Building Adaptive Applications: On The Need For Congestion Control

Colin Perkins
University of Glasgow
Department of Computing Science
17 Lilybank Gardens
Glasgow G12 8QQ
United Kingdom

20 December 2004

ABSTRACT

The increasing convergence of audio/visual communication networks and IP networks, such as the Internet, has many well-documented benefits. There are, however, problems inherent in adapting video communication systems to run on the less than ideal service provided by typical IP networks. This paper highlights these problems, outlines ongoing standards activities in this area, and describes areas where further research and standards development is needed.

Keywords: Video streaming, congestion control, RTP, TCP/IP, networks.

1. INTRODUCTION

One of the key problems with multimedia communication in the Internet is the lack of quality of service support in the network. There is no deployed way for an application to request network resources, for example to reserve bandwidth or to set up a communication path with low absolute delay or delay variation. This is not due to a lack of interest – considerable research and standards activity has occurred in this area – rather it is because, for various reasons, the resulting standards have not been deployed in the network infrastructure. As a consequence of this, multimedia traffic is left to compete for network resources with other applications, and applications must adapt their transmission rate to match available network capacity through a process known as congestion control. The result is uneven and varied performance, which is a significant impediment to deployment in more demanding domains.

This paper explores the issues with designing adaptive congestion control for real-time multimedia traffic. It explains how the network behaves and the consequences this has for real-time traffic, motivates the congestion control problem, describes ongoing work in the Internet Engineering Task Force (IETF) to provide more suitable congestion control algorithms, and outlines areas where further research into both congestion control and appropriate video coding standards is needed.

The material is structured as follows: section 2 reviews the Internet service model, and outlines the traditional (TCP/IP) and the real-time (RTP over UDP/IP) transport protocols which are used in the network. This is followed by a discussion of congestion control and its implications for real-time video transport in sections 3 and 4. Possible directions in which network transport protcols and media codecs can evolve to provide better performance are discussed in section 5. Finally, conclusions are presented in section 6.

2. REAL-TIME SERVICES ON IP NETWORKS

To understand how competition for network resources affects the performance of multimedia traffic, it is first necessary to review the service model of IP networks and the protocols used to provide real-time transport in this environment.

The service provided by an IP network, such as the Internet, is best effort delivery of datagrams to a host. A host injects a datagram packet into the network, and the network makes its best attempt to deliver that data.

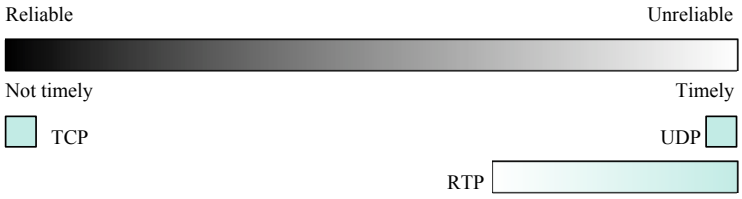

Figure 1. Timeliness vs. Reliability in Transport Protocol Design

There is no reliability or performance guarantee: the network may discard, delay, reorder or corrupt packets in transit. Reliability is the responsibility of higher layer protocols residing in the end hosts, not the network, a design choice influenced by the end-to-end argument.[1] Each transport protocol is free to choose how it responds to different network behaviour: for example the combination of UDP/IP accepts the inherent unreliability of the network in return for timely delivery of data to the application, but TCP/IP masks the unreliable network through retransmission at the expense of unpredictable delays in data delivery. The spectrum of choices is shown in Figure 1.

Modern IP networks are, in general, well behaved. Packet loss or corruption in the network core is rare and queueing delays are generally small. As an example, recent measurements taken by the author and colleagues show only 22 packets being lost from a total of approximately 60 million sent in an experiment to test the performance of a commerical IP backbone network at rates up to 1 Gbps.[2,3] In the same experiment the absolute variation in transit time was small, although sufficient to cause a higher than expected rate of packet reordering. Data corruption was not observed. Other studies[4] confirm these conclusions for the core network, although it is expected that edge networks are less well provisioned and can suffer from transient network congestion. This is most likely to be observed when there are many users behind a relatively slow connection (e.g. a business on a T1 line) or when a single host is connected via a low-capacity link (e.g. a dial-up or mobile user). The implication is that while much of the network is loss free, there are significant regions where transient congestion can occur and cause packet loss or queueing delay. It is necessary to design systems that are tolerant to some degree of packet loss, delay and reordering. These effects may be handled by the transport layer, or may be exposed to the application, depending on the transport protocol used.

In addition to pushing responsibility for reliable transmission to the end points, the network also pushes responsibility for congestion control and rate adaptation out to the end hosts. This was not originally the case: in the early Internet each application was allowed to send at its natural rate, expecting that the network could support the offered load. This initially worked well, due to over-provisioning of the network, but in the mid-1980's significant problems were observed as traffic increased, culminating in a period of congestion collapse which made the network all-but useless. Two possible solutions to this problem existed: make applications adapt to the available capacity, or impose some form of admission control to prevent overload. The community opted for the former approach, introducing changes to the TCP/IP protocol[5,6] which have evolved to provide an effective form of congestion control, widely used in today's network.

TCP provides a reliable ordered byte-stream connection abstraction layered above IP using retransmission to recover from lost data. An elaborate congestion control algorithm[7–9] is used to adapt the transmission rate to match available network capacity and to ensure that bandwidth is shared approximately equally between flows.

In outline, the operation of TCP congestion control is as follows. A new connection begins sending at a low rate, increasing its transmission speed as it probes the network's capacity. This is known as the slow start phase of a connection, and is characterised by the exponential increase in transmission speed shown on the left of Figure 2. Slow start ends when the bottleneck link capacity is reached, at which point the sender enters the congestion avoidance phase. In congestion avoidance, the behaviour of a TCP flow can be characterised by a slow linear increase in sending rate to probe for the bottleneck capacity of the network. When the bottleneck capacity is reached a packet loss will occur, forcing TCP to cut its sending rate in half and retransmit the lost packet. This repeats, with a TCP flow continually probing the available network capacity to produce the characteristic

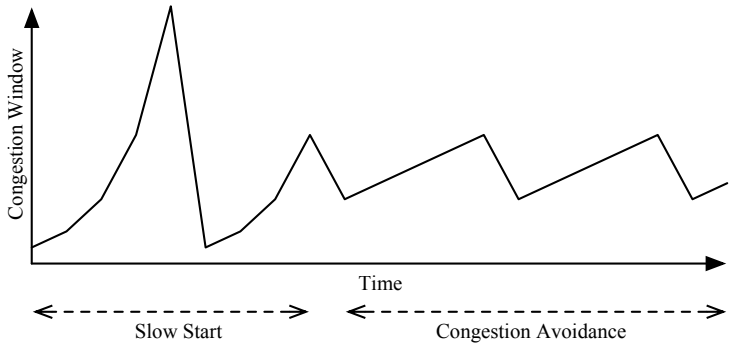

Figure 2. Ideal behaviour of the TCP Congestion Control Algorithm

saw-tooth behaviour of an additive increase multiplicative decrease (AIMD) algorithm, as shown on the right of Figure 2.

The AIMD congestion control of TCP is appropriate for applications that seek to make effective use of network capacity, but do not necessarily require timely delivery. This is because transient congestion causes a multiplicative decrease in sending rate along with a retransmission of the missing data at that lower rate. Furthermore, it stalls the delivery of other data to the application until the gap has been filled and data can be delivered in-order. This works well for data transfer applications since it hides the complexity of the network, abstracting it as a single ordered byte-stream, much like a file provides an abstraction of data on a storage device. The applications neither know, nor particularly care, about the timeliness of the data transfer.

In contrast, real-time multimedia applications can often tolerate some small degree of loss, but require timely delivery of data. For these applications, the delay due to retransmission of lost data is harmful, since it disrupts the timing of a media stream resulting in far worse quality degradation than the original loss. In addition, the rapid changes in sending rate enforced by the AIMD congestion control algorithm of TCP also cause significant disruption to the media, since they translate directly into rapid coding quality fluctuations.

While it is possible to mask these effects to some extent by introducing sufficient buffering at the receiver and by employing rate adaptive coding, this is clearly not an ideal scenario. Furthermore, it is clear that there are difficulties for interactive applications where the total end-to-end delay must be kept small, since these applications cannot afford the time for buffering. Given this, it should come as no surprise to learn that TCP is not widely deployed in interactive real-time multimedia systems.

The predominant transport protocol for interactive real-time multimedia is the Real-time Transport Protocol (RTP).[10] The RTP framework provides framing, sequencing, timing recovery, payload and user identity, and reception quality feedback for real-time applications running over IP networks. The framework is extensible to support a range of media codecs, and to different application domains. To date, RTP has found most favour in the video conferencing and telephony markets, although it is also used in some streaming media applications.

RTP was designed according to the principle of application level framing[11] to expose the details of the underlying network behaviour to the application. This is achieved since RTP typically runs on the UDP/IP datagram service, rather than over a TCP/IP connection, and hence is not required to respond to network events in the same manner as TCP. Because of this, applications using RTP have some freedom to adjust their response to variations in network transit delay, packet loss, and congestion based on their needs, rather than imposing a one size fits all solution on an entire class of applications.[12] For example, while RTP exposes the underlying packet loss behaviour of the network to applications, allowing them to choose how to respond to loss, it doesn't provide the means to recover from that loss. Packet loss recovery is supported by a range of extensions, for example using forward error correction[13] or limited application level retransmission.[14]

A disadvantage of RTP is that applications must implement congestion control themselves, but have traditionally had little incentive to do so. However, as the amount of real-time traffic in the network grows, concern

has been growing that the primitive congestion control algorithms employed in RTP based applications are not sufficient and will lead to a "tragedy of the commons" effect if not improved.[15] This is leading many to consider the issue of standard congestion control for multimedia traffic using RTP, to provide rate adaptation independent of loss recovery.

3. CONGESTION CONTROL FOR MULTIMEDIA TRAFFIC

In the absence of a deployed quality of service solution multimedia flows compete with TCP/IP traffic and, to avoid disrupting the operation of the network, must either use TCP directly or use a transport protocol with a comparable congestion control algorithm. As described previously, TCP is generally not appropriate for real-time traffic, so it is desirable to define alternatives to that can be used for video communication. Since RTP is the dominant protocol for real-time traffic, it is desirable that these alternatives integrate with the RTP framework.

There have been many proposals for new congestion control algorithms, but perhaps the most mature is the TCP-Friendly Rate Control (TFRC) algorithm, a rate based solution that attempts to provide a smoother transmission rate than TCP/IP, but with similar average throughput.[16] TFRC relies on a model of TCP throughout[17] which shows that, for a saturated steady state TCP sender, throughput is proportional to inverse of the square root of the packet loss rate, p. This is known as the TCP friendly equation, and it provides an upper bound on the steady state throughout T, for packet size S, round trip time R, retransmission timeout $t_{RTO} \approx 4R$ and the steady state loss event rate p, such that:

$$T = \frac{S}{R\sqrt{\frac{2p}{3}} + t_{RTO}(3\sqrt{\frac{3p}{8}})p(1+32p^2)} \tag{1}$$

The TFRC algorithm regulates an application's transmission rate according to equation 1 to guarantee that the transmission is TCP-friendly. A connection begins with an initial slow start period, much like a TCP connection, then moves into a steady state phase where the sender measures the loss event rate p and the round trip time R, computes its fair share of bandwidth according to the equation, and adjusts its sending rate to match by varying the inter-packet spacing. Damping is applied, which ensures that the rate of adaptation is smoother than TCP, while maintaining long-term fairness. Since the rate is generally set directly to the desired throughout, rather than following the AIMD dynamics of TCP, it is expected that a TFRC flow will achieve average throughput that is consistant with that of a TCP flow over the same path, but without the short-term variations that so affect multimedia performance.

TFRC is an abstract algorithm, that needs to be instantiated within a particular protocol before it can be used. Two such mappings are under development: the RTP profile for TFRC and the Datagram Congestion Control Protocol. The RTP profile for TFRC[18] defines a mapping of TFRC onto the RTP framework. It specifies extensions to the RTP packet headers and reception quality feedback reports to convey the information required to use the TFRC algorithm, and explains how a sender can use this information to adapt its transmission according to the dictates of congestion control. Like standard RTP, this profile is expected to be implemented as part of an application, rather than in the operating system kernel. This gives great flexibility in how the application responds to congestion signals: it can adjust its transmission to match the TCP-friendly rate in a manner that is appropriate to the application, taking into account the parameters of the media codec, human perception, etc. This can be viewed as an extension of the end-to-end[1] and application level framing[11] arguments that underpin RTP, applied to the domain of congestion control.

The Datagram Congestion Control Protocol (DCCP)[19, 20] provides a congestion controlled but unreliable alternative to TCP and UDP that is intended to be well suited to the needs to multimedia applications. DCCP operates as a replacement to TCP or UDP, running directly over IP to provide congestion control without enforcing reliability (DCCP has been described as "TCP minus bytestream semantics and reliability, or as UDP plus congestion control, handshakes, and acknowledgements"[19]). DCCP is expected to be implemented as part of an operating system kernel, as a generally useful transport protocol. Applications could then run RTP over DCCP/IP, instead of over UDP/IP, and would benefit from the congestion control provided by DCCP. An interesting feature of DCCP is that it allows the congestion control algorithm to be negotiated at connection

set up time; the list of available algorithms is extensible, with the current draft specifying TFRC as one of the alternatives, the other directly mimicing the TCP response function. The standard API for DCCP has not yet been defined, but it is clear from the specification that it should expose details of the delivery process and available bandwidth to the application. This would allow a multimedia application running on RTP over DCCP to adapt its transmission to match the available network capacity in much the same way as a similar application running the RTP profile for TFRC over UDP/IP.

With the increasing use of non-congestion controlled real-time applications, the Internet standards community has become increasingly concerned about the potential disruption that may occur to other applications and, in the extreme case, about the potential for congestion collapse. With this in mind, it is becoming clear that some form of congestion control will be mandated in future standards for real-time traffic, and will likely be retrofitted onto existing standards. Due to it's relative maturity, TFRC will almost certainly be chosen as the initial standard algorithm for these applications, implemented in one of the forms described above. Because of this, it is of great importance to understand the implications TFRC has for real time video traffic.

4. IMPLICATIONS FOR REAL-TIME VIDEO TRAFFIC

Congestion control places certain constraints on the behaviour of real-time multimedia systems. In particular, it enforces limits on both the degree to which those systems can vary their transmission rate, and the time at which they can vary their rate. These limits apply both to when a system can increase its transmission rate and, less obviously, to when it can reduce its transmission rate.

A system that wishes to conform to the dictates of TFRC congestion control must be willing, and able, to perform an initial slow start, then to send at a slowly varying steady state rate. The slow start algorithm requires the application start sending at a low initial rate of one packet per round trip time, increasing exponentially each round trip period where no loss is reported. Once an initial loss has been reported, the system adapts to the sending rate implied by the reported loss fraction. The sender must use a video codec that is capable of a rapid increase in sending rate from zero, while maintaining reasonable picture quality. The slow start period will last anywhere from tens of milliseconds to seconds depending on network conditions, but the sender does not, and cannot, know its duration (and hence the expected steady state transfer rate) in advance. To make matters worse, slow start is a critical period in terms of user perception, since this is when the intial greeting occurs in conversational applications. It is therefore vital that codecs be developed that can achieve acceptable performance under these constraints, if TFRC is to be used.

Once the steady state has been reached, a TFRC-based application will send at a roughly constant rate, based on the average loss rate observed in the network. It is expected that the network conditions will change relatively slowly – failures are rare, and statistical multiplexing obscures changes due to the varying traffic mix, allowing applications to perform based on the gross statistics of the network traffic. The roughly constant rate of a TFRC flow in steady state is often touted as a benefit of TFRC, compared to the sudden halving in rate experienced by a TCP flow when loss occurs. This is no doubt true, but such behaviour is still problematic since many codecs produce bursty output, sending more data when a scene changes rapidly than when the content is relatively static. To obey the dictates of congestion control these bursts must be smoothed out: a process that often requires buffering at the sender, introducing latency, or affects the media quality since the sender is prohibited from transmitting some required data to bound the rate.

Less obviously, a sender cannot assume that the rate specified by the congestion control algorithm is the maximum rate at which it can send, then send below that rate to be safe. A flow that transmits at below the TCP friendly rate will suffer due to the continual probing for bandwidth that TCP traffic performs (the additive increase part of the sawtooth in Figure 2) which will induce loss on the link. When transmitting at the TCP friendly rate, the loss induced on other traffic counters the loss induced by that other traffic, and flows share the link capacity fairly. If one flow is less aggressive, sending below it's share of the link, then that flow will be subject to loss more than it induces, and will therefore be "beaten down" by the other flows to an unfairly low rate. The implication here is that codecs that reduce their sending rate when there is little new content in the scene will be penalized compared to those which send at a relatively constant rate, irrespective of the stream contents. This is unfortunate, since it provides a disincentive to saving bandwidth.

While the steady state behaviour of TFRC is intended to be relatively stable, this does not mean there are no rate changes. A TFRC sender can be required to smoothly vary its transmission rate to match available capacity of the network. This is done by varying the inter-packet spacing, not the packet size, and can sometimes require quite fine adjustments in rate. This causes two issues: 1) many codecs more easily change the size of the packets they send, rather than sending fixed size packets at variable times; and 2) many codecs have a limited set of rates to which they can adapt. At present there is no known congestion control algorithm that is TCP friendly while varying packet size (with fixed spacing). To adapt according to the TFRC algorithm, codecs must be developed that vary the inter-packet spacing rather than packet size (which, given fixed video frame rates, is clearly a challenge). Codecs must also be implemented in such a way that they can easily adapt across a range of rates with relatively small steps. The more fine grained the possible rate adjustment, the closer to fairness a codec will achieve.

Finally, one must consider perceptual effects of changing the transmission rate. It is well known that variations in video quality are detrimental to the user experience, and while the relatively smooth rate provided by TFRC is likely easier for a codec designer than the sawtooth of TCP, this does not mean that user perception issues can be ignored. Many of the rate changes described previously have an affect on perceived quality, and it is unclear if a TFRC flow – in the event that a codec can be found that can meet the constraints of the congestion control algorithm – will be usable from a human factors viewpoint.

This does not mean all is lost. Despite these difficulties, there is one clear advantage to implementing congestion control: the application gets to choose which data is discarded. If congestion control is not used, the application will send at its natural rate, and the network will arbitrarily discard data to meet the available capacity constraints. With congestion control, the network feeds back to the application information on the available rate, and the application decides how to deduce its sending rate to meet that target. An intelligent application can adjust the encoding parameters to meet the target rate while maintaining better video quality than would be achieved with arbitrary data drop.

5. FUTURE DIRECTIONS

It is clear that there is a disconnect between the network transport protocol community and the video coding community. The current congestion control algorithms, TCP and TFRC, both place an undue burden on the codec designer, yet TFRC is considered by many in the network community to be a poor protection for the network. How might video codecs and congestion control algorithms evolve in future?

There are some constraints apparent in the design of congestion control algorithms that are due to limited capacity of the network. Certain other constraints have been imposed due to the desire to be fair with competing traffic, and there is clearly some potential to trade this fairness off for improved video transport performance, if the appropriate changes to the protocols can be identified and if the benefits of doing so outweigh the disadvantages.

The two hard constraints are that a codec cannot send at a higher average rate than those dictated by the congestion control algorithm, and that it must implement some form of slow start. The first is easy to justify: the network has limited capacity due to physical constraints, and the primary purpose of congestion control is to prevent flows overloading the network. Applications that send at a higher average rate than that specified by the rate control algorithm must be disallowed since they run the risk of exceeding the capacity of the network, and potentially causing congestion collapse. This does not imply that instantaneously exceeding the suggested average rate will necessarily overload the network, since there is buffering capacity in the network which can absorb tempory overloads. This buffering is limited in size, however, and a system cannot routinely overload the link capacity without running the risk of causing congestion. A fruitful area of research may be to study how the buffering capacity of the network can be estimated, to allow codecs to burst above the average rate for brief periods (this may help solve the problems due to codecs that send a higher rate burst of data after a scene change, for example).

Some form of slow start is also an absolute requirement, since a new sender cannot know the available capacity of a network path in advance. Unless the network is evolved to support resource reservation, it will be necessary to probe for the available capacity, rather than simply starting sending with the hope that the network can

support that capacity. Codecs must be developed that can support slow start, or it must be accepted that a transport protocol will send dummy data during the call setup period, to probe for capacity.

Less strict are the requirements that the tranmission rate be smoothly and continuously variable, and that the transmission rate can be adjusted by changing the inter-packet spacing. These are clearly desirable features, which make the it easier to design and operate networks, but they are not required for safe operation of the network. At present, though, there are no congestion control algorithms that change these parameters yet remain TCP friendly.

Likewise, TCP friendliness is not necessarily a goal. Provided a transport is congestion controlled and competes well with itself, it will not disrupt the operation of the network if it is not TCP friendly. This does not mean that it won't cause problems, since there will certainly be disruption to TCP traffic, but that is only problematic if there is a large volume of TCP traffic on the network. In a relatively constrained environment, where the multimedia traffic is considered more important than data traffic, this may be an appropriate tradeoff (for example, within a television distribution network or other environment purpose built to transport video traffic). It is possible that video congestion control becomes easier if the requirement for TCP friendly behaviour is removed.

Finally, it has frequently been proposed to include some form of resource reservation into the network. This would allow multimedia traffic to be segregated from other traffic on the network, reserving capacity for the multimedia flow to be run without congestion control, and allowing other traffic to fill the remainder of the network. There has been extensive work on this topics for many years, yet resource reservation has not seen widespread deployment. It is the author's belief that this is due primarily to economic issues: resource reservation is only useful if the network has insufficient capacity for all the flows which it must support, as a way to prioritise certain traffic. It practice, network operators have found it cheaper to buy more capacity than to implement resource reservation and accounting. It is not clear that the economics will shift for wired network in the medium term future, so congestion control – rather than resource reservation – can be expected to continue to be an important topic.

6. CONCLUSIONS

This paper has surveyed requirements for congestion control of real-time video traffic in the Internet. There has been considerable work in this area in recent years, with the realisation that TCP does not provide an appropriate transport for this class of traffic, and with the development of new transport protocols that claim to provide a better, and congestion controlled, basis for future deployment. These new developments have been outlined, and the constraints they impose on codec designers have been noted.

This area is not yet well defined, and it is not clear that the existing transport protocols meet the needs of the video coding community, or that the video coding community can produce codecs that meet the needs of the Internet community. The need for congestion control is well established: the challenge in the coming years it to develop workable algorithms that suit both the networking and video coding communities.

7. ACKNOWLEDGEMENTS

This work was supported by the US National Science Foundation under grant No. 0230738, and by the UK Department of Trade and Industry via a GridNet award administered by the National e-Science Centre.

REFERENCES

1. J. H. Saltzer, D. P. Reed, and D. D. Clark, "End-to-end arguments in system design," *ACM Transactions on Computer Systems* **2**, November 1984.
2. C. S. Perkins, L. Gharai, T. Lehman, and A. Mankin, "Experiments with delivery of HDTV over IP networks," in *Proceedings of the 12th International Packet Video Workshop*, (Pittsburgh, PA, USA), April 2002.
3. L. Gharai, C. S. Perkins, and T. Lehman, "Packet reordering, high speed networks and transport protocol performance," in *Proceedings of the 13th International Conference on Computer Communications and Networks (ICCCN'04)*, (Chicago, IL, USA), October 2004.

4. C. Fraleigh, S. Moon, B. Lyles, C. Cotton, M. Khan, D. Moll, R. Rockell, T. Seely, and C. Diot, "Packet-level traffic measurement from the Sprint IP backbone," *IEEE Network* **17**, November 2003.
5. J. Nagle, "Congeston control in IP/TCP internetworks." Network Working Group, January 1984. RFC 896.
6. V. Jacobson, "Congestion avoidance and control," in *Proceedings of ACM SIGCOMM '88*, pp. 314–329, (Stanford, CA, USA), August 1988.
7. M. Allman, V. Paxson, and W. Stevens, "TCP congestion control." Internet Engineering Task Force, April 1999. RFC 2581.
8. S. Floyd, J. Mahdavi, M. Mathis, and M. Podolsky, "An extension to the selective acknowledgement (SACK) option for TCP." Internet Engineering Task Force, July 2000. RFC 2883.
9. S. Floyd, T. Henderson, and A. Gurtov, "The NewReno modification to TCP's fast recovery algorithm." Internet Engineering Task Force, April 2004. RFC 3782.
10. H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A transport protocol for real-time applications." Internet Engineering Task Force, July 2003. RFC 3550.
11. D. D. Clark and D. L. Tennenhouse, "Architectural considerations for a new generation of protocols," in *Proceedings of ACM SIGCOMM '90*, pp. 200–208, (Philadelphia, PA, USA), September 1990.
12. M. Handley and C. S. Perkins, "Guidelines for writers of RTP payload format specifications." Internet Engineering Task Force, December 1999. RFC 2736.
13. J. Rosenberg and H. Schulzrinne, "An RTP payload format for generic forward error correction." Internet Engineering Task Force, December 1999. RFC 2733.
14. J. Ott, S. Wenger, N. Sato, C. Burmeister, and J. Rey, "Extended RTP profile for RTCP-based feedback (RTP/AVPF)." Internet Engineering Task Force, August 2004. Work in progress (draft-ietf-avt-rtcp-feedback-11.txt).
15. S. Floyd and J. Kempf, "IAB concerns regarding congestion control for voice traffic in the internet." Internet Engineering Task Force, March 2004. RFC 3714.
16. S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-based congestion control for unicast applications," in *Proceedings of ACM SIGCOMM 2000*, (Stockholm, Sweden), August 2000.
17. J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP throughput: A simple model and its empirical validation," *ACM Computer Communication Review* **28**, pp. 303–314, 1998 September.
18. L. Gharai, "RTP profile for TCP friendly rate control." Internet Engineering Task Force, October 2004. Work in progress (draft-ietf-avt-tfrc-profile-03.txt).
19. E. Kohler, M. Handley, and S. Floyd, "Datagram congestion control protocol (DCCP)." Internet Engineering Task Force, November 2004. Work in progess (draft-ietf-dccp-spec-09.txt).
20. E. Kohler, M. Handley, and S. Floyd, "Designing DCCP: Congestion control without reliability," tech. rep., International Computer Science Institute, Berkeley, CA, USA, May 2003.