TCP/IP Enhancements for Satellite Networks

Nasir Ghani and Sudhir Dixit, Nokia Research Center

ABSTRACT With the emerging market for high-mobility remote access broadband services, satellite networks are becoming increasingly popular. Although the ubiquitous TCP/IP protocol is widely used to provide reliable data delivery in terrestrial networks, it faces many challenges in satellite environments. These stem from the inherent features of satellite channels, such as large delays, increased error rates, and bandwidth asymmetry. To address these concerns, a variety of solutions have been proposed. These include direct TCP enhancements to better tune the TCP/IP stack and additional selective acknowledgment mechanisms. Other proposals use advanced interworking to reduce sensitivity to various channel features. Examples include multiple TCP sessions, link-layer interworking, and ACK control schemes. Improving onboard satellite features (buffer management, flow control) can also provide benefits for TCP/IP transport. The objective of this article is to present the various solutions and discuss their possible trade-offs. Overall, there exists a rich set of alternatives to meet the challenges in this important arena.

Recently, the interest in broadband satellite networks has grown rapidly. Advances in transmission technology have led to the availability of low-cost earth terminals with interfaces to standard terrestrial networks. The relative ease of deployment and superior remote access capabilities of satellite networks are envisioned to provide bandwidth on demand to geographically diverse user groups. This is clearly evidenced by the ongoing development of several large-scale space-based networks, such as Teledesic, Galaxy/Spaceway, and Astrolink. Most notably, the Teledesic system aims to provide access rates of up to 2 Mb/s per user and 150 Mb/s per gateway [1]. Overall, these trends represent a significant departure from traditional fixed-circuit broadcast satellite systems (i.e., for telephony and television). The desire to support a diverse range of services in satellite networks implies that many features inherent in broadband networks, such as asynchronous transfer mode (ATM) and Internet Protocol (IP), will also emerge in satellite networks. These include packet switching, buffer management, and call admission control techniques (see [1, 2] for good overviews). Despite these emerging similarities with terrestrial networks, however, satellite networks still pose many additional challenges.

A crucial requirement for any broadband network is the reliable transport of data traffic, namely non-real-time loss-sensitive flows (as opposed to real-time voice/video streams). Examples include e-mail and file transfers, Web messages, and facsimile transmissions. Since reliable transmission is mandatory, the ubiquitous Transmission Control Protocol (TCP)/IP has emerged as the workhorse protocol for most data applications. TCP uses a sliding window mechanism coupled with reactive congestion control techniques to gate source emissions. The protocol modulates a sender's transmission behavior contingent on returning acknowledgment (ACK) packets from the remote receiver's end, and two variants of TCP are in wide use today: Tahoe and Reno. A simple overview of a satellite TCP/IP network is shown in Fig. 1, where land-based networks are connected to satellite modems via interworking units (IWUs), such as the ATM satellite interworking unit (ASIU) [1]. These IWUs (possibly collocated with modems) perform protocol conversion between WAN protocols (e.g., IP, ATM) and satellite link-layer protocols. The latter are necessary due to satellite channel requirements, and most such protocols use time-division multiplexing (TDM) techniques, although frequency-division multiplexing (FDM) techniques and even hybrid schemes are possible [1]. To accommodate dynamic usage patterns, *demand assignment* approaches can be incorporated with various access schemes [2]. Most IWU systems use ATM and broadband integrated services digital network (B-ISDN) interfaces, although Ethernet and IP ports are also available.

However, many studies have shown that satellite channels have a largely adverse impact on TCP performance [1, 3, 4], with the main problems arising from increased bit error rates (BERs) and large propagation delays. Such conditions usually cause excessive TCP timeouts and retransmissions, resulting in large bandwidth inefficiency. In addition, other TCP features pose additional concerns in satellite environments (small window sizes, round-trip timer inaccuracies, startup windows, etc). Despite these challenges, space-based communication still offers many benefits, such as wide coverage areas, broadcast capabilities, and immunity to adverse geographic conditions. Hence, satellites will continue to play an important role in the future, and extending existing data protocols to this domain is of great importance. In light of the above, many researchers have proposed a wide range of solutions to improve TCP performance in satellite networks. These range from link-layer error control schemes and protocols [1, 3, 5] to more elaborate TCP versions [6–9], and advanced interworking [3, 4, 10–13]. Although some solutions are more effective than others, it is likely that modern systems will draw from all the above. This article presents some of these solutions and discusses their implications.

ISSUES AND CHALLENGES

The challenges in satellite TCP arise from the various inherent features of satellite channels and network features. These are briefly outlined.

CHANNEL ERROR RATES

Reduced signal-to-noise ratios (SNRs) are a big problem in satellite transmission, since signal strength falls proportional to the square of the distance. This is also complicated by various random atmospheric/space conditions (e.g., rain attenuation) and effects such as multipath and shadowing (akin to wireless networks). Measurements show that unencoded satellite channels can have BER values around 10^{-6} [3], much lower than on high-speed wireline media. Since TCP is a *loss-sensitive* protocol using packet losses to control transmission behavior, it is unable to discern between congestion losses and link *corruption* losses. Many studies confirm that noisy satellite links can greatly degrade TCP's performance since larger BER values prematurely trigger window reduction mechanisms, even if the network is uncongested [4, 11]. Additional-

 0163-6804/99/$10.00 © 1999 IEEE

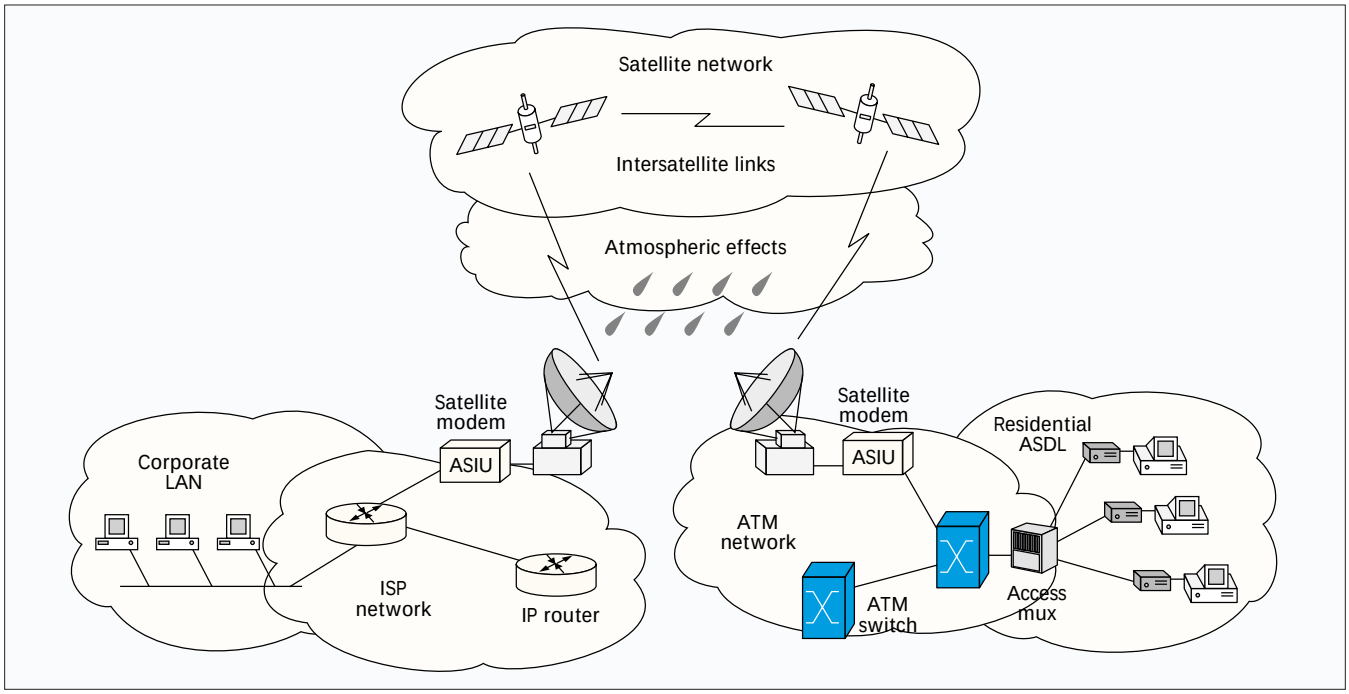

Figure 1. *An overview of a sample satellite network.*

ly, the loss of ACK packets (reverse direction) worsens throughputs further, since the corresponding data packets being acknowledged have already been received. Furthermore, errors on satellite links are usually bursty, which is also very problematic, since the fast recovery/retransmit features of TCP Reno usually cannot recover from excessive packet drops in a window size [4]. As a result, TCP's congestion avoidance mechanism can severely limit window growth. Considering that TCP needs much larger window sizes in satellite environments (order of bandwidth delay products [14]), the deleterious effects of burst errors will be more pronounced. In fact, many results show that TCP Tahoe (i.e., without fast recovery) actually *outperforms* TCP Reno under such conditions [4, 11]. This is because the latter halves its slow start threshold multiple times during burst losses, and if a timeout occurs, resulting *congestion avoidance* window growth is much slower.

Propagation Delays

Satellite network delays are influenced by several factors, the main one being the orbit type. In most low earth orbit (LEO) systems, one-way propagation delays are in the 20–25 ms range. These values increase to 110–150 ms for medium earth orbit (MEO) and go up to 250–280 ms for geostationary earth orbit (GEO) [1]. Nevertheless, since LEO systems use smaller orbits, visibility windows are relatively short, and hence larger constellations with dynamic intersatellite routing are required to provide continuous coverage over large regions (i.e., tens or even hundreds of satellites). All these factors increase end-to-end delay *variability*, the amount of which depends on the number of satellites, orbital dynamics, intersatellite routing algorithms, and so on [14]. For example, in [3] it is stated that LEO delay variability can range from a few milliseconds up to 80 ms when a satellite is near the horizon. Since TCP traffic is relatively less time-sensitive, it is reasonable to expect it to be subjected to dynamic rerouting, adding to delay variability and possibly causing packet reordering. Other issues such as onboard processing overheads and buffering can also add to delay.

In general, such delay conditions are problematic for TCP, reducing its responsiveness to packet drops. This is especially noticeable for connections wanting only to send marginally more than the default startup window size (one segment); that is, users must wait a full round-trip delay before the first ACK packet is received in the *slow-start* phase. Delayed ACK timers at the receiver can also exacerbate this problem, since the first segment cannot trigger the "ACK-every-other" rule [3]. Satellite delays combined with increasing channel speeds (ten of megabits per second or more) also require significant buffering [4, 15]. Increased delay variances can also adversely affect TCP timer mechanisms by injecting noise into the estimates. This variability can result in premature timeouts/retransmissions, and generally yields incorrect window sizes [3], reducing the overall bandwidth efficiency. Simply increasing the TCP timer granularity will not help here since larger values may reduce false timeouts, but bandwidth underutilization will also increase from longer timeouts.

Channel Asymmetry

Many satellite systems have large bandwidth asymmetries between forward and backward data channels. This is mainly due to cost limitations associated with increased transmitter power and antenna size requirements for higher bandwidth transmission. Using slower reverse channels allows more cost-effective receiver designs, and also conserves valuable satellite bandwidth. Considering the largely unidirectional nature of most large TCP transfers (e.g., from Web servers to remote hosts), slower reverse channels can be acceptable to an extent. Moreover, some solutions do away with expensive satellite transmission capabilities at the receiver altogether, simply using *terrestrial* modem dialup lines for the reverse (ACK) channel. In such cases, data and ACK packets follow separate paths in the network, allowing the latter to avoid possible corruption-loss problems (Fig. 2). However, asymmetric configurations can cause significant problems for TCP. For example, it is well known that slower reverse channels can induce deleterious effects such as ACK loss and *compression* [5], since ACK packets can get dropped or queued behind larger data packets. Here throughputs can be greatly reduced, and sample results in [5] indicate exponential decrease for increasing asymmetry. Additionally, large asymmetries between the forward and reverse channel rates can worsen *forward* buffer congestion significantly since line-rate bursts are much larger [3, 16].

ONBOARD CONSIDERATIONS

Payload size and power budgets usually limit the amount of onboard buffering and processing capacity in satellite nodes [2]. Compounding this issue is the fact that satellites usually represent longer-term investments and hence must interoperate with inevitably improving terrestrial technologies (workstations, modems, even application protocols, etc). Therefore, it is very desirable to prevent the satellite node itself from becoming the bottleneck. One way of achieving this is to limit congestion effects to the terrestrial network component, essentially relegating satellite nodes to performing simple "data relay" functions. Another solution, however, is to provide advanced traffic management features onboard satellite nodes. Contrary to expectations, simply providing ample buffering (i.e., order of bandwidth delay products) may not suffice [10]. The reason is that larger buffer sizes can result in queuing delays of the same order as propagation delays and thereby increase round-trip delay variability. This, in turn, can cause premature timeouts and reduce bandwidth utilization. Improved onboard features are considered subsequently.

LINK-LAYER ENHANCEMENTS

As discussed previously, link error rates are a major concern in satellite TCP, and as a result designers have proposed a variety of solutions. However, since these schemes are not particularly TCP-specific, they are only briefly discussed here (see [1, 2] and references therein for full details).

FORWARD ERROR CORRECTION SCHEMES

One means of improving link conditions is to simply increase transmitter power and/or use larger antennas. However, this is a rather expensive remedy and more workable link-level solutions are commonplace. A variety of forward error correction (FEC) encoding schemes have been proposed for data corruption, ranging from well-known convolution codes to concatenated codes [1]. The level of encoding used depends on the desired link BER, the prevailing link/atmospheric conditions, and even traffic type. Many advanced coding schemes also make use of bit interleaving to reduce the effects of burst errors, by spreading out the errors over multiple cells (packets). This can improve the effectiveness of existing error checksums and single-bit error correction mechanisms. Typically, good systems can achieve BER values over the 10^{-7} range, resulting in cell (packet) error rates of 10^{-9} or better. However, increased encoding complexity can slow down satellite modems and reduce bandwidth efficiency, since data redundancy incurs added overheads. Hence, many advanced satellite networks support a range of encoding schemes for different types of traffic and network conditions. For example, reliable transmission of delay-sensitive streams usually requires the highest level of encoding/interleaving, and more delay-insensitive applications can use reduced encoding levels. In addition, many state-of-the-art satellite devices also use data compression techniques to counteract bandwidth inefficiencies caused by encoding [6].

AUTOMATIC REPEAT REQUEST PROTOCOLS

Various link-level *retransmission* schemes, termed *automatic-repeat-request* (ARQ) [1] protocols, have been developed for satellite networks. Senders segment user datagrams into smaller link-layer "mini-packets" and assign sequence numbers to them before transmission. These sequence numbers are acknowledged by receivers and allow the sender side to retransmit any corrupted mini-packets using a variety of approaches, such as *stop-and-wait*, *go-back*-N, or *selective-repeat*. All ARQ schemes add to channel delay and delay variability, and generally are not well suited in high BER environments due to excessive retransmission delays. Furthermore, the more rudimentary link-layer protocols (go-back-*N*, stop-and-wait) will yield even longer average link delays and can give reduced TCP throughputs when compared to selective repeat schemes. Although selective repeat ARQ schemes are most efficient, more processing complexity is necessary in the modems. The detailed interaction between TCP and such ARQ protocols needs further study. In particular, ARQ retransmission timer granularities need to be

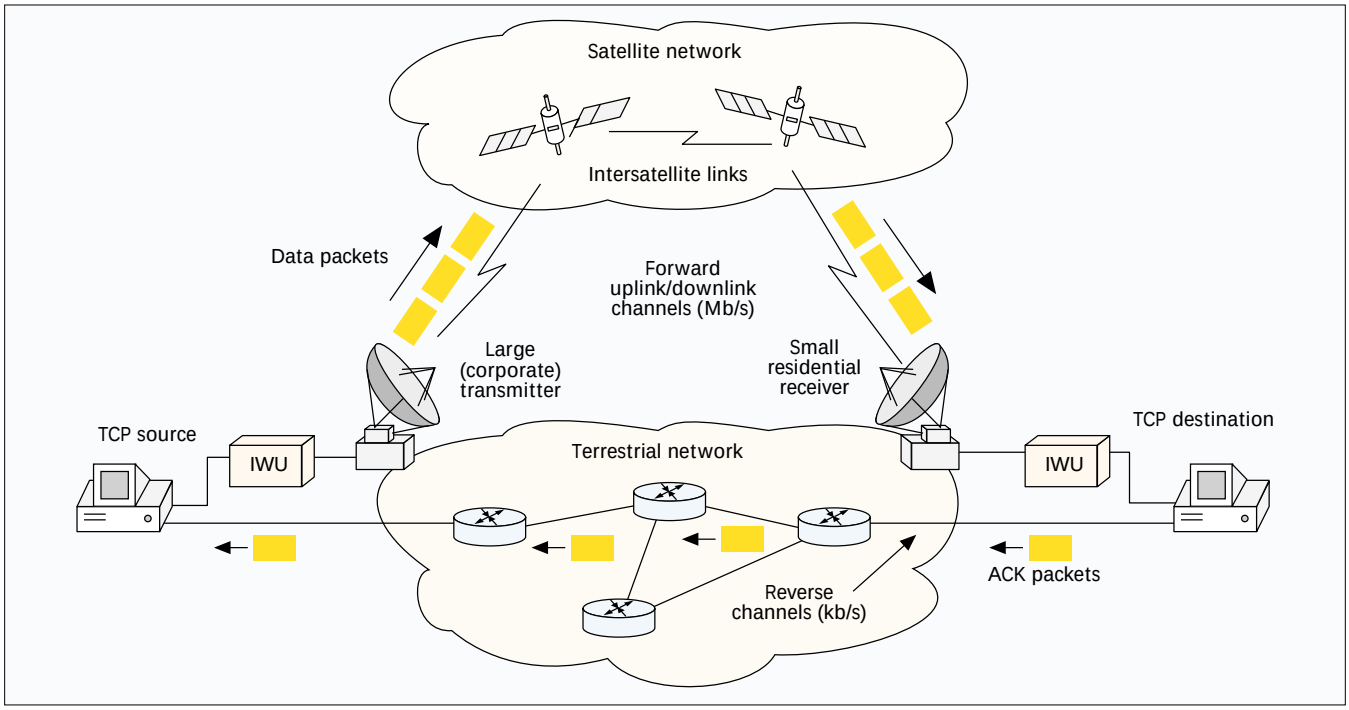

Figure 2. *Asymmetric reverse ACK channel configuration.*

carefully tuned to avoid destructive interactions with TCP's own timeout mechanism (e.g., multiple retransmissions).

IMPROVED TCP VERSIONS

Recognizing the increasing importance of satellite TCP, the Internet Engineering Task Force's (IETF's) TCP over Satellite (tcpsat) working group has addressed many of the issues in detail. The resulting proposals range from moderate TCP alterations to more comprehensive data recovery techniques (see [3, 6] and IETF RFC 1072 for comprehensive surveys). Moreover, the design of newer, improved TCP versions has also been widely studied in the context of high-speed broadband and wireless networks. Many findings from these areas can also be applied to satellite networks. Some of the proposals are now reviewed.

BASIC TCP ENHANCEMENTS

One problem with TCP is that its default window size representation is limited to 16 bits, and this value is insufficient for satellite bandwidth-delay products. Since required window sizes can easily exceed the maximum allowable 65,536 bytes, this limits maximum goodput to roughly 1 Mb/s, below T1 speeds. Simply assigning more bits for the TCP window size is not feasible since associated changes to the header pose many interworking complications with older versions. The *window scaling* option (IETF RFC 1072) addresses this issue, allowing connections to negotiate a scaling factor at startup. This factor is normally a power of 2 and allows for window sizes up to 32 bits in length, more than adequate for satellite networks. However, increased window sizes can also cause sequence number *wraparound* problems and require added *protection against wraparound sequence numbers* (PAWS) mechanisms. Both these features are essential for satellite TCP, especially in GEO systems.

Large round-trip delay variability can yield inaccurate round-trip time estimates, which will inevitably reduce the efficiency of TCP's loss detection mechanism, possibly leading to congestion collapse. One shortcoming is that TCP's timer mechanism only clocks a single segment at a time, resulting in too coarse a sampling rate for dynamic conditions and large window environments, especially if buffering delays are of the same order as propagation delays. The proposed TCP *echo* option, which is signaled at startup (i.e., SYN handshake), solves this problem by associating a sender-side timestamp with each segment. Receivers echo these timestamps, and provisions are given for handling delayed ACK timers and noncontiguous sequence numbers (i.e., dropped segments); see IETF RFC 1072. The echo option is crucial for TCP satellite networks considering the larger delay variability and increased buffering requirements.

Larger TCP segment sizes will help open up the slow-start congestion window faster in long-delay environments. To this end, the well-known *path maximum transmission unit (MTU) discovery* approach can be used, essentially probing the network for the maximum IP (TCP) packet size which can be supported along the *end-to-end* path. Although the delays incurred during the probing stage can be on the order of multiple round-trip times, MTU discovery can yield good benefits if maximum segment sizes are not known a priori. Furthermore, storing the results of MTU discovery can reduce latencies for future connections (in relatively static topologies). Although some have argued against MTU discovery, stating that larger packets are more prone to corruption losses, improving FEC schemes and ARQ protocols are lessening this concern.

Since the slow start phase relies on returning ACK packets to increase window sizes, there is a direct dependence between round-trip times and bandwidth efficiency. To address this issue, researchers have proposed increasing the default window size, allowing more packets to be sent at startup, thereby triggering more ACKs and hence faster growth. In addition, a startup window size of two or more segments will also eliminate any additional delays caused by the delayed ACK timer. The exact choice of the initial window size is an open issue. Overly large values can be too aggressive and actually worsen packet drop rates. However, studies have shown that an initial value of four segments improves startup times significantly, without a corresponding (noticeable) increase in the packet loss (see [3] and references therein); an analytical expression for startup window sizes is also given. Results show savings of up to three round-trip times plus the ACK delay timer interval. Additionally, the "extra" round-trip delay incurred by the three-way TCP handshake can also be avoided by allowing senders to transmit data with the first (SYN) segment (IETF RFC 1644). Satellite TCP can benefit significantly from such modifications.

Continuing the above reasoning, increasing the slow-start threshold from its default value of 65,536 bytes is also recommended for long-delay high-bandwidth paths. This will prevent overly conservative congestion avoidance window growth and improve ramp-up times. Ideally, this threshold should be tuned contingent on the bandwidth-delay product and amount of reverse-channel asymmetry ([16] presents some analytical results). Increased throughput can also be achieved by modifying the ACK generation/window increase procedures. Acknowledging every segment (versus every other segment) or using *byte-counting* techniques at the sender can expedite window growth. The latter technique increments the window based on the number of newly acknowledged bytes, independent of the receiver's ACK generation rate. However, increased burstiness can result here, and further considerations may be necessary to prevent large line-rate bursts (i.e., *limited byte counting* [17]).

Other modifications to TCP have also been proposed in the more general literature to improve performance in large bandwidth-delay networks [4]. A notable area is the design of newer, improved congestion control/avoidance algorithms, such as Vegas [5] and NewReno [4]. The former makes significant changes to the linear increase congestion avoidance phase, allowing senders to increase/decrease/maintain window sizes. TCP Vegas uses delay-sensing techniques to control window increases, and a new retransmission timer mechanism is used to reduce the inaccuracies of coarse timer granularities. Although the details are too lengthy to consider here, results in terrestrial networks indicate over 30 percent improvement in throughput and much fewer losses than TCP Reno. However, TCP Vegas can require substantial tuning in satellite networks [5]. Meanwhile, TCP NewReno proposes an improved fast recovery phase to handle multiple losses, entailing moderate sender-side changes. Specifically, a more aggressive retransmission policy is used to maintain both forward transfers and returning ACK streams (i.e., "self-clocking").

SELECTIVE ACKNOWLEDGE ENHANCEMENTS

A major shortcoming of TCP (Tahoe/Reno) continues to be its vulnerability to multiple losses, which can cause it to lose its "self-clocking" property and timeout [7]. Ideally, it is desirable to prevent unnecessary window reductions and only retransmit corrupted/dropped packets. In long-delay networks, such functionality can help improve bandwidth efficiency, as is the case with the International Telecommunication Union — Telecommunication Standardization Sector (ITU-T) SSCOP protocol for satellite ATM [1]. Along these lines, a significant enhancement to the TCP protocol has been developed, termed *selective acknowledgment* (i.e., TCP SACK, IETF RFC 2018). TCP SACK is a *data recovery* algorithm [7], where receivers can selectively indicate which blocks (segments)

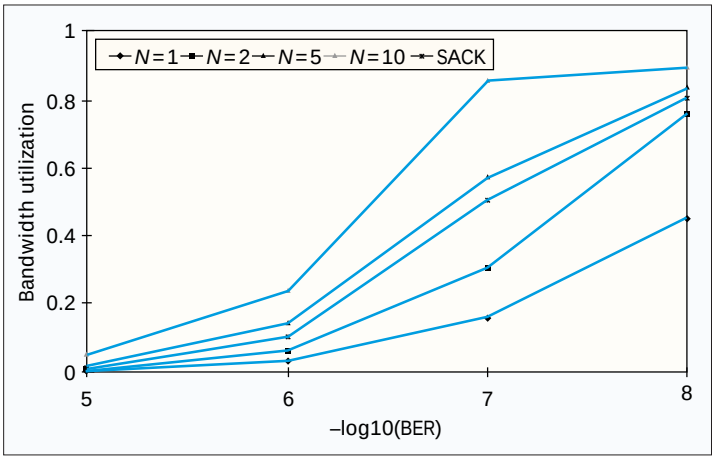

Figure 3. *Bandwidth utilization with multiple TCP sessions (bulk transfer).*

were not received. This allows senders to explicitly retransmit only these omissions and can significantly reduce unnecessary retransmissions. Furthermore, careful consideration has been made to also allow incorporation of existing congestion control features, such as slow start, congestion avoidance, and fast recovery/retransmit (i.e., Reno + SACK [4, 7]). Nevertheless, excessive packet losses can cause SACK to run out of ACKs, and again, the basic timeout mechanism must be deployed here. SACK uses TCP headers to carry information on the missing blocks, although the current size only has space for indicating three regions.

Results show that TCP SACK performs very well in long-delay environments with moderate losses (under 50 percent of the window size [7]), retransmitting all lost segments within the first round-trip time after fast recovery is triggered. Some comprehensive studies state that the SACK feature significantly reduces timeouts and is not overly aggressive when competing with non-SACK versions [8, 9]. Average throughput improvements are measured at anywhere from 15 to 50 percent over various terrestrial Internet settings. The buffering requirements for maximum bandwidth utilization are also unchanged from regular TCP [4, 15], on the order of bandwidth delay products. Nevertheless, despite these improvements, under extreme link error rates (about 10^{-6}), even TCP SACK is unable to prevent excessive timeouts, and average TCP throughputs are below 15 percent [4]. The SACK option requires significant modifications to both the sender and receiver protocol stacks.

The *forward* ACK (FACK) [3, 7] proposal is yet another improvement using the information provided by the SACK option. The intent is to more clearly decouple the congestion control (i.e., when and how much to send) and data recovery (i.e., what to send) algorithms [7]. FACK introduces several new variables to the implementation to more accurately track the amount of outstanding data in the network and maintain TCP self-clocking with multiple losses. By using this information to control emissions, FACK is shown to be less bursty than TCP SACK and more robust against heavy losses [3, 7]. Additional results here show that both SACK and FACK also provide good improvements for networks with reverse-direction congestion (i.e., ACK compression effects). Although more investigation is necessary for noisy satellite links, FACK is expected to provide good performance gains.

Asymmetry Considerations

An effective solution for asymmetric channel problems is ensuring adequate reverse direction bandwidth and using sufficiently large packet sizes [16]. Otherwise, increased *forward* buffering is required to handle larger line-rate bursts. However, more elaborate ACK handling schemes that push out "older" ACK packets (i.e., *drop-from-front strategies* [16]) or even manipulate ACK sequence numbers (i.e., ACK *filtering* schemes [3]) can also be of help here. These schemes tend to keep and return ACK packets that contain higher sequence numbers, improving overall goodputs. Reducing the receiver's ACK generation rate has also been suggested, namely an "ACK-every-*k*" strategy [5] where $k > 2$. Although the above techniques can reduce the loading on a slower ACK channel, at the same time they can significantly lower startup window growth. This problem can be remedied using byte-counting mechanisms (as described previously), although increased TCP burstiness can result, requiring further mechanisms to limit line-rate bursts. A more difficult problem with lower ACK generation rates, however, is the reduced effectiveness of fast recovery (and even SACK) algorithms, since fewer ACKs are generated. The use of selective negative acknowledgements (SNACKs) has been proposed as a remedy, with a modified TCP source simply retransmitting NACK blocks (i.e., not relying on the fast-retransmit mechanism to detect losses). Results here show improved linear throughput degradation for increasing asymmetry [5]. Additionally, since many fields in the TCP and IP headers remain constant over the life of a flow, it is not necessary to repeatedly include them in all transmissions. Well-known *header compression* techniques can reduce the volume of data on bandwidth-constrained (reverse) links (see [3, 5] for an indepth treatment).

Intelligent Interworking

Network designers have also looked at intelligent TCP interworking to improve end-user performance (so-called performance enhancing proxies, PEPs). Although there are many proposals, only some solutions with application in satellite networks are discussed. Many of these schemes are not mutually exclusive, and thus a rich set of possibilities exists for developing further improvements.

Multiple TCP Sessions

One method of improving *bulk* transfers over satellite paths is to instantiate several parallel TCP sessions and stream data over them. Results show significant performance improvements in noisy satellite environments compared to a single connection (utilization, transfer times, etc). In addition, connection ramp-up times are greatly reduced by essentially increasing the initial window size to *N* segments, where *N* is the number of TCP connections. However, this approach results in an aggregate "mega-connection" which is less responsive to packet losses than is a single TCP connection [18] (i.e., "non-TCP-like" behavior). This is not difficult to see, since a corrupted packet only causes one TCP subconnection to reduce its window. Therefore, some have argued that multiple TCP sessions can possibly lead to congestion collapse in public networks [3]. Nevertheless, in private satellite networks where link errors are more the concern, violating "TCP-friendly" behaviors may be more tolerable. It should also be noted that throughput fairness levels between different *bulk* (i.e., application-level) transfers, each using multiple TCP connections, may be more difficult to control since all the subconnections will not see equal loss rates. Several authors have described the effect of multiple TCP sessions as inducing "SACK-like" behavior on the aggregate transfer; moreover, results show that performance improvements are actually better than those with a single TCP SACK session [11]. This is in part due to the aforementioned aggressive

source policy. Furthermore, since multiple sessions imply smaller operating window sizes per session, burst losses are less likely to only affect a given flow. This "spreads out" the burst losses over multiple TCP connections, improving the effectiveness of TCP Reno's fast recovery/retransmit algorithm.

To reveal these performance benefits more clearly, sample simulation results with this approach are presented for various BERs on an MEO link (Fig. 1). The one-way forward propagation delay is set to 120 ms, and the reverse channel is restricted to 200 kb/s. The link rate is fixed at 10 Mb/s (ignoring FEC overheads), and no ARQ link protocols are used. Multiple TCP Reno sessions are used along with a single TCP SACK connection. The window scaling/PAWS options are enabled and the TCP segment size is fixed at 9140 bytes (ATM value). Assuming limited onboard storage, most of the buffering is placed at the access devices, roughly 300 kbytes, close to the round-trip bandwidth-delay product. The average bandwidth utilization results for bulk TCP transfers are shown in Fig. 3, measured over 30 s runs. A clear improvement is seen with increasing N for BER values at or above 10^{-7}. As expected, the results confirm that more sessions are required for higher BER values. Also, five plain TCP connections outperform the TCP SACK version in all cases (Fig. 3). However, lower BER values near 10^{-6} still yield bandwidth utilization below 25 percent, confirming the results in [11]. In fact, even with 10 connections, only several *congestion* drops occurred for BER values of 10^{-7}. This indicates that high BER rates can actually *reduce* buffering requirements.

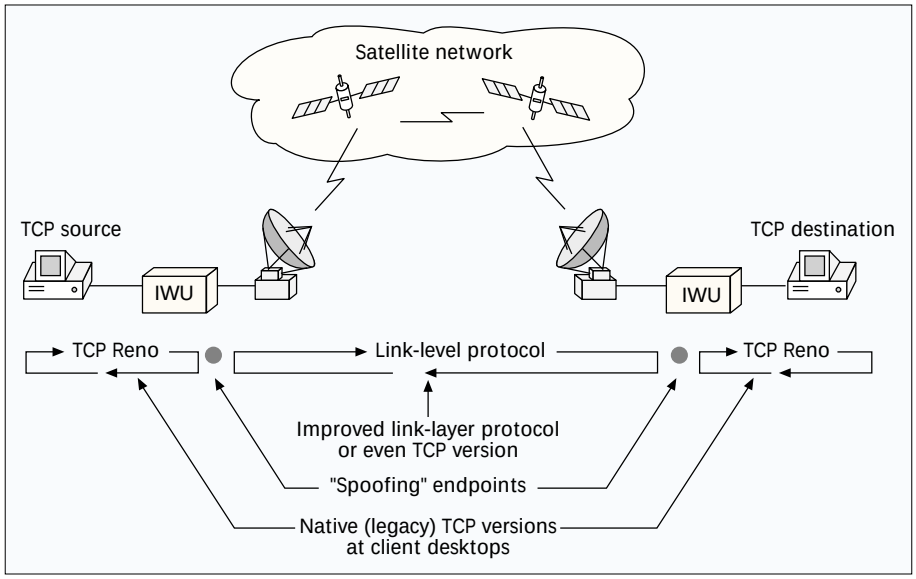

Figure 4. *TCP/IP satellite link spoofing configuration.*

In general, multiple connections impose additional complexity in the protocol stacks, since user files must be proportioned between TCP streams (i.e., *file striping* [11]). Careful considerations are necessary here, since all TCP connections will rarely exhibit similar throughputs. Simply using fixed partitioning between connections can result in one connection slowing down the aggregate transfer process. Therefore, it is much more effective to divide the file into smaller blocks and dynamically assign these blocks to connections which are free to transmit them. This entails further complexity; moreover, the choice of how many TCP connections to use (for a given file size, link condition, MSS value, etc.) needs further study. Along these lines, it is felt that multiple TCP SACK sessions will yield even better results. In addition, the aforementioned "aggressive" mega-source behavior can be remedied by modifying the window growth to use shared congestion information between the parallel TCP sessions, thereby achieving more "TCP-like" transfers [18].

LINK-LAYER INTERWORKING

Various "TCP-aware" interworkings can also be used for lossy satellite links to help discern more clearly between congestion and corruption loss. Within the context of *terrestrial* wireless TCP, such schemes have shown good performance improvements by incorporating corruption loss information into the feedback. These include using link error feedback schemes, such as explicit loss notification (ELN) in ACK packets, ACK snooping, and multiple/partial ACKs [3, 5 and references therein]. Similarly, researchers have also integrated link-layer corruption indications more directly into modified satellite TCP stacks. For example, the Space Communications Protocol Standards-Transport Protocol (SCPS-TP) [3] changes TCP's default window-reduction property upon loss detection by using a broader loss definition, namely congestion, corruption, or even connectivity. Remote base stations echo *congestion-experienced* or *link-outage* ICMP messages, in addition to regular duplicate ACK packets, and modified TCP sources use this information to avoid unnecessary window reduction. This gives very good improvements for higher BER channels [5], although some may argue that it violates the layering approach.

Spoofing refers to techniques which "split" TCP connections between the (end) source and destination clients into two or more parts [3]. Essentially, this breaks the end-to-end semantics of TCP by creating multiple connections terminated at the designated (access) nodes. For example, Fig. 4 shows a setup where TCP Reno clients are "spoofed" at satellite access nodes using more advanced link-layer corruption-loss handling schemes. These can include link-layer ARQ protocols or even specialized TCP versions (SACK, FACK, etc.) on the satellite portion of the path, providing end users with improved throughputs *without* costly upgrades to their simpler TCP stacks (i.e., hiding link losses). (Note that although Fig. 4 shows the use of improved TCP versions as "link-layer" solutions, they were originally intended for *end-to-end* improvements). In private networks where many design features are under the operator's control, TCP spoofing can yield good improvements. For example, a priori knowledge of the link conditions and bandwidth/buffering capacities can allow for full line-rate emissions by the sender side, avoiding sluggish TCP slow start ramp-ups altogether. In addition, by segmenting the end-to-end TCP session into multiple smaller ones, improved responsiveness can be attained, akin to the virtual source/virtual destination (VS/VD) properties of the ATM available bit rate (ABR) service category. Quantitative studies of this effect and TCP spoofing in general are worth pursuing further. Nevertheless, TCP spoofing imposes significant *per-flow* complexity and added buffering overheads at the access nodes, since they must act on behalf of the end clients. In particular, control-plane signaling issues arise, since the access nodes must "trap" TCP connection start sequences (SYN-ACK handshakes) and then spawn intermediate connections. The associated overheads here can be notable, especially if the flows are short-lived. In addition, the interaction between different TCP versions on the *end-to-end* transfer process is not well understood and needs further study. Future aggregation techniques which merge several client flows onto fewer,

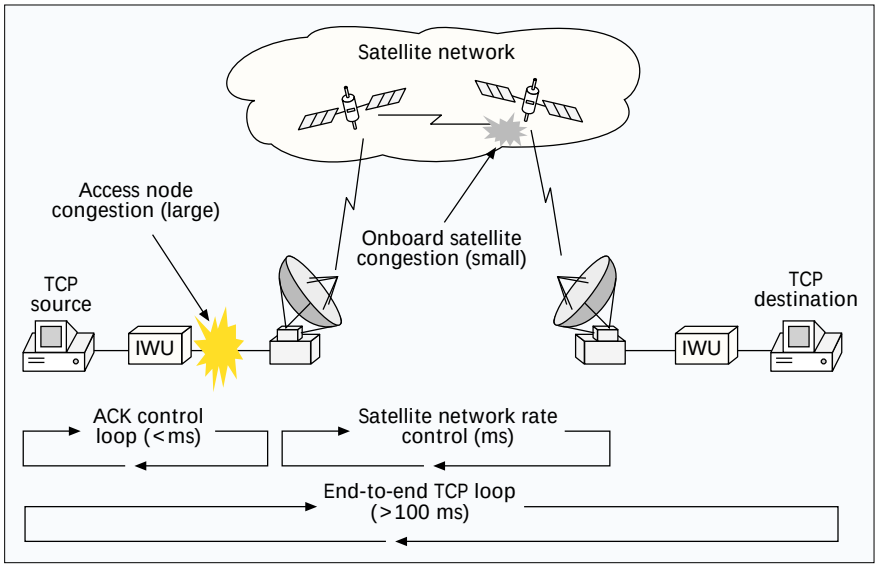

Figure 5. *TCP/IP ACK control in a satellite network.*

preferably pre-established "spoofed" link-layer connections can also be designed to reduce signaling overheads/latencies.

ACK Control Schemes

Recently researchers have tried to improve TCP performance over ATM more directly by using *ACK control* schemes [12, 13]. Since there is a large timescale mismatch between the TCP feedback loop (hundreds of milliseconds) and ATM congestion control (milliseconds), the intention here is to induce a tighter coupling (Fig. 5). These schemes are basically *congestion* control techniques and assume that TCP sources can congest the network, implying adequate link FEC and/or multiple TCP sessions. In general, two such schemes are of note, namely those delaying returning ACK packets (*ACK pacing* [12]) and those modifying the receiver window field in returning ACK packets (*receiver window modification* [13]). A benefit of ACK control is that no changes are necessary for TCP protocol stacks, and only sender-side complexity is incurred. Moreover, ACK control schemes are not ATM-specific and are very well-suited for satellite networks. Specifically, returning ACK packets can be controlled at the ground station's IWU contingent on congestion levels at the forward (outgoing) data buffers (Fig. 5). By concentrating buffering to the edge of a noisy link, onboard processing requirements can also be reduced. However, ACK control schemes assume relatively good rate control in the core (i.e., in this case satellite) network, essentially abstracting the network to a fairly constant bandwidth "pipe" [12].

ACK pacing (*ACK spacing* [3]) can improve TCP's interaction with more advanced ATM transport methodologies, such as per-connection queuing and feedback flow control [12 and references therein]. ACK pacing exploits the fact that TCP's emission behavior (window growth) is controlled by the *rate* of returning ACK packets [16], and hence appropriately delays returning ACK packets to control forward access congestion. Figure 6 presents a simplified overview of such a device, where the *ACK control* unit performs the necessary functions. A key benefit is that ACK pacing can provide good buffer size reduction for bulk TCP transfers, since queue buildups are more dependent on access network delays, assuming good rate control. In this regard, other TCP control schemes still require buffering on the order of round-trip delays [4, 13]. In addition, per-flow (or per-class) ACK buffering can improve bandwidth fairness between competing TCP transfers. The implementation complexity of ACK pacing ranges from moderate to complex. Namely, buffering ACK packets poses nominal storage overheads (due to their relatively small size, around 40 bytes). However, advanced schemes can require per-flow accounting and even ACK scheduler mechanisms.

To show some benefits of ACK pacing, the enhanced ACK pacing (EAP) [12] scheme is tested using an MEO-link scenario. The forward channel is 50 Mb/s and has 120 ms delay, whereas the reverse channel is 1.5 Mb/s and has 100 ms delay. Also, the link BER values are varied dynamically between the 10^{-9} – 10^{-10} range, implying good FEC. The results are presented for three competing bulk transfers, each using the previously defined multiple connections approach, varying between one and three (sub)connections. The access delays (TCP workstations to ASIU) are fixed at 5 ms, and the ASIU buffering capacity is varied from fractional to full bandwidth-delay product range (roughly 30,000 cells). The TCP performance is measured using the multiple connections approach (with up to three subconnections), whereas ACK pacing only uses a single connection. The bandwidth utilization results (Fig. 7) indicate that ACK pacing provides good savings in buffering capacity in addition to improved utilization (over 20 percent on average). For example, even with an access buffer size of only 10 percent of the bandwidth delay product, average link utilization is still about 95 percent versus about 65 percent with TCP Reno. To gauge bandwidth fairness, average throughputs are measured. Specifically, each user should receive about 15 Mb/s goodput (after accounting for ATM cell and IP header

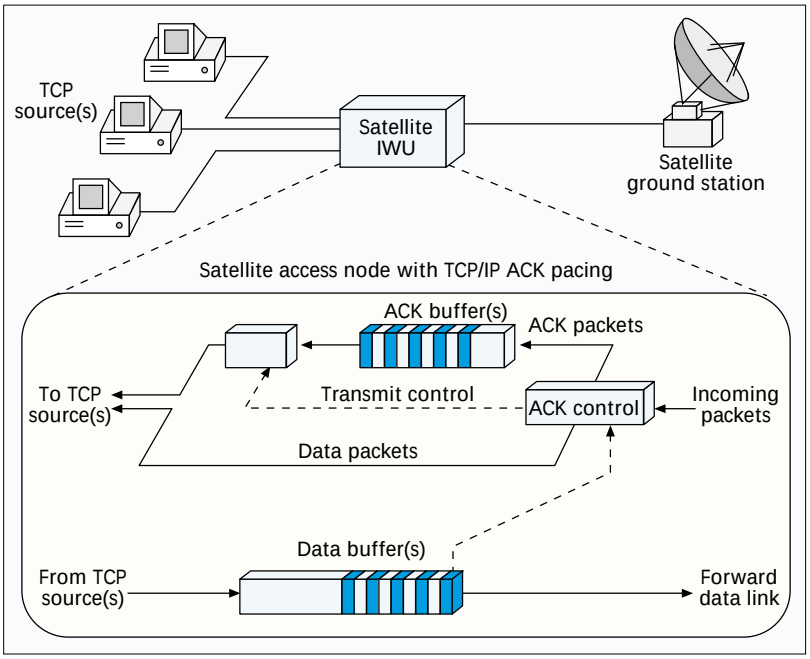

Figure 6. *An overview of a TCP ACK pacing module.*

overheads). The average deviation from this value is plotted in Fig. 8 for varying buffer sizes, and again, ACK pacing yields very small deviations, about 2 percent. Although TCP shows improved fairness with much larger buffer sizes, its throughput deviation does not improve much beyond 25 percent (i.e., 3 Mb/s, Fig. 8). To illustrate ACK pacing performance improvements more closely, the ASIU queue length is plotted in Fig. 9, showing tight control about a prespecified buffer threshold of 1500 cells. The received-byte counts for all three bulk transfers are plotted in Fig. 10, and from the slopes it is clear that all of them are getting equal throughputs. In general, extensive tests show that ACK pacing throughputs are usually much better (and not lower) than those with basic TCP for a given buffer size. Since the above results are for a 50 Mb/s link rate, buffer savings will increase further for larger link rates. Although these results are encouraging, further study is necessary, especially with bursty transfers.

Window-scaling techniques have also been tested at the network edge to modulate source activity [13]. Specifically, the receiver window fields of returning ACK packets are over-written (reduced) by congested access nodes to control emission behaviors. Like ACK pacing, these schemes assert more direct control over TCP connections. Although core (not access) buffering requirements are reduced, some of these schemes require knowledge of individual connections' round-trip delay times and end-to-end bandwidth levels [13]. Measuring these accurately in dynamic environments may be difficult. In fact, the bandwidth fairness levels and queue behaviors will vary depending on the accuracy of these estimates, making them less robust than ACK pacing schemes. Nevertheless, receiver window control has some benefits, especially for controlling bursty sources. Therefore, future work could investigate combining the two approaches. It should be noted, however, that IP security concerns may complicate TCP header modification, and this issue needs further investigation.

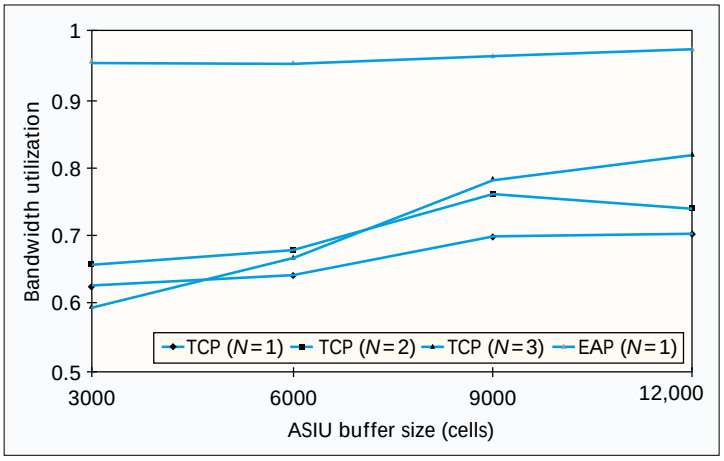

Figure 7. *Bandwidth utilization with ACK pacing/multiple connections.*

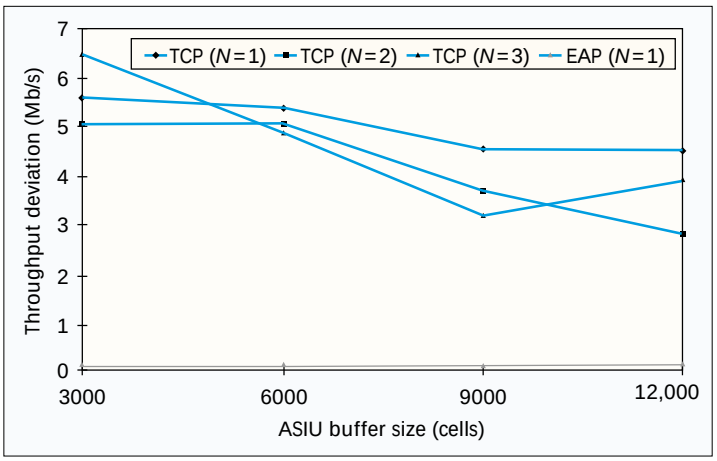

Figure 8. *Average throughput deviation from ideal 15 Mb/s fair share.*

Improved Satellite Features

Satellite systems are now beginning to support improved access and onboard traffic management features [2]. In this context, a significant amount of work has been done on TCP transport over satellite ATM networks [4, 10, 14, 15] using advanced scheduling, buffer management, and even feedback flow control (ABR) algorithms. In general, such features can improve TCP performance, in terms of both throughput and fairness, although TCP modifications, especially SACK, provide more notable performance gains for longer delays [4]. However, many ATM-related studies have not considered the effects of link error rates. It is likely that smaller buffer sizes may give roughly the same link utilization performance as larger values in the presence of increased link error rates, since link losses tend to reduce TCP emissions anyway. Another issue, ensuring bandwidth fairness between competing applications under noisy link conditions, may be much more difficult due to corruption-induced losses. The existing work is now briefly discussed, and interested readers are referred to [4] for more thorough treatment.

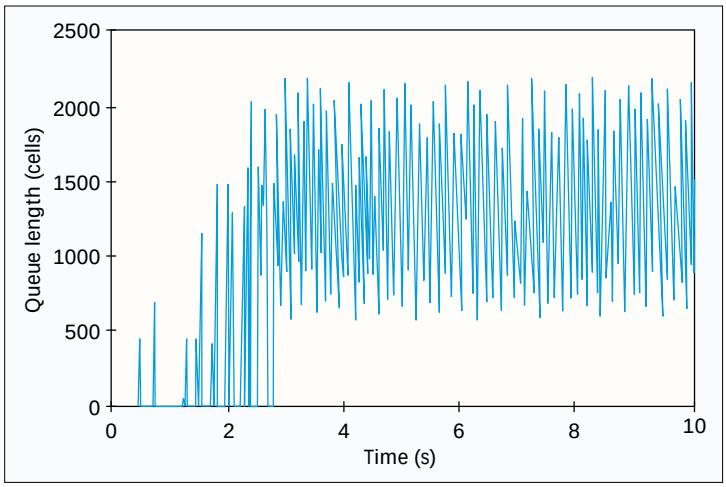

Figure 9. *ASIU queue length with ACK pacing (1500-cell threshold).*

Many buffer allocation schemes have been proposed to improve throughput and fairness for TCP flows in more general packet network settings. Although simpler schemes such as partial packet discarding (PPD) and random early discard (RED) can improve throughput significantly, usually more advanced renditions are required to improve fairness. These schemes use per-flow accounting to perform intelligent drops and are applicable to various ATM service categories — for example, fair buffer allocation (FBA) or flow random early discard (FRED); see [4] for full taxonomy. Additionally, for ATM networks, the guaranteed frame rate (GFR) category can further improve fairness over the simpler unspecified bit rate (UBR) category by providing rate guarantees. Nevertheless, UBR or GFR buffer allocation solutions cannot fully exploit larger access

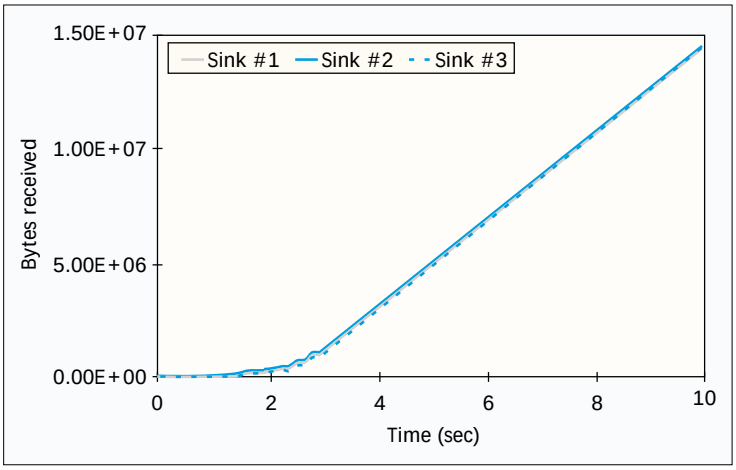

Figure 10. *Received byte counts with ACK pacing.*

buffering and usually imply increased storage/processing complexity at all network nodes (even onboard satellite nodes). Furthermore, results in terrestrial WAN (and satellite) networks are not as good, since window and buffer sizes are of the same order [4]. Some buffer management schemes can also exhibit a strong trade-off between link utilization (throughput) and fairness levels for TCP Tahoe/Reno [10]. More complex solutions can use ATM ABR flow control schemes. If access nodes have such feedback capabilities, the effects of long one-way propagation times can be reduced (VS/VD feature), and more important, increased access buffering can improve throughput and fairness. Results with advanced explicit rate (ER) ABR schemes over lossless satellite links show reduced buffering requirements for multiple flows compared to UBR and GFR [4, 10]. Findings with simpler binary feedback schemes, namely explicit forward congestion indication (EFCI), are not as good in terms of throughput and fairness. In general, advanced ABR ER flow control capabilities can greatly reduce storage requirements onboard satellite and downstream (receiver) nodes. Coupling ABR flow control with ACK control can potentially provide further gains.

Conclusions

As satellites begin to play an increasingly important role in modern communication infrastructures, there is a growing demand for them to support a broader range of user services. An important concern in communication networks is the reliable transport of data traffic, and TCP/IP protocol has emerged as a widely used solution. However, many features of satellite networks pose significant challenges for achieving good TCP performance. Among other issues, these include long delays, increased bit error rates, network asymmetries, and onboard restrictions. Effective solutions to these problems are necessary to ensure reliable end-to-end data delivery.

Many schemes have been proposed for improving satellite TCP performance. On a lower level, improved link-level encoding and access protocols can reduce susceptibility to link error rates. More direct TCP-specific solutions also exist. These include basic parameter tuning/extensions, improved timer mechanisms, and advanced data recovery procedures such as SACK, FACK, and NewReno. Other schemes such as link-layer interworking techniques or advanced ACK control algorithms can also bring benefits. The latter can provide improved TCP throughput and fairness and also reduce buffer size requirements. Overall, a rich variety of solutions have been developed for satellite TCP, and forthcoming work will continue to meet the challenges in this important area.

References

[1] I. Akyildiz and S. Jeong, "Satellite ATM Networks: A Survey," *IEEE Commun. Mag.*, vol. 35, no. 7, July 1997, pp. 30–43.
[2] J. Gilderson and J. Chekaoui, "Onboard Switching for ATM via Satellite," *IEEE Commun. Mag.*, vol. 35, no. 7, July 1997, pp. 66–70.
[3] M. Allman *et al.*, "Ongoing TCP Research Related to Satellites," Internet draft, draft-ietf-tcpsat-res-issues-08.txt, June 1999.
[4] R. Goyal *et al.*, "Traffic Management for TCP/IP Over Satellite ATM Networks," *IEEE Commun. Mag.*, Mar. 1999, pp. 56–61.
[5] R. Durst, G. Miller, and E. Travis, "TCP Extensions for Space Communications," *Wireless Networks*, Oct. 1997, vol. 3, no. 5, pp. 389–403.
[6] M. Allman and D. Glover, "Enhancing TCP Over Satellite Channels Using Standard Mechanisms," Internet RFC 2488, Jan. 1999.
[7] M. Mathis, and J. Mahdavi, "Forward Acknowledgment: Refining TCP Congestion Control," *Proc. ACM Sigcomm 1996*, Stanford, CA, Aug. 1996.
[8] K. Fall and S. Floyd, "Simulation-based Comparisons of Tahoe, Reno, and SACK TCP," *Comp. Commun. Rev.*, vol. 26, no. 3, July 1996, pp. 5–21.
[9] R. Bruyeron, B. Hemon, and L. Zhang, "Experimentations with TCP Selective Acknowledgment," *ACM Comp. Commun. Rev.*, vol. 28, no. 2, Apr. 1998, pp. 54–77.
[10] S. Kalynaraman et al., "Buffer Requirements for TCP/IP over ABR," *Proc. ATM '96 Wksp.*, San Francisco, CA, Aug. 1996.
[11] M. Allman *et al.*, "An Application-Level Solution to TCP's Satellite Inefficiencies," *Proc. 1st Int'l. Wksp. Satellite-Based Info. Services*, Nov. 1996.
[12] N. Ghani, "Enhanced TCP/IP ACK Pacing for High Speed Networks," *Proc. 16th Int'l. Teletraffic Cong.*, Edinburgh, Scotland, June 1999.
[13] R. Satyavolu, K. Duvedi, and S. Kalyanaraman, "Explicit Rate Control of TCP Applications," ATM Forum-Traffic Management 98-0152, Feb. 1998.
[14] R. Goyal *et al.*, "Analysis and Simulation of Delay and Buffer Requirements of Satellite-ATM Networks for TCP/IP Traffic," work in progress.
[15] M. Allman, "TCP Performance Over Satellite Links," *Proc. ComSoc Wksp. Comp.-Aided Modeling, Analysis and Design of Commun. Links and Networks*, Mclean, VA, Oct. 1996.
[16] T. Lakshman *et al.*, "Window-Based Error Recovery and Flow Control with a Slow Acknowledgement Channel: A Study of TCP/IP Performance," *Proc. IEEE INFOCOM 1997*, Kobe, Japan, Apr. 1997.
[17] M. Allman, "On the Generation and Use of TCP Acknowledgments," *ACM Comp. Commun. Rev.*, Oct. 1998.
[18] H. Balakrishnan *et al.*, "TCP Behavior of a Busy Internet Server: Analysis and Improvements," *Proc. IEEE INFOCOM 1997*, Kobe, Japan, Apr. 1997.

Biographies

NASIR GHANI [M] (nasir.ghani@research.nokia.com) received a B. Eng. degree in computer engineering from the University of Waterloo, Canada, in 1991, an M. Eng. degree in electrical engineering from McMaster University, Hamilton, Canada, in 1992, and a Ph.D. degree in electrical and computer engineering from the University of Waterloo, Canada, in 1997. From 1992 to 1993 he was employed at IBM Canada as a systems analyst and subsequently, from 1993 to 1994, as a design engineer at Motorola Codex, Canada. Currently he is a senior research engineer in the Broadband Networks group at Nokia Research Center, Burlington, Massachusetts, investigating end-to-end services provisioning over IP, ATM, and optical networks. His research interests include traffic management, flow control, routing, scheduling, IP internetworking, TCP/IP, fault management, and optical networks.

SUDHIR DIXIT [SM] (sudhir.dixit@research.nokia.com) received a B.E. degree from Maulana Azad College of Technology (MACT), Bhopal, India, an M.E. degree from Birla Institute of Technology and Science (BITS), Pilani, India, and a Ph.D. degree from the University of Strathclyde, Glasgow, Scotland, all in electrical engineering. He also received an M.B.A. degree from Florida Institute of Technology, Melbourne. He joined Nokia Research Center, Burlington, Massachusetts in 1996, where he is responsible for managing research and development in broadband networking, specializing in such areas as ATM, Internet, and all-optical networks. From 1991 to 1996 he was a broadband network architect at NYNEX Science and Technology (now Bell Atlantic). Prior to that he held various engineering and management positions at other major companies, such as GTE, Motorola, Harris, and Nortel. During his career, he has acquired experience in such diverse areas as image processing, compression, computer graphics, signal processing, telecommunications, computer networks, and VLSI design. He was a guest editor for a special issue of *IEEE Network* on digital video dial-tone networks published in Oct/Nov 1995, and a guest editor for a feature topic on service and network interworking in a WAN environment published in *IEEE Communications Magazine* in June 1996.