techniques have been evaluated in detail via analysis, simulation, and/or implementation on asymmetric subnetworks forming part of the Internet. There is however as yet insufficient operational experience for some techniques, and these therefore currently remain items of on-going research and experimentation.

The following table summarizes the current recommendations. Mechanisms are classified as recommended (REC), not recommended (NOT REC) or experimental (EXP). Experimental techniques may not be well specified. These techniques will require further operational experience before they can be recommended for use in the public Internet.

The recommendations for end-to-end host modifications are summarized in table 1. This lists each technique, the section in which each technique is discussed, and where it is applied (S denotes the host sending TCP data packets in the forward direction, R denotes the host which receives these data packets).

Technique	Use	Section	Where
Modified Delayed ACKs	NOT REC	4.1	TCP R
Large MSS & NO FRAG	REC	4.2	TCP SR
Large MSS & IP FRAG	NOT REC	4.2	TCP SR
ACK Congestion Control	EXP	4.3	TCP SR
Window Pred. Mech (WPM)	NOT REC	4.4	TCP R
Window Cwnd. Est. (ACE)	NOT REC	4.5	TCP R
TCP Sender Pacing	EXP *1	4.6	TCP S
Byte Counting	NOT REC *2	4.7	TCP S
Backpressure	EXP *1	4.8	TCP R

Table 1: Recommendations concerning host modifications.

*1 Implementation of the technique may require changes to the internal design of the protocol stack in end hosts.
*2 Dependent on a scheme for preventing excessive TCP transmission burst.

The recommendations for techniques that do not require the TCP sender and receiver to be aware of their existence (i.e., transparent techniques) are summarized in table 2. Each technique is listed along with the section in which each mechanism is discussed, and where the technique is applied (S denotes the sending interface prior to the upstream bottleneck link, R denotes receiving interface following the upstream bottleneck link).

Mechanism	Use	Section	Type
Header Compr. (V-J)	REC *1	5.1.1	0 SR
Header Compr. (ROHC)	REC *1 *2	5.1.2	0 SR
ACK Filtering (AF)	EXP *3	5.2.1	1 S
ACK Decimation	EXP *3	5.2.2	1 S
ACK Reconstruction (AR)	NOT REC	5.3.1	2 *4
ACK Compaction/Compand.	EXP	5.3.2	2 S *4
Gen. Traff. Shap. (GTS)	REC	5.3.3	2 *5
Fair Queueing (FQ)	REC	5.4.1	3 S
ACKs-First Scheduling	NOT REC	5.4.2	3 S

Table 2: Recommendations concerning transparent modifications.

*1 At high asymmetry these schemes may degrade TCP performance, but are not considered harmful to the Internet.
*2 Standardisation of new TCP compression protocols is the subject of ongoing work within the ROHC WG, refer to other IETF RFCs on the use of these techniques.
*3 Use in the Internet is dependent on a scheme for preventing excessive TCP transmission burst.
*4 Performed at a point along the reverse path after the upstream bottleneck link.
*5 Performed at a point along the forward path.

8. Acknowledgments

This document has benefited from comments from the members of the Performance Implications of Links (PILC) Working Group. In particular, the authors would like to thank John Border, Spencer Dawkins, Aaron Falk, Dan Grossman, Randy Katz, Jeff Mandin, Rod Ragland, Ramon Segura, Joe Touch, and Lloyd Wood for their useful comments. They also acknowledge the data provided by Metricom Inc., concerning operation of their packet data network.

9. References

References of the form RFCnnnn are Internet Request for Comments (RFC) documents available online at http://www.rfc-editor.org/.

9.1 Normative References

[RFC793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.

[RFC1122] Braden, R., Ed., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, October 1989.

[RFC1144] Jacobson, V., "Compressing TCP/IP Headers for Low-Speed Serial Links", RFC 1144, February 1990.

[RFC1191] Mogul, J. and S. Deering, "Path MTU Discovery", RFC 1191, November 1990.

[RFC2581] Allman, M., Paxson, V. and W. Stevens, "TCP Congestion Control", RFC 2581, April 1999.

[RFC2784] Farinacci, D., Li, T., Hanks, S., Meyer, D. and P. Traina, "Generic Routing Encapsulation (GRE)", RFC 2784, March 2000.

[RFC3135] Border, J., Kojo, M., Griner, J., Montenegro, G. and Z. Shelby, "Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations", RFC 3135, June 2001.

9.2 Informative References

[abc-ID] Allman, M., "TCP Congestion Control with Appropriate Byte Counting", Work in Progress.

[All97b] Allman, M., "Fixing Two BSD TCP Bugs", Technical Report CR-204151, NASA Lewis Research Center, October 1997.

[ANS01] ANSI Standard T1.413, "Network to Customer Installation Interfaces - Asymmetric Digital Subscriber Lines (ADSL) Metallic Interface", November 1998.

[ASB96] Arora, V., Suphasindhu, N., Baras, J.S. and D. Dillon, "Asymmetric Internet Access over Satellite-Terrestrial Networks", Proc. AIAA: 16th International Communications Satellite Systems Conference and Exhibit, Part 1, Washington, D.C., February 25-29, 1996, pp.476-482.

[AST00] Aggarwal, A., Savage, S., and T. Anderson, "Understanding the Performance of TCP Pacing", Proc. IEEE INFOCOM, Tel-Aviv, Israel, V.3, March 2000, pp. 1157-1165.

[Bal98] Balakrishnan, H., "Challenges to Reliable Data Transport over Heterogeneous Wireless Networks", Ph.D. Thesis, University of California at Berkeley, USA, August 1998. http://nms.lcs.mit.edu/papers/hari-phd/

[BPK99] Balakrishnan, H., Padmanabhan, V. N., and R. H. Katz, "The Effects of Asymmetry on TCP Performance", ACM Mobile Networks and Applications (MONET), Vol.4, No.3, 1999, pp. 219-241. An expanded version of a paper published at Proc. ACM/IEEE Mobile Communications Conference (MOBICOM), 1997.

[BPS00] Bennett, J. C., Partridge, C., and N. Schectman, "Packet Reordering is Not Pathological Network Behaviour", IEEE/ACM Transactions on Networking, Vol. 7, Issue. 6, 2000, pp.789-798.

[Cla88] Clark, D.D, "The Design Philosophy of the DARPA Internet Protocols", ACM Computer Communications Review (CCR), Vol. 18, Issue 4, 1988, pp.106-114.

[CLC99] Clausen, H., Linder, H., and B. Collini-Nocker, "Internet over Broadcast Satellites", IEEE Communications Magazine, Vol. 37, Issue. 6, 1999, pp.146-151.

[CLP98] Calveras, A., Linares, J., and J. Paradells, "Window Prediction Mechanism for Improving TCP in Wireless Asymmetric Links". Proc. IEEE Global Communications Conference (GLOBECOM), Sydney Australia, November 1998, pp.533-538.

[CR98] Cohen, R., and Ramanathan, S., "Tuning TCP for High Performance in Hybrid Fiber Coaxial Broad-Band Access Networks", IEEE/ACM Transactions on Networking, Vol.6, No.1, 1998, pp.15-29.

[DS00] Cable Television Laboratories, Inc., Data-Over-Cable Service Interface Specifications---Radio Frequency Interface Specification SP-RFIv1.1-I04-00407, 2000

[DS01] Data-Over-Cable Service Interface Specifications, Radio Frequency Interface Specification 1.0, SP-RFI-I05-991105, Cable Television Laboratories, Inc., November 1999.

[DMT96] Durst, R., Miller, G., and E. Travis, "TCP Extensions for Space Communications", ACM/IEEE Mobile Communications Conference (MOBICOM), New York, USA, November 1996, pp.15-26.

[EN97] "Digital Video Broadcasting (DVB); DVB Specification for Data Broadcasting", European Standard (Telecommunications series) EN 301 192, 1997.

[EN00] "Digital Video Broadcasting (DVB); Interaction Channel for Satellite Distribution Systems", Draft European Standard (Telecommunications series) ETSI, Draft EN 301 790, v.1.2.1

[FJ93] Floyd, S., and V. Jacobson, "Random Early Detection gateways for Congestion Avoidance", IEEE/ACM Transactions on Networking, Vol.1, No.4, 1993, pp.397-413.

[FSS01] Fairhurst, G., Samaraweera, N.K.G, Sooriyabandara, M., Harun, H., Hodson, K., and R. Donardio, "Performance Issues in Asymmetric Service Provision using Broadband Satellite", IEE Proceedings on Communication, Vol.148, No.2, 2001, pp.95-99.

[ITU01] ITU-T Recommendation E.681, "Traffic Engineering Methods For IP Access Networks Based on Hybrid Fiber/Coax System", September 2001.

[ITU02] ITU-T Recommendation G.992.1, "Asymmetrical Digital Subscriber Line (ADSL) Transceivers", July 1999.

[Jac88] Jacobson, V., "Congestion Avoidance and Control", Proc. ACM SIGCOMM, Stanford, CA, ACM Computer Communications Review (CCR), Vol.18, No.4, 1988, pp.314-329.

[Ken87] Kent C.A., and J. C. Mogul, "Fragmentation Considered Harmful", Proc. ACM SIGCOMM, USA, ACM Computer Communications Review (CCR), Vol.17, No.5, 1988, pp.390-401.

[KSG98] Krout, T., Solsman, M., and J. Goldstein, "The Effects of Asymmetric Satellite Networks on Protocols", Proc. IEEE Military Communications Conference (MILCOM), Bradford, MA, USA, Vol.3, 1998, pp.1072-1076.

[KVR98] Kalampoukas, L., Varma, A., and Ramakrishnan, K.K., "Improving TCP Throughput over Two-Way Asymmetric Links: Analysis and Solutions", Proc. ACM SIGMETRICS, Medison, USA, 1998, pp.78-89.

[LM97] Lin, D., and R. Morris, "Dynamics of Random Early Detection", Proc. ACM SIGCOMM, Cannes, France, ACM Computer Communications Review (CCR), Vol.27, No.4, 1997, pp.78-89.

[LMS97] Lakshman, T.V., Madhow, U., and B. Suter, "Window-based Error Recovery and Flow Control with a Slow Acknowledgement Channel: A Study of TCP/IP Performance", Proc. IEEE INFOCOM, Vol.3, Kobe, Japan, 1997, pp.1199-1209.

[MJW00] Ming-Chit, I.T., Jinsong, D., and W. Wang,"Improving TCP Performance Over Asymmetric Networks", ACM SIGCOMM, ACM Computer Communications Review (CCR), Vol.30, No.3, 2000.

[Pad98] Padmanabhan, V.N., "Addressing the Challenges of Web Data Transport", Ph.D. Thesis, University of California at Berkeley, USA, September 1998 (also Tech Report UCB/CSD-98-1016). http://www.cs.berkeley.edu/~padmanab/phd-thesis.html

[RFC1323] Jacobson, V., Braden, R. and D. Borman, "TCP Extensions for High Performance", RFC 1323, May 1992.

[RFC2018] Mathis, B., Mahdavi, J., Floyd, S. and A. Romanow, "TCP Selective Acknowledgment Options", RFC 2018, October 1996.

[RFC2402] Kent, S. and R. Atkinson, "IP Authentication Header", RFC 2402, November 1998.

[RFC2406] Kent, S. and R. Atkinson, "IP Encapsulating Security Payload (ESP)", RFC 2406, November 1998.

[RFC2507] Degermark, M., Nordgren, B. and S. Pink, "IP Header Compression", RFC 2507, February 1999.

[RFC2525] Paxson, V., Allman, M., Dawson, S., Heavens, I. and B. Volz, "Known TCP Implementation Problems", RFC 2525, March 1999.

[RFC2686] Bormann, C., "The Multi-Class Extension to Multi-Link PPP", RFC 2686, September 1999.

[RFC2760] Allman, M., Dawkins, S., Glover, D., Griner, J., Henderson, T., Heidemann, J., Kruse, H., Ostermann, S., Scott, K., Semke, J., Touch, J. and D. Tran, "Ongoing TCP Research Related to Satellites", RFC 2760, February 2000.

[RFC2988] Paxson, V. and M. Allman, "Computing TCP's Retransmission Timer", RFC 2988, November 2000.

[RFC3077] Duros, E., Dabbous, W., Izumiyama, H., Fujii, N. and Y. Zhang, "A link Layer tunneling mechanism for unidirectional links", RFC 3077, March 2001.

[RFC3095] Bormann, C., Burmeister, C., Degermark, M., Fukushima, H., Hannu, H., Jonsson, E., Hakenberg, R., Koren, T., Le, K., Liu, Z., Martensson, A., Miyazaki, A., Svanbro, K., Wiebke, T., Yoshimura, T. and H. Zheng, "RObust Header Compression (ROHC): Framework and four profiles: RTP, UDP ESP and uncompressed", RFC 3095, July 2001.

[RFC3150] Dawkins, S., Montenegro, G., Kojo, M. and V. Magret, "End-to-end Performance Implications of Slow Links", BCP 48, RFC 3150, July 2001.

[RFC3168] Ramakrishnan K., Floyd, S. and D. Black, "A Proposal to add Explicit Congestion Notification (ECN) to IP", RFC 3168, September 2001.

[Sam99] Samaraweera, N.K.G, "Return Link Optimization for Internet Service Provision Using DVB-S Networks", ACM Computer Communications Review (CCR), Vol.29, No.3, 1999, pp.4-19.

[Seg00] Segura R., "Asymmetric Networking Techniques For Hybrid Satellite Communications", NC3A, The Hague, Netherlands, NATO Technical Note 810, August 2000, pp.32-37.

[SF98] Samaraweera, N.K.G., and G. Fairhurst. "High Speed Internet Access using Satellite-based DVB Networks", Proc. IEEE International Networks Conference (INC98), Plymouth, UK, 1998, pp.23-28.

[ZSC91] Zhang, L., Shenker, S., and D. D. Clark, "Observations and Dynamics of a Congestion Control Algorithm: The Effects of Two-Way Traffic", Proc. ACM SIGCOMM, ACM Computer Communications Review (CCR), Vol 21, No 4, 1991, pp.133-147.

10. IANA Considerations

There are no IANA considerations associated with this document.

Appendix - Examples of Subnetworks Exhibiting Network Path Asymmetry

This appendix provides a list of some subnetworks which are known to experience network path asymmetry. The asymmetry in capacity of these network paths can require mitigations to provide acceptable overall performance. Examples include the following:

- IP service over some wide area and local area wireless networks. In such networks, the predominant network path asymmetry arises from the hub-and-spokes architecture of the network (e.g., a single base station that communicates with multiple mobile stations), this requires a Ready To Send / Clear To Send (RTS/CTS) protocol and a Medium Access Control (MAC) protocol which needs to accommodate the significant turn-around time for the radios. A high per-packet transmission overhead may lead to significant network path asymmetry.

- IP service over a forward satellite link utilizing Digital Video Broadcast (DVB) transmission [EN97] (e.g., 38-45 Mbps), and a slower upstream link using terrestrial network technology (e.g., dial-up modem, line of sight microwave, cellular radio) [CLC99]. Network path asymmetry arises from a difference in the upstream and downstream link capacities.

- Certain military networks [KSG98] providing Internet access to in-transit or isolated hosts [Seg00] using a high capacity downstream satellite link (e.g., 2-3 Mbps) with a narrowband upstream link (e.g., 2.4-9.6 kbps) using either Demand Assigned Multiple Access (DAMA) or fixed rate satellite links. The main factor contributing to network path asymmetry is the difference in the upstream and downstream link capacities. Some differences between forward and reverse paths may arise from the way in which upstream link capacity is allocated.

- Most data over cable TV networks (e.g., DOCSIS [ITU01, DS00]), where the analogue channels assigned for upstream communication (i.e., in the reverse direction) are narrower and may be more noisy than those assigned for the downstream link. As a consequence, the upstream and downstream links differ in their transmission rate. For example, in DOCSIS 1.0 [DS00], the downstream transmission rate is either 27 or 52 Mbps. Upstream transmission rates may be dynamically selected to be one of a series of rates which range between 166 kbps to 9 Mbps. Operators may assign multiple upstream channels per downstream channel. Physical layer (PHY) overhead (which accompanies upstream transmissions, but is not present in the downstream link) can also increase the network path asymmetry. The Best Effort service, which is typically used to carry TCP, uses a

contention/reservation MAC protocol. A cable modem (CM) sending an isolated packet (such as a TCP ACK) on the upstream link must contend with other CMs to request capacity from the central cable modem termination system (CMTS). The CMTS then grants timeslots to a CM for the upstream transmission. The CM may "piggyback" subsequent requests onto upstream packets, avoiding contention cycles; as a result, spacing of TCP ACKs can be dramatically altered due to minor variations in load of the cable data network and inter-arrival times of TCP DATA packets. Numerous other complexities may add to, or mitigate, the asymmetry in rate and access latency experienced by packets sent on the upstream link relative to downstream packets in DOCSIS. The asymmetry experienced by end hosts may also change dynamically (e.g., with network load), and when best effort services share capacity with services that have symmetric reserved capacity (e.g., IP telephony over the Unsolicited Grant service) [ITU01].

- Asymmetric Digital Subscriber Line (ADSL), by definition, offers a downstream link transmission rate that is higher than that of the upstream link. The available rates depend upon channel quality and system configuration. For example, one widely deployed ADSL technology [ITU02, ANS01] operates at rates that are multiples of 32 kbps (up to 6.144 Mbps) in the downstream link, and up to 640 kbps for the upstream link. The network path asymmetry experienced by end hosts may be further increased when best effort services, e.g., Internet access over ADSL, share the available upstream capacity with reserved services (e.g., constant bit rate voice telephony).

Authors' Addresses

Hari Balakrishnan
Laboratory for Computer Science
200 Technology Square
Massachusetts Institute of Technology
Cambridge, MA 02139
USA

Phone: +1-617-253-8713
EMail: hari@lcs.mit.edu
Web: http://nms.lcs.mit.edu/~hari/

Venkata N. Padmanabhan
Microsoft Research
One Microsoft Way
Redmond, WA 98052
USA

Phone: +1-425-705-2790
EMail: padmanab@microsoft.com
Web: http://www.research.microsoft.com/~padmanab/

Godred Fairhurst
Department of Engineering
Fraser Noble Building
University of Aberdeen
Aberdeen AB24 3UE
UK

EMail: gorry@erg.abdn.ac.uk
Web: http://www.erg.abdn.ac.uk/users/gorry

Mahesh Sooriyabandara
Department of Engineering
Fraser Noble Building
University of Aberdeen
Aberdeen AB24 3UE
UK

EMail: mahesh@erg.abdn.ac.uk
Web: http://www.erg.abdn.ac.uk/users/mahesh

Full Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.