Abstract

Traditional transport layer flow control schemes proposed for IP networks have low performance when satellite links are involved in the communication. In this article, these problems are described along with the solutions recently proposed in the literature. For each solution the advantages and drawbacks of the existing solutions are pointed out. Both real-time and non-real-time applications are considered.

Research Issues for Transport Protocols in Satellite IP Networks

IAN F. AKYILDIZ, GEORGIA INSTITUTE OF TECHNOLOGY

GIACOMO MORABITO AND SERGIO PALAZZO, UNIVERSITY OF CATANIA

In this article we survey the problems of transport protocols in satellite IP networks along with the most recent solutions proposed in the literature. Non-real-time applications use Transmission Control Protocol (TCP). Current TCP schemes have throughput problems in networks with long propagation delays and relatively high link error rates such as satellite networks [1]. TCP throughput decreases mostly because:

- The long propagation delays cause longer duration of the *slow start* phase during which the TCP sender may not use the available bandwidth.
- TCP was initially designed to work in networks with low link error rates (i.e., most packet losses were due to network congestion). As a result, the TCP sender decreases its transmission rate. However, this causes unnecessary throughput degradation if packet losses occur due to link errors.

In a shared network such as the Internet, real-time traffic flows are also expected to be *TCP-friendly* [2]; that is, they should comply with the following rules:

- *Rule 1*: Their transmission rate can increase slowly as long as the network is not congested.
- *Rule 2*: Their transmission rate must decrease immediately when the network is congested.

Next-generation IP routers will penalize traffic flows not compliant with these rules.

For real-time applications there have been several studies in the recent past [2–7]. The transmission rate value is adjusted in such a way that rules 1 and 2 are satisfied.

In order to comply with rule 2, traffic sources decrease their transmission rate when packet losses are detected because these are the only congestion signal in the current Internet. However, in satellite networks packet losses can occur due to link errors with probability even higher than 10^{-2}. If the source decreases its transmission rate when a packet loss occurs due to link errors, the network efficiency decreases drastically.

TCP Problems in Satellite IP Networks and Related Work

Slow Start Problems

In the beginning of a new connection, TCP has no information about the traffic load on the connection path; thus, it executes the slow start [8] to probe the availability of bandwidth along the path. The congestion window, *cwnd*, which gives the maximum number of unacknowledged packets[1] the sender can have in transit to the receiver, is set to one and then incremented by one for each received acknowledgment (ACK). For TCP connections the transmission rate, b, is approximately

$$b \approx \mathrm{cwnd}/\mathrm{RTT}, \quad (1)$$

where RTT is the round-trip time. Therefore, the time required by slow start to reach a bit rate B is

$$t_{SS} \approx \mathrm{RTT} \cdot (1 + \log_2 B \cdot \mathrm{RTT}/l), \quad (2)$$

where l is the average packet length expressed in bits. Note that Eq. 2 is valid when the *delayed ACK option* is not implemented. In Table I we give the duration of the slow start phase for low earth orbit (LEO), medium earth orbit (MEO) and geosynchronous earth orbit (GEO) satellites, and for different values of B when l = 1 kB, which is a common value for packet size.

Many actual TCP applications, such as Hypertext Transfer Protocol (HTTP), are based on the transfer of small files. Thus, the entire transfer may occur within the slow start phase. In other words, it is possible that a TCP connection is not able to utilize all available resources in the network. To

[1] *In this article we use the packet as the unit of data.*

1070-9916/01/$10.00 © 2001 IEEE

Satellite type	t_{SS} (B = 1 Mb/s)	t_{SS} (B = 10 Mb/s)	t_{SS} (B = 155 Mb/s)
LEO	0.18 s	0.35 s	0.55 s
MEO	1.49 s	2.32 s	3.31 s
GEO	3.91 s	5.11 s	7.91 s

■ **Table 1.** *Duration of the slow start phase for different satellite systems and different values of* B.

cope with the performance problems of the slow start algorithm in satellite networks, several solutions have been proposed in recent years [1].

Increasing the initial window [9, 10]: The congestion window, *cwnd*, is initially set to a value larger than 1 but lower than 4 ($1 \leq cwnd \leq 4$). With this option, t_{SS} values reported in Table 1 are reduced by up to ($3 \cdot$ RTT). However, these values can still be very high.

TCP spoofing [11]. A router near the source sends back ACKs for TCP packets in order to give the source the illusion of a short delay path. TCP spoofing improves throughput performance but has some problems:

- *Problem 1*: The router must do a considerable amount of work because it becomes responsible for correct delivery of the TCP segments it acknowledges to the source.
- *Problem 2*: Spoofing requires ACKs to flow through the same path as data. On the contrary, in the Internet it is very common for ACKs to flow through a different path than data.
- *Problem 3*: If the path changes or the router crashes, data may get lost.
- *Problem 4*: If IP encryption is used, the scheme cannot be applied.

Cascading TCP or split TCP [11]: TCP connection is divided into multiple connections. This solution has the same problems as TCP spoofing except for problem 2.

Fast start [12]. The fast start algorithm, an alternative to slow start, is introduced for Web transfers in [12]. The basic idea of fast start is to reuse transmission rate values of the recent past. However, the transmission rate used in the past might be too high for current actual network conditions, which may lead to congestion in the network. Thus, the TCP segments transmitted during this fast start period are carried by low-priority IP packets so that the throughput of actual data segments treated as high-priority segments will not decrease. Note that one of the eight bits of the *type of service* (TOS) field, now renamed the *differentiated service* (DS) field, in the IP header specifies the priority of the packet in traditional IP. More recent IP implementations aimed to support the DiffServ model can define several priority levels. Experiments in [12] show the effectiveness of the fast start algorithm compared to slow start. However, fast start has the following problems:

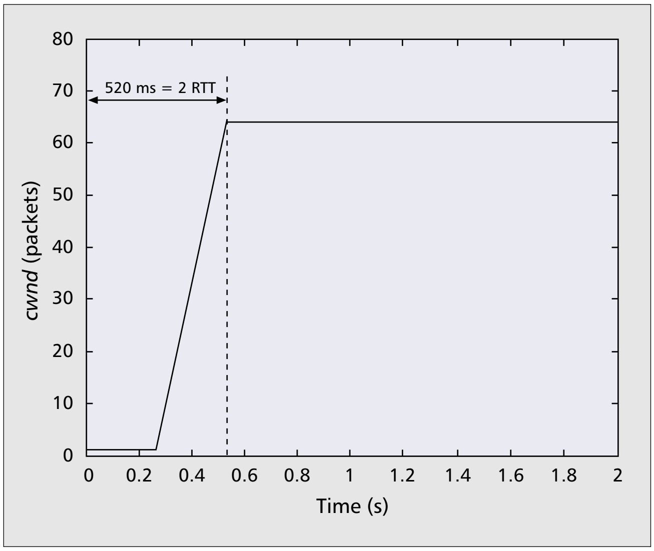
- *Problem 1*: The low-priority packets transmitted carry new information to the TCP receiver; thus, they are still data packets, and if they are lost, they must be recovered. Since these low-priority data packets may be lost easily, the TCP sender needs to enhance its recovery algorithms [12].
- *Problem 2*: Fast start can be used only if a recent value of the congestion window for the same path is available at the sender. This requires that within a short time the same server (sender) transfers several files to the same user (receiver), which may often not be the case.

Sudden start [13]: The sudden start algorithm is executed at the beginning of a new connection in order to avoid the low throughput performance of slow start. The sudden start algorithm is based on the use of packets not carrying any new information to the receiver, called *dummy packets*. The dummy packets are treated as low-priority packets; therefore, they can reach the receiver only if there are resources unused by conventional data packets in the connection path. In other words, they do not affect the transfer of conventional data packets. The receiver acknowledges the dummy packets. For the sender, an ACK for a dummy packet is the sign that there are still unused resources in the network; accordingly, it increases its congestion window. The sudden start algorithm lasts for one RTT; then the sender enters the congestion avoidance phase. During the sudden start the sender transmits one data packet and (*rwnd* – 1) dummy packets, where *rwnd* is the maximum value allowed for the congestion window size provided by the receiver. The dummy packets reach the receiver, and thus their ACKs arrive at the sender, only if there are unused resources in the network. The sender will receive these ACKs when the sudden start algorithm is over and the congestion avoidance algorithm is running. In this phase, the sender will increase its congestion window, *cwnd*, by one packet each time it receives an ACK for a dummy packet. The congestion avoidance is modified accordingly. As a result, if n_{Absorbed} is the number of dummy packets the network is able to absorb, then, at the end of the sudden start algorithm, the transmission rate for the new connection suddenly jumps from 1/RTT to ($n_{Absorbed}$/RTT). In Fig. 1 we show the behavior of cwnd for a TCP connection using the sudden start algorithm. The figure was obtained using RTT = 0.26 s and *rwnd* = 64 packets. Note that *cwnd* reaches *rwnd* within two RTTs.

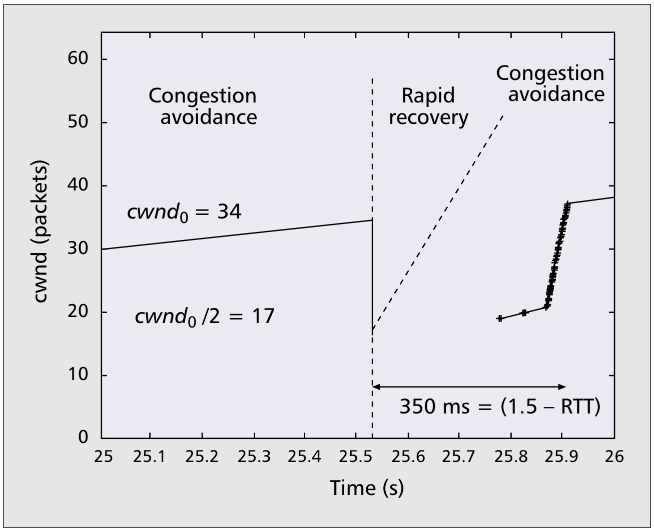
The main problem of sudden start is that it requires all the routers in the connection path to apply some type of priority mechanism [13].

Problems Due to Link Errors

Link errors affect two aspects of throughput. First, the corrupted data packets due to link errors must be retransmitted. Second, the TCP sender always assumes a packet loss as the signal of congestion and accordingly decreases its transmission rate. Due to high error rates characterizing satellite links, this assumption may lead to drastic and unnecessary decrease in resource utilization. This problem is amplified by the long delay characterizing satellite networks. To cope with this problem, several solutions have been proposed in the literature.



■ **Figure 1.** *The initial behavior of TCP with sudden start.*



■ **Figure 2.** *The behavior of rapid recovery.*

Explicit congestion notification (ECN) [14]: ECN mechanisms notify the end nodes of incipient congestion by setting the ECN bit in the IP packet header to one. Accordingly, the senders decrease the transmission rate. Therefore, heavy congestion is avoided and unnecessary packet drops are prevented. Besides, the senders can be informed of congestion without waiting for either a retransmission timer or three duplicate ACKs. Therefore, if a packet is lost and congestion is not notified, the loss is assumed to be due to link errors. Accordingly, the transmission rate is not decreased. The main problem of ECN is that it requires all the routers in the Internet to be redesigned accordingly.

Link corruption notification [15]. This algorithm is part of the Space Communication Protocol Standard Transport Protocol (SCPS-TP) [15]. The receiver earth station continuously monitors the satellite channel condition. With this objective, it requests the link layer information regarding the number of corrupted packets, N_{Corr}, and maintains a weighted moving average, $\overline{N}_{Corr}$, of N_{Corr}. If $\overline{N}_{Corr}$ exceeds a given threshold N_{Thr}, the receiver earth station assumes that the channel condition is bad, and enters the link-corrupted state, and sends *corruption experienced* messages to the destinations. The senders are informed of the bad channel condition by receiving appropriately marked ACKs. The sender behavior depends on the channel condition:

- If the channel condition is good, the cause of packet losses is assumed to be network congestion as in traditional TCP implementations.
- If the channel condition is bad, the cause of packet losses is assumed to be link errors, and the transmission rate is controlled using an open-loop token bucket [15].

The problems of this technique are:

- It requires a modification of the link layer which needs to report information about the channel condition to the transport layer.
- It is difficult to set a proper transmission rate for the token bucket and a proper value for N_{Thr}.

Decoupling error and congestion control [16]: Error and congestion control are decoupled. TCP would then be responsible only for congestion control, while error control is handled by the link layer. This solution is impractical because the link layers of all subnetworks composing the network need to be redesigned.

Rapid recovery [13]: The rapid recovery algorithm first keeps the classical fast recovery conservative assumption that all packet losses are due to network congestion. Accordingly, the TCP sender halves *cwnd*, as in TCP-Reno. However, in order to probe the availability of network resources, the TCP sender transmits a certain number of dummy packets. If the packet loss is due to link errors (i.e., the network is not congested), the dummy packets will reach the receiver and be acknowledged to the sender, which will increase *cwnd* accordingly. Upon a packet loss due to link errors, the congestion window can reach its previous value within two RTTs. In Fig. 2 we show the behavior of rapid recovery when a packet loss occurs due to link errors. Figure 2 was obtained experimentally in a physical testbed with 260 ms RTT. In Fig. 2, a data packet loss due to link errors is detected at time $t = 25.52$ s when the congestion window is $cwnd = 34$ packets. Accordingly, the lost packet is retransmitted, the congestion window is halved (i.e., $cwnd = 17$ packets), and rapid recovery is executed. Rapid recovery lasts approximately for one RTT. Then the sender returns to the congestion avoidance phase. During this phase, the ACKs for the dummy packets transmitted during rapid recovery are received, and thus the congestion window, *cwnd*, increases rapidly. At time $t = 25.91$ the congestion window reaches the value it had before the packet loss was detected. Therefore, the time, δt, needed for *cwnd* to recover completely from a packet loss due to link errors is approximately one-half RTT.

The main problems of rapid recovery are:

- It requires all the routers on the connection path to implement some priority mechanism [13].
- Since dummy packets do not carry any new information, they cause overhead for the network. For example, for high bit error rates the overhead can be as high as 17.21 percent. However, the rapid recovery gives higher throughput than the fast recovery.

Problems Due to Bandwidth Asymmetry

Let us define:

- **Forward channel**: The channel from the sender to the receiver. Data packets flow through the forward channel.
- **Backward channel**: The channel from the receiver to the sender. ACK packets flow through the reverse channel.

Most configurations in satellite communications are bandwidth asymmetric [17], that is, the bandwidth available in the forward channel is much higher than that in the reverse channel. As a result, ACK packets may congest the reverse channel, and be delayed and lost. This causes two performance problems:

- Traffic burstiness increases.
- The throughput may decrease because TCP interprets the delay of an ACK or its loss as a sign that the forward channel is congested and, accordingly, falsely decreases the transmission rate.

Header compression techniques [18] can alleviate these problems but do not solve them completely. As a consequence, many solutions have been proposed in the last few years.

Periodic acknowledgment [15]: This algorithm is part of SCPS-TP [15]. The receiver does not acknowledge every data packet. ACKs are transmitted at a constant rate dependent on RTT. Accordingly, the flow of ACK packets in the reverse channel is lighter than in traditional TCP implementations. However, it is not easy to determine the optimal value of the acknowledgment rate, and the receiver does not respond properly to congestion in the reverse channel.

ACK congestion control (ACC) [19]: ACKs are delayed by a factor *d*, that is, one ACK is transmitted for each *d* data packets. If the average queue size in a router on the reverse channel exceeds a given threshold, the ECN bit of the ACK packets is set to one. The receiver is notified of incipient congestion in the reverse channel, and increases the value of *d* and thus

decreases the ACK flow. ACC alleviates congestion problems in the reverse channel but has the following problems:
- The traffic burstiness increases.
- The window growth slows down.
- The fast retransmit efficiency decreases.

ACK filter (AF) [19, 20]: When an ACK is about to enqueue, the router checks whether there are any previous ACKs belonging to the same connection in the queue. If this is the case, the router removes some or all of these redundant ACKs and saves space for other ACKs. AF has the same problems as ACC. Moreover, it requires IP routers to access transport layer information.

Sender adaptation (SA) [19]. TCP SA is proposed to adjust the transmission rate and window size so that a TCP sender can adapt well to situation where fewer ACKs are received. To overcome the burstiness problem, SA sets an upper bound, *UB*, on the number of packets the sender can transmit back-to-back and breaks large bursts of data into smaller bursts spread out over time. To avoid slowed down window growth, the sender considers the amount of data acknowledged by each ACK, not the number of ACKs. Note that:
- If *UB* decreases, the transmission rate and thus the network efficiency decrease.
- If *UB* increases, the traffic burstiness increases.

It is very difficult to find an appropriate trade-off, that is, to set *UB* to a value appropriate for any traffic load and network configuration.

ACK reconstruction (AR) [19]: The ACK stream is reconstructed after it traverses the bottleneck link on the reverse channel. A large number of ACKs are transmitted at a consistent rate so that the burstiness is low and the congestion window can increase fast. The main problems of AR are:
- Intermediate IP routers are responsible for a large amount of work.
- IP routers need to access transport layer information.

Satellite Transport Protocol (STP) [17]. The transmitter earth station sends data packets and stores them for potential retransmission until they are acknowledged. The transmitter earth station also periodically sends one POLL packet to ask the receiver which packets have been successfully received. The receiver sends a STAT packet as a response. As soon as a packet loss is detected, the receiver informs the transmitter of the packet loss explicitly. Therefore, the bandwidth usage on the reverse channel mainly depends on the polling period, not the transmission rate on the forward channel. In addition, selective negative acknowledgment can inform the sender about packet losses within half of an RTT. STP has some limitations [17]; for example, it can only be used as either the satellite portion of a split TCP connection, or the transport protocol for control and network management traffic within a satellite network.

Problems of Real-Time Applications and Related Work

Motivation

Several users are interested in using real-time applications at low cost. These users will share network resources without any reservation; thus, their transmission rate must adapt to network load conditions. In particular, their transmission rate, *S*, must decrease immediately when a network congestion is detected. Recently, much research has been done to define TCP-friendly rate control protocols for real-time applications in terrestrial networks. For example, in [3] a TCP-like scheme that does not perform retransmissions is proposed. The

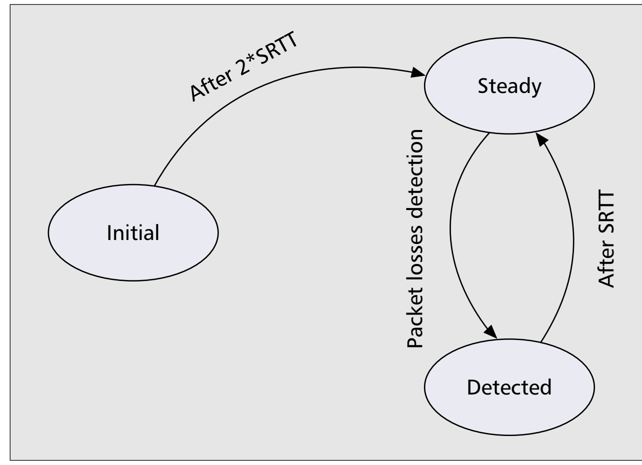


Figure 3. *The behavior of rapid recovery.*

Streaming Control Protocol (SCP) is introduced in [6]. SCP is a modified version of TCP that performs TCP-Vegas-like rate adjustment. In [4, 5] the transmission rate is adjusted based on TCP throughput models. In [2, 7], two rate adaptation protocols, LDA and RAP, are presented. Both of them perform flow control for real-time streams by means of mechanisms very similar to those of TCP.

All the above schemes assume that packet losses are due to network congestion and accordingly decrease the transmission rate. However, in satellite networks packet losses may occur due to link errors. Decreasing the transmission rate when the network is not congested causes a dramatic decrease in network efficiency. This problem is amplified by the long propagation delays.

Another problem for adaptive real-time applications in the Internet is the choice of an appropriate initial transmission rate, $S_{Initial}$. In fact, let $S_{Available}$ be the transmission rate sustainable by the network. This choice is important because:
- If $S_{Initial} \gg S_{Available}$, the new connection will cause network congestion.
- If $S_{Initial} \ll S_{Available}$, resource utilization is low and will be low for a time period proportional to the RTT.

To the best of our knowledge, the Rate Control Scheme (RCS) [21] is the only solution for real-time traffic in satellite IP networks to date, and it provides an efficient solution for the problems mentioned above.

The Rate Control Scheme

RCS uses the additive-increase, multiplicative-decrease (AIMD) discipline and the concept of dummy packets to probe the resources in the network, in order to produce TCP-friendly traffic flows while maintaining high throughput performance in satellite networks.

RCS runs on top of RTP/RTCP and UDP and is mainly implemented at the source but needs some functions at the receiver. Note that no retransmission is performed. At the destination, the RCS layer acknowledges the received packets. Note that ACKs are used only for flow control. RCS is a finite state machine model with three states: *initial*, *steady*, and *detected*, as shown in Fig. 3. In the beginning of a new connection, the source is in the initial state and sends dummy packets at the rate needed to transmit the real-time stream with the highest quality. The destination acknowledges the received dummy packets. At the end of two RTTs, the source sets its data transmission rate according to the number of dummy packets acknowledged by the destination and enters the steady state.

In the steady state, the RCS source assumes that the network is not congested. Thus, according to the additive-increase scheme, it increases its transmission rate in a steplike fashion periodically. However, as soon as a data packet loss is detected, the source enters the detected state, halves its data transmission rate, and sends dummy packets. If the packet loss is due to network congestion, the router drops the dummy packets because they have low priority. As a consequence, the source does not receive the ACKs for dummy packets and maintains its data transmission rate at a low value. On the contrary, if the packet loss is due to link errors, the dummy packets can reach the destination and will be acknowledged. The source then increases its data transmission rate according to the number of dummy packets acknowledged. The source returns back to the steady state after one estimated RTT. Using dummy packets, RCS achieves high throughput performance as well as TCP friendliness.

The problem of RCS is the overhead caused by dummy packet transmission. We define overhead as the bandwidth occupied by dummy packet transmission over the bandwidth occupied by total packet transmission. The overhead can be as high as 21.5 percent when the loss probability is high. However, using dummy packets, the RCS provides higher throughput. Moreover, it achieves double the throughput of other rate control schemes [2] for real-time traffic in many cases. Another problem of RCS is that it requires all routers in the connection path to support some priority policy [21].

Conclusions

Traditional transport layer flow control schemes for IP networks have low performance when satellite links are involved in the communication. This is an important problem because satellites will play a key role in the Internet infrastructure for both developing and developed countries. Consequently, much research effort has been made to optimize transport protocols for satellite networks. Most recent work concerns TCP [1], which is used for non-real-time applications; however, recent work addresses the case of real-time applications as well. In this article we present the basic problems and survey the most recent solutions, pointing out their advantages and disadvantages.

Acknowledgments

The work of I. F. A. and G. M. was supported by NASA-Ames under contract NAG 2-1262. G. M. was with the Broadband and Wireless Networking Laboratory at Georgia Institute of Technology.

References

[1] M. Allman *et al.*, "Ongoing TCP Research Related to Satellites," RFC 2760, Feb. 2000.
[2] R. Rejaie, M. Handley, and D. Estrin, "RAP: An End-to-End Rate-Based Congestion Control Mechanism for Realtime Streams in the Internet," *Proc. IEEE INFOCOM '99*, Apr. 1999.
[3] S. Jacobs and A. Eleftheriadis, "Real-Time Dynamic Rate Shaping and Control for Internet Video Applications," *Proc. Wksp. Multimedia Sig. Proc. '97*, June 1997.
[4] J. Mahdavi and S. Floyd, "TCP-Friendly Unicast Rate-Based Flow Control," tech. note, June 1997; http://ftp.ee.lbl.gov/floyd/papers.html.
[5] T. Turletti, S. F. Prisis, and J.-C. Bolot, "Experiments with a Layered Transmission Scheme over the Internet," Rapport de recherche 3296, INRIA, Nov. 1997.
[6] S. Cen, C. Pu and J. Walpole, "Flow and Congestion Control for Internet Media Streaming Applications," *Proc. Multimedia Comp. and Net.*, Jan. 1998.
[7] D. Sisalem and H. Schulzrinne, "The Loss-Delay Based Adjustment Algorithm: A TCP-Friendly Adaptation Scheme," *Wksp. Network and Op. Sys. Support for Digital Audio and Video*, July 1998.
[8] V. Jacobson, "Congestion Avoidance and Control," *ACM Sigcomm*, Aug. 1988.
[9] M. Allman, S. Floyd, and C. Partridge, "Increasing TCP's Initial Window," RFC 2414, 1998.
[10] M. Allman, D. Glover, and L. Sanchez, "Enhancing TCP over Satellite Channels Using Standard Mechanism," RFC 2488, 1999.
[11] H. Balakrishnan *et al.*, "A Comparison of Mechanisms for Improving TCP Performance over Wireless Links," *IEEE/ACM Trans. Net.*, Dec. 1997.
[12] V. N. Pamanabhan and R. Katz, "TCP Fast Start: A Technique for Speeding Up Web Transfer," *Proc. IEEE GLOBECOM '98*, Nov. 1998.
[13] I. F. Akyildiz, G. Morabito, and S. Palazzo, "TCP-Peach: A New Congestion Control Scheme for Satellite IP Networks," *IEEE/ACM Trans. Net.*, June 2001.
[14] S. Floyd, "TCP and Explicit Congestion Notification," *ACM Commun. Rev.*, Oct. 1994.
[15] R. C. Durst, G. J. Miller, and E. J. Travis, "TCP Extensions for Space Communications," *Proc. ACM Mobicom '96*, Nov. 1996.
[16] I. F. Akyildiz and I. Joe, "A New ATM Adaptation Layer for TCP/IP over Wireless ATM Networks," *ACM-Baltzer J. Wireless Networks*, June 2000.
[17] T. R. Henderson and R. H. Katz, "Transport Protocols for Internet-Compatible Satellite Networks," *IEEE JSAC*, Feb. 1999.
[18] V. Jacobson, "Compression of TCP/IP Headers for Low-Speed Serial Links," RFC 1144. Feb. 1990.
[19] H. Balakrishnan, V. N. Padmanabhan, and R. H. Katz, "The Effects of Asymmetry on TCP Performance," *Proc. ACM Mobicom '97*, Sept. 1997.
[20] P. Karn, "Dropping TCP ACKs," message to end-to-end mailing list, Feb. 1996.
[21] J. Tang, G. Morabito, I. F. Akyildiz, and M. Johnson, "RCS: A Rate Control Scheme for Real-Time Traffic in Networks with High Bandwidth-Delay Products and High Bit Error Rates," *Proc. IEEE INFOCOM 2001*, Apr. 2001.

Biographies

IAN F. AKYILDIZ [F] (ian@ece.gatech.edu) is a Distinguished Chair Professor with the School of Electrical and Computer Engineering, Georgia Institute of Technology, and director of the Broadband and Wireless Networking Laboratory. His current research interests are in wireless networks, satellite networks, and the Internet. He is Editor-In-Chief of *Computer Networks Journal*, and an Editor for *ACM-Springer Journal for Multimedia Systems*, *ACM-Baltzer Journal of Wireless Networks*, and *Journal of Cluster Computing*. He was an Editor for *IEEE/ACM Transactions on Networking* (1996–2001) and *IEEE Transactions on Computers* (1992–1996). He served as the program chair of the 9th IEEE Computer Communications Workshop in 1994, and as the program chair for ACM/IEEE MOBICOM '96 (Mobile Computing and Networking) and IEEE INFOCOM '98. He will be General Chair for ACM/IEEE MOBICOM 2002 and the Technical Program Chair for IEEE ICC 2003. He is an ACM Fellow. He received the Don Federico Santa Maria Medal for his services to the Universidad de Federico Santa Maria, Chile. He served for the ACM Distinguished Lecturer program (1989–1997), and received the ACM Outstanding Distinguished Lecturer Award for 1994 and the 1997 IEEE Leonard G. Abraham Prize.

GIACOMO MORABITO (gmorabi@iit.unict.it) received his laurea degree in electrical engineering and a Ph.D. degree in electrical, computer, and telecommunications engineering from the Istituto di Informatica e Telecomunicazioni, University of Catania, Italy, in 1996 and 2000, respectively. Since April 2001 he is with the Dipartimento di Informatica e Telecomunicazioni of the University of Catania. From November 1999 to March 200, he was with the Broadband and Wireless Networking Laboratory of the Georgia Institute of Technology. His research interests include satellite and wireless networks, quality of service, and broadband network performance analysis.

SERGIO PALAZZO [M'92] (palazzo@diit.unict.it) received his degree in electrical engineering from the University of Catania, Italy, in 1977. Until 1981 he was at ITALTEL, Milano, where he was involved in the design of operating systems for electronic exchanges. He then joined CREI, which is the center of the Politecnico di Milano for research on computer networks. Since 1987 he has been at the University of Catania, where is now a full professor of telecommunications networks. In 1994 he spent the summer at the International Computer Science Institute (ICSI), Berkeley, California, as a senior visitor. Since 1992 he has served on the Technical Program Committee of INFOCOM, the IEEE Conference on Computer Communications. He has recently been appointed General Vice Chair of ACM MobiCom 2001. He will also be General Chair of the 2001 International Tyrrhenian Workshop on Digital Communications, focused on "Evolutionary Trends of the Internet." Since 1997 he is a member of the Editorial Board of *IEEE Personal Communications* and was a Guest Editor of its Special Issue on Adapting to Network and Client Variability in Wireless Networks. Recently, he joined the Editorial Boards of *Computer Networks and Wireless Communications* and *Mobile Computing*. Also, he was a Guest Editor of the Special Issue of *IEEE JSAC* on "Intelligent Techniques in High-Speed Networks." His current research interests include mobile systems, broadband networks, intelligent techniques in network control, multimedia traffic modeling, and protocols for the next generation of the Internet.