University of Helsinki
Department of Computer Science
Series of Publications C, No. C-2010-31

Rate Control State-of-the-art Survey

L. Hyttinen, L. Daniel, I. Järvinen and M. Kojo

Helsinki, May 2010

Technical Report C-2010-31

University of Helsinki
Department of Computer Science
P. O. Box 68 (Gustaf Hällströmin katu 2b)
FIN-00014 University of Helsinki, Finland

Rate Control State-of-the-art Survey

L. Hyttinen, L. Daniel, I. Järvinen and M. Kojo
Department of Computer Science, University of Helsinki
Technical Report C-2010-31
May 2010
26 pages

Abstract. The majority of Internet traffic use Transmission Control Protocol (TCP) as the transport level protocol. It provides a reliable ordered byte stream for the applications. However, applications such as live video streaming place an emphasis on timeliness over reliability. Also a smooth sending rate can be desirable over sharp changes in the sending rate. For these applications TCP is not necessarily suitable. Rate control attempts to address the demands of these applications. An important design feature in all rate control mechanisms is TCP friendliness. We should not negatively impact TCP performance since it is still the dominant protocol.

Rate Control mechanisms are classified into two different mechanisms: window-based mechanisms and rate-based mechanisms. Window-based mechanisms increase their sending rate after a successful transfer of a window of packets similar to TCP. They typically decrease their sending rate sharply after a packet loss. Rate-based solutions control their sending rate in some other way. A large subset of rate-based solutions are called equation-based solutions. Equation-based solutions have a control equation which provides an allowed sending rate. Typically these rate-based solutions react slower to both packet losses and increases in available bandwidth making their sending rate smoother than that of window-based solutions. This report contains a survey of rate control mechanisms and a discussion of their relative strengths and weaknesses. A section is dedicated to a discussion on the enhancements in wireless environments.

Another topic in the report is bandwidth estimation. Bandwidth estimation is divided into capacity estimation and available bandwidth estimation. We describe techniques that enable the calculation of a fair sending rate that can be used to create novel rate control mechanisms.

Key Words: Rate Control, Bandwidth Estimation, Survey

Contents

1 Introduction

Majority of Internet traffic such as email, Web, file transfer, P2P file sharing and media streaming use Transmission Control Protocol (TCP) as the transport protocol [Pos81]. TCP provides reliable ordered delivery of a stream of packets between the two end systems. Future mobile Internet is a difficult environment for traditional reliable transport protocols since it introduces new traffic types, which have different latency and throughput needs compared to traditional bulk transfer. More users want to enjoy streaming video and audio while on the move. New mechanisms and protocols have been developed to operate in this environment. While TCP is no longer adequate for these requirements, the new methods should not negatively influence TCP performance when competing for network resources as TCP is still the dominant transport protocol.

There are two main approaches into providing multimedia streaming for clients: window-based and rate-based rate control. Window-based rate control increases its sending rate as a result of the successful transmission of a window of packets, and decreases the sending rate upon the detection of a packet loss event. Rate-based rate control attempts to smooth the transmission rate by estimating the available bandwidth in some other way. A large subset of rate-based control is called equation-based rate control. In this scheme the sending rate is controlled by an equation which estimates the allowed sending rate based on feedback from the receiver. The tradeoff usually is that rate-based mechanisms are less aggressive in increasing their sending rate, but in case of packet losses they lower their sending rate slower than window-based approaches. For that reason they are often called slowly-responsive algorithms.

Various approaches have been taken to create a mechanism or a protocol, for providing a smooth transmission rate which multimedia streams require. Early work into providing multimedia streaming in wireline networks include Rate Adaptation Protocol (RAP) [RHE99] and Streaming Control Protocol (SCP) [CPW97]. They both behave similarly to TCP as they implement an Additive Increase Multiplicative Decrease (AIMD) algorithm. In response to a packet loss both halve their sending rate since both treat a packet loss as a sign of congestion.

Rate-based approaches attempt to remove abrupt changes in the sending rate whilst still behaving as a good network citizen ie. not negatively affecting TCP performance. In equation-based control, an equation is used to limit the sender transmission rate. TCP-Friendly Rate Control (TFRC) [FHPW00] and Smooth and Fast Rate Mechanism (SFRAM) [KKK04] both use the TCP-throughput model [PFTK98] to adjust their transmission rate. In this category TFRC has become the dominant mechanism/protocol in literature. TCP-Friendly Window-based Control (TFWC) uses the same TCP throughput model as TFRC and SFRAM but introduces a TCP-like ACK mechanism [CH09].

All of the previous mechanisms treat packet loss as an indication of congestion but with wireless networks, error related packet losses become more common. New approaches are therefore warranted. Some approaches attempt to improve TFRC's performance in wireless networks. Examples include Multi-TFRC (MULTFRC) [CZ06], All-In-One-TFRC (AIO-TFRC) [CZ05], Loss Event Rate Discounting (TFRC-LERD) [TGH04] and TFRC-wireless [CCV03]. On the other hand Analytical Rate Control (ARC) rejects the TCP throughput model as ill-suited for wireless environments and instead creates a new control equation [AA04]. Both ARC and Rate Control Scheme (RCS) [TMAJ01] use probe/dummy packets to detect available bandwidth. Some approaches such as Video Transport Protocol (VTP) opt for a totally new window-based mechanism for real-time multimedia streaming [YSG+06].

Some protocols are designed to be multimedia streaming friendly from the start and these include Datagram Congestion Control Protocol (DCCP) [KHF06a], Real-Time Protocol/RTCP [SCFJ03] and SCTP [Ste07].

After a suitable sending rate has been calculated or estimated, the multimedia stream is often adapted to the network conditions. This is called rate adaptation. Several approaches to rate adaptation vary from one static stream to varying the bit stream on the fly.

Estimates of the available bandwidth and the capacity of the network path could be used to enhance rate control schemes. With this information the sender can more accurately increase or decrease its sending rate.

The rest of the review is organized as follows: Section 2 presents window-based rate control mechanisms. Section 3 presents rate-based rate control mechanisms. Section 4 presents several issues with rate control mechanisms. Section 5 provides improvements to both window-based and equation-based rate control mechanisms in wireless environments. Section 6 discusses various protocols which attempt to be rate control friendly. Section 7 has a discussion on adapting the stream quality in response to available bandwidth. Section 8 presents the basics of capacity and available bandwidth estimation. Section 9 concludes the paper.

2 Window-based

As a large part of the Internet traffic is TCP traffic, most new mechanisms or protocols that have been developed attempt to be TCP-Friendly ie. not adversely affect TCP performance. TCP uses the additive increase multiplicative decrease algorithm (AIMD) to manage its window size. The congestion window is increased upon a successful transmission of a window of packets, and decreased upon a detection of a packet loss. These are the rules AIMD as follows:

$$Increase: w_{t+1} = w_t + \alpha, \alpha > 0,$$

$$Decrease: w_{t+1} = w_t - \beta w_t, 0 < \beta < 1$$

In the equations w_t stands for window size at time t whereas α and β are constants (in TCP Reno $\alpha = 1$ and $\beta = 0.5$). Since following this equation leads to halving the window size in case of a single packet loss other algorithms, called binomial algorithms, have been proposed to be used instead [BB01]. They generalize AIMD and use the following control rules:

$$Increase: w_{t+1} = w_t + \frac{\alpha}{{w_t}^k}, \alpha > 0,$$

$$Decrease: w_{t+1} = w_t - \beta {w_t}^l, 0 < \beta < 1$$

If $k = 0$ and $l = 1$ these equations are reduced to AIMD rules. However varying the values of k and l will result in the so-called binomial algorithms. The binomial algorithms are TCP-friendly if and only if $k + l = 1$ and $l \leq 1$, for suitable value of α and β [BBFS01]. Two such binomial algorithms were tested, both of which attempt to improve window-based rate control in providing streaming media. They are called Inverse Increase Additive Decrease (IIAD) and SQRT [BB01]. For IIAD the values are $k = 1$ and $l = 0$ and for SQRT the values are $k = l = 0.5$ respectively. Both IIAD and SQRT react to congestion less drastically and increase their transmission rate more slowly than AIMD (and TCP) and therefore should provide a smoother transmission rate for multimedia streams.

Another approach is to vary the values of α and β. General additive increase multiplicative decrease (GAIMD) does just that [YL00]. Choosing the values $\alpha = 0.31$ and $\beta = 7/8$ leads to GAIMD flows being TCP-friendly and to improved smoothness compared to TCP flows [YL00].

Streaming Control Protocol (SCP) behaves similarly to TCP during the start-up phase, by using a TCP-style slow start policy in discovering the available bandwidth. However when TCP switches to the congestion avoidance phase, SCP begins to use a combined rate- and window-based flow control policy which maintains smooth streaming with low latency. It tries to achieve this by maintaining an appropriate amount of buffering in the network for sufficient utilization of the available bandwidth. When losses do occur, SCP considers them as congestion related and halves its congestion window size.

3 Rate-Based

Since traditional window-based approaches typically halve the sending rate in case of congestion, the effect for real-time streams is quite severe. A solution to this is called a rate-based rate control. Instead of the window-based transmission model the sending rate is controlled by an estimation of the available bandwidth. By using the estimation the sender attempts to keep the sending rate steadier. The tradeoff is that the sender does not increase its sending rate as aggressively as TCP.

An early work called Rate Adaptation Protocol (RAP) implements the additive increase multiplicative decrease (AIMD) algorithm but abandons the window-based congestion control. In RAP if no losses are detected, the transmission rate is periodically increased and when losses are detected the transmission rate is immediately reduced in a dramatic fashion. Since it does not use the window model it determines a suitable rate according to its algorithm. It does use the acknowledgments to calculate its sending rate but the data transmissions are not directly tied to arriving acknowledgments. To safeguard against overshooting, RAP will stop sending if no acknowledgments arrive in a certain period of time. Since RAP was designed for wired environments it considers all losses as a sign of congestion and reduces its sending rate accordingly.

Equation-based rate control uses a control equation which provides the maximum sending rate. The sender adapts its sending rate according to the equation. The primary goal of equation-based rate control is to avoid aggressively finding and using available bandwidth and instead maintain a steady sending rate while still being responsive to congestion [FHPW00]. Equation-based rate control has some design principles in contrast to traditional TCP behavior [FHPW00]:

- Avoid aggressively increasing the sending rate but increase it slowly in response to a decrease in the loss event rate.

- Halve the sending rate only in response to several successive loss events.

For unicast traffic for which the equation-based TCP-Friendly Rate Control (TFRC) was developed it is also required that:

- Receiver reports feedback to the sender at least once per round-trip time if it has received packets in that interval.
- Sender will reduce its sending rate in case it has not received feedback after several round-trip times. Ultimately it will stop sending altogether.

Adhering to these policies results in a mechanism that provides smoother throughput for multimedia applications in wired networks. For wireless networks the mechanism is not ideal as it considers packet loss as a sign of congestion and not due to errors in the wireless channel.

For the control equation, TFRC uses the refined TCP throughput model [PFTK98]:

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

where T stands for the upper bound of the sending rate in bytes/sec, s is the packet size, R is the round-trip time, p is the steady-state loss event rate and t_{RTO} is the TCP retransmit timeout value. A loss event rate is not the same thing as packet loss rate; a loss event can contain several packets lost in a single round-trip time. The loss event rate is calculated at the receiver end and sent back to the sender. Then the sender will calculate a smoothed loss event rate to be used in the equation. This value is smoothed with a method called Average Loss Interval Method. This smoothing is done in order to avoid reacting violently to a single unrepresentative loss event rate reported by the receiver. The method computes a weighted average of the loss rate over the last n loss intervals, with equal weights on each of the most recent n/2 intervals. The rest of the intervals are given lower weights. After testing 8 was deemed a sufficient value for n in order to achieve a smooth loss event rate but still be responsive enough when loss event rates are changing. All the other factors in the equation can be measured at the sender side.

TFRC has emerged as the standard mechanism to provide smooth and predictable throughput for multimedia applications. The latest version is described in RFC 5348 [FHPW08]. Datagram Congestion Control Protocol (DCCP) has adopted TFRC as one of its congestion control mechanisms [KHF06a, KPF06].

Another mechanism using the the same TCP throughput model as TFRC is the Smooth and Fast Rate Adaptation Mechanism (SFRAM). There are two main differences between TFRC and SFRAM [KKK04]. First of all SFRAM uses UDP for the data channel and TCP for the feedback channel. Secondly the receiver performs the rate estimation calculations and reports this to the sender. This removes some burden on the sender and improves multicast performance. SFRAM can also be coupled with network-aware error control (NAEC) which allows the video coder to select a proper error-control scheme but this is beyond the scope of this review [KKK04].

TCP Emulation At the Receivers (TEAR) shifts the sending rate decision making to the receiver [ROY00]. TEAR determines the appropriate sending rate based on congestion signals observed at the receiver. This information includes packet arrival times, packet losses and timeouts. Based on the information a TEAR receiver will calculate a TCP-friendly sending rate. This calculated sending rate is smoothed and sent to the sender. Sending rate updates are sent less often than TCP acknowledgments which improves TEAR performance in multicast.

4 Issues with Rate Control

Four properties of a TCP-friendly protocol are very important when comparing it to the traditional TCP: fairness, smoothness, responsiveness and aggressiveness. A study comparing TFRC, TEAR and GAIMD with TCP showed several drawbacks with these protocols [YKL03]. Three changes were introduced into the target network to compare the protocols. The first change is the inherent network fluctuations in a stationary network environment. The other two environment changes are an increase in network congestion and a step increase of available bandwidth. The study shows that for all four protocols the smoothness and fairness become worse as loss rate increases. However the rate of degredation is different for each. TFRC and TEAR have smoother sending rates than TCP and GAIMD, but with high loss rate both have undesirable performance: TFRC sending rate drops to almost zero and TEAR sending rate becomes too high. With higher congestion TCP remains the most responsive but can overshoot which causes it recover from that state. TEAR reacts the worst to high congestion, unless feedback is able to reach the sender it will not lower its sending rate and worsen the situation. The most aggressive protocol of the four is TCP which causes it to overshoot again. TFRC and GAIMD have similar aggressiveness whereas TEAR is the least aggressive of the four protocols.

Another study on fairness, aggressiveness and responsiveness of various TCP-friendly congestion control protocols introduces a new criterion for comparing the schemes: TCP-equal share [TLL07]. The study compares the various schemes based on TCP-compatibility, TCP-equivalence and TCP-equal share. TCP-compatibility means that a TCP-compatible flow must not use more bandwidth than TCP in steady state in comparable conditions. TCP-equivalence means that a TCP-equivalent flow has the same bandwidth as a TCP flow in identical network conditions. TCP-equal share focuses on flows inhabiting the same bottleneck. Both flows (TCP and TCP-equal share) should be able to coexist and share bandwidth in the same bottleneck. The results of the tests showed that TFRC performs well in most categories but lacks aggressiveness [TLL07]. GAIMD and other window-based schemes had issues with fairness while TEAR was again unaggressive.

In an effort to improve long term throughput and slow responsiveness a new adaption of the TFRC protocol has been designed. Adaptive TCP Friendly Rate Control (ATFRC) uses two throughput models in order to calculate the sending rate [CWL03]. In a period without timeouts, but using only triple duplicate ACKs, a model is used which reflects TCP performance in such a situation:

$$R = \frac{s}{RTT}\sqrt{\frac{3}{2bp}}$$

where R is the sending rate, s is the average packet size, p is the static loss rate and b the number of ACKed packets when TCP uses delayed acknowledgments. After a timeout, the sending rate is halved and the next sending rate is calculated using the original TCP throughput model. This model is used until τ triple duplicate ACK events occur. After the τth event occurs the sending rate calculation is switched back to the new model. Results of tests showed that this approach improved the responsiveness of TFRC when more bandwidth becomes suddenly available [CWL03].

Several drawbacks have been presented in regard to TFRC and rate-based protocols in general. A study was performed to see how feedback delays affect window-based and rate-based protocols [ZL04]. In particular the feedback delays introduced by buffering delays in the bottleneck routers. The results show that window-based protocols have less packet losses when the feedback delays grow larger. However, they also exhibited large oscillations when the buffering delay grew. The amount of data lost during each overshoot by a window-based protocol is proportional to the square root of the buffering delay $\sqrt{D}$ [ZL04]. With TFRC the amount of lost data is D^2.

Another paper studied the behavior of slowly-responsive congestion control algorithms, such as TFRC and RAP, in dynamic conditions [BBFS01]. Its findings show that, since the slowly-responsive algorithms lack the self-clocking mechanism present in TCP, they can overwhelm the link and can cause massive amounts of packet losses when an abrupt drop in the available bandwidth occurs. To improve TFRC performance in such networks, a conservative mode is added to TFRC which limits the sending rate of TFRC in case of a packet loss. This is called TFRC with self clocking [BBFS01]. Another issue arising from slow responsiveness is that when the available bandwidth is suddenly increased, faster protocols such as TCP will hog the available bandwidth causing unfairness between TCP and TFRC [BBFS01]. TFRC performed better than the other tested protocols (RAP and a binomial algorithm), but its performance depended on the packet loss patterns [BBFS01].

The media-friendliness of TFRC has been called into question in a study [WBJ04]. A protocol is deemed to be media-friendly if it considers the characteristics of the streaming media and it provides streaming media with an uninterrupted transport service. There is an inherent tradeoff between media-friendly and TCP-friendly. While TCP-friendliness might require the sender to reduce its sending rate at the sign of congestion, in order to remain friendly to the streamed media the rate should be kept steady. The study showed that TFRC failed to prevent a reduction in the sending rate during transient workload increases and TFRC had some fairness issues when competing with TCP. In particular in the case when a TFRC sender is sending at a constant data rate less than its fair share of the available bandwidth. If the competing traffic is a TCP-FTP sender, it would utilize the rest of the bandwidth. Upon experiencing congestion both senders would lower their sending rate since congestion control principles demand this. This happens even if the TFRC sender was sending less than its fair share to begin with.

Three main factors determine TFRC throughput: TCP friendly equation, loss event rate estimation and network delays (including RTO estimation). A study shows that when competing TCP and TFRC flows on the same bottleneck have different sending rates, their observed loss event rates can also be significantly different [RX05]. These different loss event rates can greatly amplify the initial throughput difference. Three reasons, which explain why TCP and TFRC would have different average sending rates, have been discovered. The first is the excessive friendliness of TFRC in comparison to TCP caused by the convexity of the TFRC equation [VB05]. The second reason is the convexity of $1/x$, where x is a loss event period [RX05]. The third reason are the different methods of estimating RTO in TCP and TFRC [RX05]. These reasons cause a TFRC sender, in some extreme simulated cases, to use 20 times more bandwidth than TCP or at times 10 times less bandwidth than TCP.

Another issue with TFRC deals with flows traversing typical DSL links with drop-tail queuing. In that case TFRC can starve TCP traffic. This is because although on average TCP and TFRC respond similarly to loss, TFRC lacks the fine-grained congestion avoidance mechanism which TCP's ack-clocking provides. This leads to TFRC sometimes overshooting the available link capacity, filling the buffer at the bottleneck link and causing TCP's ack-clock to reduce the sending rate. Another problem with TFRC is the potential oscillatory behavior due to the above-mentioned overshoot. To guard against this TFRC uses a short-term mechanism to reduce the transmission rate as the RTT increases.

To solve the issues stemming from DSL-links, a new mechanism has been proposed called TCP-Friendly Window-based Control (TFWC) [CH09]. It uses the same throughput model as TFRC but has modified it to fit a window-based mechanism. Basically it is the same equation but with t_{rtt}/s multiplied on both sides.

$$W = \frac{1}{\sqrt{\frac{2p}{3}} + \left(12\sqrt{\frac{3p}{8}}\right)p(1+32p^2)}$$

where W stands for window size in packets and p is the loss event rate. Simulation results show this approach to behave more fairly than TFRC in a typical DSL-like environment (low statistically multiplexed network with drop-tail queuing) [CH09].

Another problem with basic TFRC occurs when a low-bandwidth, small packet TFRC sender shares a bottleneck with a high-bandwidth, large-packet TCP sender [FK07]. The TFRC sender might be forced to slow down, even if the TFRC application's nominal rate in bytes per second is less than achieved by the TCP flow. A variant of TFRC aimed to improve TFRC performance in such an environment is called TFRC-Small Packets (TFRC-SP) [FK07]. A TFRC-SP sender calculates its fair rate as if the bottleneck restriction was in bytes per second instead of packets per second.

The TCP throughput equation used by TFRC suffers from the same problems as traditional TCP in high-speed long distance networks. Several variants of TCP have been proposed to improve its performance in such networks, such as HSTCP [Flo03] and BICTCP [XHR04]. High-speed equation based rate control (HERC) attempts to bring the same benefits to TFRC as HSTCP brings to TCP [Xu07]. HERC uses the sending rate function of HSTCP when the loss event rate is lower than 10^{-3}. Otherwise is uses the same equation as TFRC.

$$X_{HERC} = \begin{cases} X_{HSTCP} & \text{if } p < 10^{-3} \\ X_{TCP} & \text{otherwise} \end{cases}$$

The tests run by the designers of the mechanism showed that HERC was not as smooth as TFRC. In order to improve smoothness the loss event history was increased to 22 from the 8 used by TFRC originally.

The TCP-friendliness of TFRC is defined as deterministic TCP friendliness (DTF), which means that it achieves the same or lower average sending rate as TCP on a short time scale [FX08]. However on a long time scale TFRC flows can cause TCP response times to grow to the point of system instability [FX08]. For this reason another class of TCP friendliness has been defined; stochastic

TCP friendliness. Stochastic TCP friendliness (STF) considers the impact of UDP traffic on the rate distribution of all TCP flows. Deterministic TCP friendliness considers the specific impact of UDP traffic on the rate of each TCP flow. TCP-friendly CBR-Like Rate Control (TFCBR) aims to be stochastically friendly to other TCP flows while maintaining a steady sending rate [FX08]. TFCBR has been implemented on top of TFRC with the same slow start phase. However in congestion avoidance TFCBR maintains all the TCP rates estimated by TFRC in the past β seconds. In this way TFCBR will remain TCP friendly on the long term. The drawback of keeping more past data is greater resource consumption but otherwise it is quite simple to implement.

5 Wireless Enhancements

Since all the above mentioned mechanisms consider a packet loss as a sign of congestion they are not well suited to perform well in the wireless networks. Non-congestion related packet loss rates arise considerably in such networks and this leads to the mechanisms lowering their sending rate needlessly.

5.1 Window-based

An early approach into providing rate control in wireless networks involved using the traditional AIMD algorithm. This rate control mechanism is called Rate Control Scheme (RCS) [TMAJ01]. RCS runs over RTP/RTCP [SCFJ03] and UDP [Pos80]. A key element in RCS is the use of low priority dummy packets. RCS therefore relies on routers providing some priority discipline. This approach may cause problems with routers without priority policy. Also applications generating low priority traffic can be penalized by the dummy packets. To mitigate the effects of the dummy packets for such applications the dummy packets are sent only at the beginning of the connection and when packet loss is detected. They are used to probe the network for available resources. So when a data packet is lost, RCS halves its sending rate similar to TCP. At this point RCS sender starts sending 2 dummy packets for each data packet. If the loss was due to congestion these dummy packets will be lost and recovery continues as in traditional TCP. However if the packet loss was due to channel error, the dummy packet acknowledgments will arrive at the RCS sender and it will allow it to increase its transmission rate faster than traditional TCP.

Video Transport Protocol (VTP) is a window-based approach of which key features are a loss discrimination algorithm (LDA) and an achieved rate (AR) estimation [YSG+06]. Achieved rate is defined as the rate that the sender has successfully pushed through a bottleneck router on the path. This is the rate that the receiver has measured, plus the fraction corresponding to packet loss at the exit of the bottleneck router due to random errors. This measurement is sent back to the sender which updates its AR value with an Exponential Weighted Moving Average (EWMA). If packets were lost, VTP will use a loss discrimination algorithm to verify whether the losses were congestion or error related. The LDA chosen by VTP is a variant of Spike, which keeps track of RTT values [TTM+00]. If the current RTT is close to the minimum RTT, the network is unlikely to be congested. If packets are estimated to be lost due to wireless error, they are not used to calculate the AR. When packets are estimated to be lost due to congestion, VTP will decrease its sending rate, less than TCP would. In order to remain TCP friendly this decrease in sending rate remains

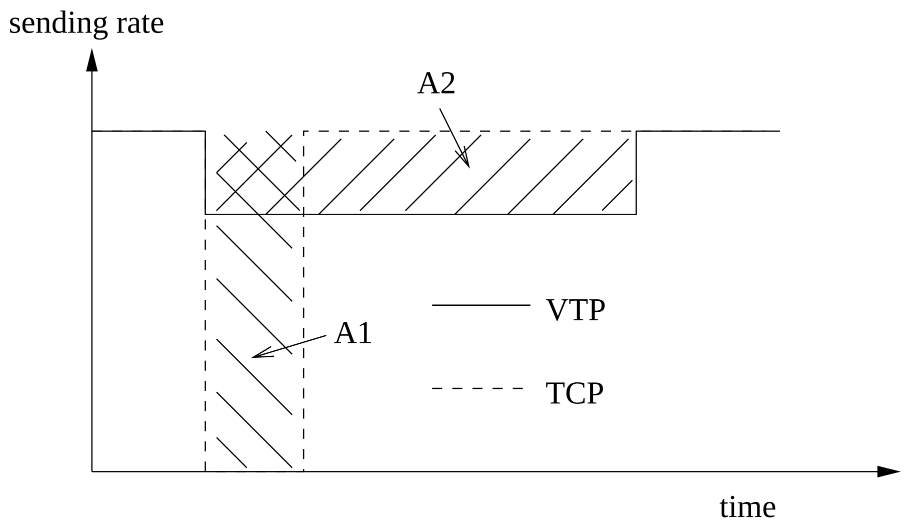

Figure 1: Comparison of rate control between TCP and VTP

longer than it would in TCP. This is called opportunistic friendliness [YSG$^+$06]. This concept is demostrated in Figure 1. In the event of packet loss TCP drastically lowers its sending rate but recovers quickly. VTP on the other hand lowers its sending rate by a smaller portion but keeps its sending rate lower for a longer period of time. The areas A1 and A2 represent the amount of data TCP and VTP would send if the loss had not occurred. Area A1 equals A2 in order for VTP to remain opportunistically friendly.

5.2 Rate-Based

Since TFRC is the major standard in providing rate control in wired networks, much work has been done to improve its performance in the wireless environment. One approach is to attempt to differentiate between the two different packet losses i.e., packets lost due to congestion and packets lost due to a channel error. TFRC-Wireless is such an approach [CCV03]. In this approach when the receiver detects a packet loss it invokes an end-to-end loss differentiation algorithm (LDA). Several such algorithms have been developed and they use different metrics to evaluate the reasons for packet loss. For example Biaz uses packet inter-arrival times to differentiate losses [BV99] whereas Spike uses relative one-way trip times (ROTT) [TTM$^+$00]. If the algorithm estimates that the packet loss is due to congestion, the packet is normally used by the receiver in the loss event rate calculations. However if the packet was lost due to wireless channel error the receiver does not use it in its calculations. The algorithm would only be used in connections with a wireless link in the path between a sender and a receiver. The algorithm used by TFRC-Wireless is a hybrid scheme of several base algorithms (Biaz, Spike and ZigZag [CCV03]), called ZBS, from which it chooses the best fitting one based on the network conditions.

The above approach tries to differentiate packet losses between congestion and non-congestion related packet losses introduces some computational complexity. As computational resourcers in small handheld devices are limited, an approach which does not attempt to differentiate between packet losses, has been proposed called Loss Event Rate Discounting (TFRC-LERD) [TGH04]. TFRC-LERD uses the same throughput model as normal TFRC with one modification:

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

In the above equation p_d stands for loss event rate p which is dynamically adjusted with a discounting factor d. The discounting factor is a number between 0 and 1. In this approach the loss event rate is reported back to the sender, same as in TFRC, which will then multiply it with the discounting factor. The discounting factor is maintained using an Inverse Increase Additive Decrease algorithm (IIAD) which basically looks at the average round-trip time and if the average round-trip time exceeds a predetermined value the discount level is lowered. If the average round-trip is less than that value then the discount level is increased.

In heterogeneous mobile networks, rate control mechanisms must naturally deal with handovers. Traditional TFRC has major problems when handovers occur [GK04]. TFRC receives only 10–50% throughput if the handover is to a fast link and in the opposite case TFRC can starve normal TCP. Two solutions were proposed and experimented with to improve TFRC performance [GK04]. Either overbuffering is introduced to help TFRC to smoothly change between links with different bandwidth-delay products or an explicit handover notification is used to adjust TFRC feedback reports. With overbuffering the buffer size of all the links are configured to the maximum of the bandwidth-delay product (BDP) of any link in the end-to-end path. Though overbuffering helps TFRC to have a smooth handoff between links of different BDPs, this scheme is not easy to implement as the bandwidth and delay of all the links in the path should be known beforehand. Overbuffering the low BDP links increases the queuing delay which affects interactive applications. Overbuffering also inflates the RTO which delays the RTO recovery.

Another modification to improve TFRC performance with vertical handovers is called Measurement-Based TFRC (MBTFRC) [LKW$^+$03]. MBTFRC uses the same algorithm as TFRC when network conditions change slowly and smoothly. However when sharp changes in network conditions are detected MBTFRC does one of two things:

- MBTFRC enters slow start to probe for available bandwidth, if the bandwidth increase sharply.
- MBTFRC will decrease its sending rate to reduce lost packets, if the bandwidth decreases sharply.

MBTFRC detects bandwidth changes at the receiver end with a window based EWMA (exponential window moving average) filter called, bottleneck-bandwidth filter (MBF). EWMA filter are used to provide smooth estimates and are used for instance with TCP's Round Trip Time (RTT) estimation. The stability and flexibility of a EWMA filter is determined by the weight, which is fixed in traditional EWMA filters. In MBTFRC an adaptive filter is used to improve performance with vertical handovers.

As TFRC connection will underutilize a wireless link since it reacts to any loss as a sign of congestion, a mechanism has been proposed which opens multiple TFRC connections for a streaming application (MULTFRC) [CZ06]. The main benefit of this approach is that it is an end-to-end solution and therefore does not require any additional support from the network infrastructure. It also allows the sender to more fully utilize the available bandwidth if care is taken when choosing the number of connections and packet sizes. Naturally the main drawback is that opening more connections requires more resources, which in particular for handheld devices is an issue. Also the receiver might receive packets out of order if a single media stream is divided into several connections, although buffering at the receiver end can mitigate the problem. Also the mechanism is aggressive in closing down connections but conservative when opening them.

In MULTFRC the optimal number connections is not an integer so some underutilization still appears due to the quantization effect. For instance if the optimal number connections is 2.4, MULTFRC opens 2 connections. To overcome this problem in addition to the problem of large resource usage due to multiple connections an improved scheme over MULTFRC called All-In-One TFRC (AIO-TFRC) was invented [CZ05]. In this scheme the control law for opening multiple connections in MULTFRC is integrated into one TFRC connection with the same utilization performance as that of MULTFRC.

Another way to improve MULTFRC is to integrate the Loss Event Rate Discounting mechanism into MULTFRC (MULTFRC-LERD) [TH04]. As in this scheme the individual connections are better utilized and fewer connections are required. However, this approach still suffers from the quantization effect. In the mechanism a new connection is opened after a previous one reaches the highest discounting level.

Another approach into providing a smooth transmission rate in a wireless environment is to abandon the rate control equation used by TFRC completely. The designers of Analytical Rate Control (ARC) argue that the TCP throughput equation is not suitable for wireless environments. The problem is that it models the steady-state TCP behavior over error-free wireline links. The solution is to create a new equation which models the desired TCP behavior over lossy links [AA04]. This behavior model is characterized by two principles:

- The TCP source should not reduce data rate in the case of packet loss due to wireless error.
- It should behave as standard TCP otherwise.

The key features of ARC are that it decouples a single packet loss event from triggering the rate control process and that it achieves a smooth variance in transmission rate. The designers first created an end-to-end path model using Markov chains. The control equation is a function of the state probabilities of the path model and the RTT and is as follows:

$$T = \frac{1}{4 * RTT}\left(3 + \sqrt{25 + 24\left(\frac{1-\omega}{\pi-\omega}\right)}\right)$$

In the above equation ω stands for the probability of packet loss due to wireless link errors and π stands for total packet loss probability. The rate control scheme consists of a probe period and a steady period. In the probe period the sender determines the initial data transmission rate and switches into the steady period after 2*RTT. In the steady period it uses the rate control equation as values of ω and π become available via acknowledgments.

5.3 Cross Layer Approaches

In order to distinguish between wireless-error and congestion related packet losses related loss and some approaches have opted to use link layer information. These cross layer approaches can verify packet losses more accurately but implementing a cross-layer solution is usually more complicated and is not always feasible. TCP-friendly multimedia streaming protocol for wireless Internet (WMSTFP) uses the radio link control layer (RLC) and radio resource control layer (RRC) information when operating in third generation (3G) wireless systems [YZZZ04]. With the information obtained, it can accurately measure the loss event rate. This value is then used in the TCP throughput equation. WMSTFP also performs media adaptation with a separate module.

TCP-Friendly rate control with compensation (TFRCC) uses the same algorithm as TFRC when operating in a wired network. An improvement to TFRCC in wireless environments is called TFRCC-W which has opted to use the same algorithm as ARC for the throughput model [ZZL07]. TFRCC-W uses MAC layer information to measure the packet loss rate due to wireless link error. TFRCC-W also has a strong link to quality-of-service requirements and so if the sending rate calculated by the equation is less than that QoS dictates (source rate), TFRCC-W can violate TCP friendliness temporarily. In order to remain TCP friendly on the long term, a compensation algorithm is used. A middleware component between the application layer and the transport layer will make the decisions between the sending rate and the source rate. In a heavily congested network this approach will perform badly and increase congestion even more. A solution to include an "intelligent switching" function which will revert to a strict TCP-friendly mode if heavy congestion is measured can be useful here.

6 Protocols

Datagram Congestion Control Protocol (DCCP) designers had a simple goal. To create protocol which similar to UDP but which would include mechanisms to provide congestion control [KHF06b]. Many of the current multimedia applications use UDP as the underlying transport protocol but this can lead to problems since UDP itself does not contain congestion control mechanisms. DCCP allows applications to choose the congestion control mechanism to use and these mechanisms are called congestion control IDs (CCID) [KHF06a]. For example CCID 2 is congestion control similar to that of TCP [KF06]. CCID 3 implements the TFRC mechanism and CCID 4 is the small packet variant of TFRC [KPF06, FK07]. The usage of the Quick-Start (QS) mechanism with DCCP has also been specified in a recent draft [FA09]. With the use of the mechanism, a DCCP sender is able to negotiate a sending rate at the beginning of a connection and, at times, in the middle of a connection. Events which would require a DCCP sender to negotiate new sending rate in the middle of a connection could be, for example, after an idle period or after an application-limited period. Another optional mechanism for DCCP is Faster Restart [KFS08]. Faster Restart specifies that the allowed sending rate is never to be reduced by an idle period below four or eight packets per RTT, depending on the packet size. It also allows flows to reach their earlier achieved rate faster, by allowing them to quadruple their sending rate for every congestion-free RTT.

Real-time Transport Protocol (RTP) provides end-to-end transport functions for applications transmitting real-time data, such as audio or video [SCFJ03]. RTP is a simple protocol since it neither guarantees quality-of-service nor addresses resource reservation. RTP is augmented by a control protocol (RTCP), which allows monitoring of the data delivery and provides minimal control and identification functionality. RTP is typically run over UDP but can be run over other transport protocols. As packets may arrive out-of-order, some sort of buffering is needed at the end host. The services that RTP provides for the applications are: payload-type identification, sequence numbering, time stamping and delivery monitoring. RTCP can provide feedback on the quality of the reception to the various rate control mechanisms. Such information includes estimates about the interarrival jitter and packet loss rate. With this information, higher level rate control mechanisms can make adjustments to their transmission rate according to their algorithms.

Stream Control Transmission Protocol [Ste07] is a reliable transport layer protocol which provides service similar to TCP. The difference being that TCP is only interested in bytes, whereas SCTP is transaction oriented. In order to provide service to time sensitive traffic a partially reliable (PR-SCTP) extension has been proposed [SRX$^+$04]. It allows the sender to indicate how long a sender should try to transmit/retransmit a packet. PR-SCTP provides both an unreliable unordered data transfer service (similar to UDP) and an ordered unreliable data transfer service. SCTP can improve the timeliness of some applications, since missing packets from one stream do not delay the packets for other streams [KHF06b].

There have been some studies on proprietary protocols such as Skype, to understand the kind of rate control schemes they employ. Skype uses a codec that supports real-time video encoding and decoding using some sort of data control [CMP08]. This data control adjusts the frame quality, video resolution and the number of frames per second. These adjustments are made according to bandwidth variations. It employs some sort of congestion control as well [CMP07]. Measurements on the protocol have shown that it is more aggressive than TCP when packet drops occur and that it has packet drops when the bandwidth changes [CMP08] [CMP07].

7 Rate Adaptation

In providing a smooth real-time streaming, rate adaptation is also important. When the sender has decided on a sending rate, the next choice is choosing the stream quality. This is called rate adaptation. The simplest approach is to have a constant stream quality which is sent as is, with no extra processing. A more adaptive approach will have several different quality streams from which a stream is chosen according to the available sending rate. This allows the receiver to receive a coherent stream with no jitter. Some sort of state machine is used to select the rate and most likely the ratescale is quite coarse.

For the layered codec approach the video stream is divided into layers. Whenever bandwidth allows it, a layer is added on top of the existing layers to improve image/audio quality. This approach is related to the previous adaptive approach but there are no separate streams, merely layers to be added. This approach might increase header overhead when new layers might have their own headers. An example of this layered encoding is the MPEG standard [MPLC02]. The video stream is encoded into three frames I, P and B. The I frame is the base frame and the most important as losing it negates the relevant P and B frames for that frame. Therefore it is important that at least the I frame is delivered. For this reason the bottleneck routers must be able to choose which frames to drop in case the queues become full.

Scalable video coding (SVC) extension to H.264/AVC video coding standard is a recent approach to layered codecs [SMW07]. A high quality video bitstream encoded with SVC can contain several subset bitstreams than can themselves be decoded with a similar complexity and reconstruction quality to that of normal H.264/AVC stream. SVC allows for three types of scalability: temporal (frame rate), spatial (picture size) and quality scalability. Any combination of the three above is also possible. A single SVC file can therefore be streamed to several different clients with different scalability options. A substream can be formed easily by dropping packets from the larger bitstream and this could be used when available bandwidth changes were detected.

A fully adaptive approach will vary the bitrate according to the situation. This might include choosing different codecs for certain situations. It requires more computing resources and is not a very good solution if there are multiple streams from the same source. Transforming each stream according to its situation would require a large amount of computing power.

An error resistant approach will send the data to the applications even when the data is corrupted. This might require link layer support so that the corrupted data packets are not discarded at the link layer. It is possible to combine this approach with any of the other approaches. Currently at least UDP Lite [LDP$^+$04] and DCCP have support for this approach. They both allow the usage of partial checksums to cover only the header part of the message.

New audio codecs employ what is called silence suppression, where the stream goes totally idle sometimes for several seconds while one side listens to what the other side has to say [Phe07]. For video this is called motion compensation [Phe07]. The sending codec sends only the changes from one video frame to the other. This can lead to drastic changes in the transmission rate and can cause problems to congestion control mechanisms. The latest TFRC specification alleviates this problem somewhat by allowing a sender to collect unused send credits and after a data-limited period send a burst of packets [FHPW08].

8 Bandwidth Estimation

Bandwidth estimation techniques are used to estimate two values: the capacity of a path and the available bandwidth. Available bandwidth is usually defined as the bandwidth a new flow can achieve. On a given path the link with the minimum capacity is called a narrow link and the link with minimum available bandwidth is called a tight link. In the case when a link is both the narrow and the tight link on a given path it is often called a bottleneck. In a wireless setting this is often the last hop router. With the knowledge about the available bandwidth and capacity, it is possible to optimize the end-to-end performance of applications.

8.1 Capacity Estimation

Packet dispersion techniques, such as packet pair or packet train probing, measure the end-to-end capacity of a network path. These techniques rely on seminal work by Jacobson [Jac88], Keshav [Kes95] and Bolot [Bol93]. These techniques send two or more packets back-to-back into the network. After traversing the narrow link, the time dispersion between the two packets is

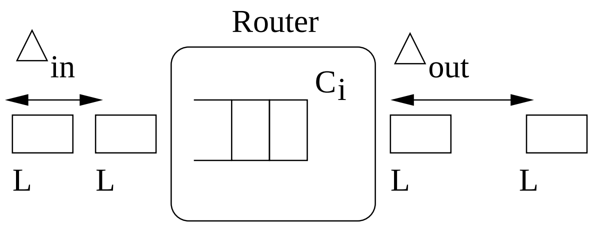

Figure 2: Packet dispersion illustrated

linearly related to the narrow link capacity. Crossing traffic and certain wireless features such as dynamic rate adaptation can cause problems to these methods. Figure 2 illustrates the basic idea.

In the figure L is the size of the probing packets, C_i is the capacity of the bottleneck router and Δ_{in} and Δ_{out} are the time dispersions before and after traversing the router. Capacity can be easily calculated with the following equation:

$$C = \frac{L}{G}$$

In the equation C stands for capacity, L is the size of the probing packets and G the packet dispersion measured at the receiver (ie. gap between the packets $\Delta_{out} - \Delta_{in}$). Crossing traffic and certain wireless features such as dynamic rate adaptation can cause problems to these methods. Also traditional packet drops and competing traffic cause erroneous results. The key is to send enough packet pairs that one of them will not encounter congestion and use the result from that pair.

The packet pair technique is widely used in capacity estimation tools such as bprobe [CC96], sprobe [SGG02] and pathrate [DRM04].

Bprobe is built upon the ICMP echo mechanism. The probe runs are divided to several distinct phases. Each phase consist of 10 packets or a certain size. After a phase concludes the next phase will have packets with a larger size. The smallest size used is 124 bytes and the largest is 8000 bytes. By varying the packets size it will eliminate the case where the packets go through the path without experiencing any queueing. To deal with competing traffic it relies on sending many packets and therefore increasing the likelihood that some packets go through without experiencing any disruption by competing traffic. By sending many packets it also deals with the issue of packet drops and potential downstream congestion. Again some packets will likely get through without negatively impacted by the mentioned phenomena. Using ICMP packets is its main drawback, currently ICMP packets are often dropped, filtered or even answered by an ISP on behalf of its clients [SGG02].

SProbe uses an exploit of the TCP protocol to do its measurements. It sends a SYN packet pair to an inactive port on the remote machine. This causes the remote machine to respond with a RST packet pair. Since normal SYN packets are only 40 bytes this is most likely not enough to cause time dispersion when it travels the path. For this reason SProbe appends a large payload of 1460 bytes to each sent SYN packet. The RST packets received at the local host are small (40 bytes) and unlikely to queue at the bottleneck link on the reverse path. The time dispersion is measured from these RST packets and the capacity is estimated from that. Some links might compress packets so therefore the payload of the SYN packets are precompressed to minimize the effect of such links. Also the main drawback of the approach is that some firewalls drop SYN packets to inactive ports which will cause the operation to fail. Traditionally tools which estimate capacity send massive amounts of packet-pairs to protect against cross traffic, reordering or interference from multi-channel links. SProbe's goal is to be fast so this approach is not suitable for it. It will send a small (40 byte) SYN packet-pair both before and after the larger SYN packet pair to create a packet train. The sequence numbers of the received RST packets are used to detect whether reordering occurred (due to multi-channel links or otherwise). Also if the time dispersion is greater on the smaller packet pairs it proves that some cross traffic has interfered with the probe packets. SProbe is a fast technique but it trades accuracy for speed.

Pathrate uses UDP for measuring the capacity but has also a TCP connection for control information between the hosts. The actual estimation is divided into two phases. In phase one pathrate sends 1000 variable sized packet pairs into the network and analyzes the distribution. It can find several potential capacity estimates and moves onto phase two. In phase two pathrate sends 500 packet trains with maximum sized segments and again analyzes the distribution, which in this case is typically unimodal. It then uses the results from both phase one and two to arrive to a conclusion for the capacity. Those capacity estimates from phase one which are smaller than the result from phase two are thrown away and the most common estimate of the remaining is chosen as the capacity. One should note that the result from phase two is the lower bound for the path's capacity and an upper bound for the path's available bandwidth.

The size of the probing packets is an important metric that needs to be carefully chosen. Traditional wisdom says that larger probing packets are better since they cause a larger dispersion which is easier to measure, more robust to queuing delay noise and less sensitive to the timestamping resolution at the receiver [DRM04]. The drawback is that larger packets are more prone to interference from cross traffic. The creators of pathrate suggest varying the size is beneficial to the estimates and that the packet sizes should be larger than "not a very small packet" and smaller than "not a very large packet" [DRM04]. Pathrate uses 550 bytes as a minimum size for the packets and the maximum segment size of the control channel as the upper bound.

Packet dispersion techniques measure the capacity of the entire path. Another solution for capacity estimation is the so called Variable Packet Size (VPS) probing schemes [PMDC03]. These techniques measure the capacity of each hop along a path. VPS uses the Time-To-Live (TTL) field of IP packets to make the packets of a particular probe reach only as far as some particular hop. This router then discards the packets and sends an ICMP "Time Exceeded" message back to the sender. The source then uses these ICMP packets for its analysis of the capacity. The principal idea is to use the RTT measured from the source to each hop of the path as a function of the probing packet size. Solutions which rely on VPS include pathchar [Jac97, Dow99] and clink [Dow98].

8.2 Available Bandwidth Estimation

Available bandwidth estimation is harder and therefore a multitude of solutions exist. They can be roughly divided into two approaches:

Direct Probing (also known as the Probe Gap Model (PGM))

The sender transmits a periodic probing stream rate R_i (packet pairs or packet trains) and the receiver measures the output rate R_o. The basic idea is that if the sending rate is more than the available bandwidth A, the available bandwidth can be calculated with an equation:

$$A = C_t - R_i \left(\frac{C_t}{R_o} - 1 \right)$$

The caveat is that the capacity of the bottleneck C_t needs to be estimated in advance. Direct probing is used by such tools as IGI/PTR [NS03] and Spruce [SKK03].

Spruce assumes a single bottleneck that is both the narrow and the tight link along the path. It also needs the capacity estimate known in advance which needs to be provided by another tool. It uses a variant of the equation above:

$$A = C * \left(1 - \frac{\Delta_{out} - \Delta_{in}}{\Delta_{in}} \right)$$

In the equation Δ_{in} stands for the time gap between the packets before the bottleneck router and Δ_{out} is the time gap after traversing the router. Spruce analyzes the arrival rate by sending a pair of packets spaced so that the second packet arrives at the bottleneck queue before the first probe packet departs the queue. Spruce then calculates the amount of bytes that arrived at the queue between the two probe packets. The Δ_{in} is chosen by spruce to the transmission time of a 1500B data packet on the bottleneck router. To improve accuracy spruce performs a sequence of measurements and reports the average. To limit intrusiveness it uses a Poison sampling process instead of packet trains (or chirps).

Initial Gap Increasing (IGI) and Packet Transmission Rate (PTR) are related algorithms which share the probing procedure [NS03]. The main difference between them is that IGI calculates the competing background traffic load and PTR calculates the available bandwidth directly. The IGI algorithm uses the information about changes in the gap values of a packet train to estimate the competing bandwidth on the tight link. The PTR method uses the average rate of the packet train as an estimate of the available bandwidth. The authors of the algorithms argue that the critical parameter in estimating available bandwidth with packet pairs is the initial probing gap. When the initial probing gap is too small, the method is flooding the system and when the gap is too large, the probes do not accurately report the available bandwidth. A key feature of the IGI/PTR method is to send packet trains with varying initial gaps and observe the turning point. At the turning point both methods report accurate estimates. The results provided in the paper show that IGI loses accuracy if the tight link is not the bottleneck link. Also if there is significant competing traffic after the tight link, the IGI algorithm loses accuracy.

Iterative Probing (also known as self-loading techniques or Probe Rate Model (PRM))

With iterative probing packet pairs (or trains) are sent at varying rates while the receiver observes the differences between the input and the output rates. When the output rate is lower than the input rate, the probing rate is higher than the available bandwidth. The changes to probing rate can be handled in various ways. Pathload uses a binary search to change the probing rate and to converge to the available bandwidth [JD03]. TOPP, on the other hand, increases its probing rate linearly between probing runs [SA03]. Pathchirp increases its sending rate exponentially [RRB$^+$03]. TOPP is also able to detect multiple bottlenecks along the path.

Iterative probing methods are more intrusive to the network compared to direct probing. However, the accuracy of direct probing has been called to question in recent studies [LDS06, JD04].

Both PGM And PRM methods assume that the routers employ FIFO queues. Another assumption is that the traffic follows a fluid model and that the average rates of cross traffic change slowly and is constant for the duration of a single measurement [SKK03].

These early tools were developed mainly for wireline networks, methods have been developed to improve their accuracy in wireless networks. CapProbe analyzes both packet dispersion and packet delay in filtering out samples which have been affected by cross-traffic [KCL$^+$04]. It produces only a capacity estimate of the path. DietTopp uses a simplified TOPP model to estimate the available bandwidth [JBM06]. Also it no longer detects multiple bottlenecks along the path but is faster than TOPP. ProbeGap gathers one-way-delay samples and uses these samples to estimate the idle time fraction of a link [LPP04]. It requires a third-party capacity estimation tool in order to provide a bandwidth estimate. IdleGap estimates the available bandwidth via the ratio of free time in the wireless links [LHY$^+$07]. To get this information it needs information from a lower layer. WBest uses a packet pair technique to estimate the capacity of a path and then uses a packet train to estimate the available bandwidth [LCK08].

9 Conclusions

We have presented a survey of rate control mechanisms. These mechanisms are divided into window-based and rate-based rate control mechanisms. Several issues with the various rate control mechanisms have been discussed. Some discussion on protocols and rate adaptation is also included. While TCP-Friendly Rate Control (TFRC) has become the standard, several other solutions still exist. Some mechanisms are more complex and might require support from the link layer. Which makes them more difficult to employ on a large scale. TCP-friendliness is a major concern in all of the mechanisms, but in the future if TCP dominance were to change, this fairness would become less important. Also included in the paper are the bandwidth estimation techniques which could be used to enhance a rate control scheme by providing measurements of the conditions in the network. What sort of rate control will ultimately be used by applications is unclear since software developers might opt for their own proprietary protocols such as Skype.

References

[AA04] O. B. Akan and I. F. Akyildiz. ARC: the analytical rate control scheme for real-time traffic in wireless networks. *IEEE/ACM Transactions on Networking*, 12(4):634–644, August 2004.

[BB01] D. Bansal and H. Balakrishnan. Binomial congestion control algorithms. *Proceedings of the Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2001)*, 2:631–640, 2001.

[BBFS01] D. Bansal, H. Balakrishnan, S. Floyd, and S. Shenker. Dynamic behavior of slowly-responsive congestion control algorithms. In *Proceedings of ACM SIGCOMM 2001*, pages 263–274, 2001.

[Bol93] J.-C. Bolot. End-to-End Packet Delay and Loss Behavior in the Internet. In *SIGCOMM '93: Conference proceedings on Communications architectures, protocols and applications*, pages 289–298. ACM, 1993.

[BV99] S. Biaz and N.H. Vaidya. Discriminating congestion losses from wireless losses using inter-arrival times at the receiver. In *ASSET '99: Proceedings of the 1999 IEEE Symposium on Application - Specific Systems and Software Engineering and Technology*, pages 10–17, Richardson, TX, USA, March 1999. IEEE Computer Society.

[CC96] R. Carter and Mark. Crovella. Measuring Bottleneck Link Speed in Packet-Switched Networks . Technical report, Boston, MA, USA, 1996.

[CCV03] S. Cen, P.C. Cosman, and G.M. Voelker. End-to-end differentiation of congestion and wireless losses. *IEEE/ACM Trans. Netw.*, 11(5):703–717, 2003.

[CH09] S. H. Choi and M. Handley. Designing TCP-Friendly Window-based Congestion Control for Real-time Multimedia Applications. In *PFLDNeT '09: Proceedings of the 2009 PFLDNeT conference*, Tokyo, Japan, 2009.

[CMP07] L. De Cicco, S. Mascolo, and V. Palmisano. An Experimental Investigation of the Congestion Control by Skype VoIP. WWIC'07, May 2007.

[CMP08] L. De Cicco, S. Mascolo, and V. Palmisano. Skype Video Responsiveness to Bandwidth Variations. NOSSDAV'08, May 2008.

[CPW97] S. Cen, C. Pu, and J. Walpole. Flow and congestion control for internet media streaming applications. Technical report, Oregon Graduate Institute School of Science & Engineering, 1997.

[CWL03] S. Cho, H. Woo, and J.-W. Lee. ATFRC: Adaptive TCP Friendly Rate Control Protocol. In *Lecture Notes in Computer Science*. SpringerVerlag, 2003.

[CZ05] M. Chen and A. Zakhor. AIO-TFRC: a light-weight rate control scheme for streaming over wireless. *2005 International Conference on Wireless Networks Communications and Mobile Computing*, 2:1124–1129, June 2005.

[CZ06] M. Chen and A. Zakhor. Multiple TFRC connections based rate control for wireless networks. *IEEE Transactions on Multimedia*, 8(5):1045–1062, Oct. 2006.

[Dow98] A.B. Downey. Clink: a tool for estimating internet link characteristics. Online, 1998.

[Dow99] A.B. Downey. Using pathchar to estimate internet link characteristics. In *SIGMETRICS '99: Proceedings of the 1999 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 222–223, 1999.

[DRM04] C. Dovrolis, P. Ramanathan, and D. Moore. Packet-dispersion techniques and a capacity-estimation methodology . *IEEE/ACM Trans. Netw.*, 12(6):963–977, 2004.

[FA09] G. Fairhurst and A.Sathiaseelan. Quick-Start for Datagram Congestion Control Protocol (DCCP) . IETF Internet-Draft "draft-ietf-dccp-quickstart-05.txt", June 2009. Work in progress.

[FHPW00] S. Floyd, M. Handley, J. Padhye, and J. Widmer. Equation-based congestion control for unicast applications. In *Proceedings of the conference on Applications, Technologies, Architectures, and Protocols for Computer Communication(ACM SIGCOMM 2000)*, pages 43–56, New York, NY, USA, may 2000.

[FHPW08] S. Floyd, M. Handley, J. Padhye, and J .Widmer. TCP Friendly Rate Control (TFRC): Protocol Specification. RFC 5348, Internet Society, September 2008.

[FK07] S. Floyd and E. Kohler. TCP Friendly Rate Control (TFRC): The Small Packet (SP) Variant. RFC 4828, Internet Society, April 2007.

[Flo03] S. Floyd. Highspeed TCP for large congestion windows. RFC 3649, Internet Society, December 2003.

[FX08] J. Feng and L. Xu. TCP-Friendly CBR-Like Rate Control. In *Proceedings of the Internation Conference on Network Protocols (IEEE ICNP 2008)*, pages 177–186, Orlando, FL, USA, October 2008.

[GK04] A. Gurtov and J. Korhonen. Effect of vertical handovers on performance of TCP-friendly rate control. *SIGMOBILE Mob. Comput. Commun. Rev.*, 8(3):73–87, 2004.

[Jac88] V. Jacobson. Congestion Avoidance and Control. In *SIGCOMM '88: Symposium proceedings on Communications architectures and protocols*, pages 314–329. ACM, 1988.

[Jac97] V. Jacobson. Pathchar: A Tool to Infer Characteristics of Internet Paths. MSRI Presentation, April 1997.

[JBM06] A. Johnsson, M. Bjorkman, and B. Melander. An Analysis of Active End-to-end Bandwidth Measurements in Wireless Networks. pages 74–81, April 2006.

[JD03] M. Jain and C. Dovrolis. End-to-end available bandwidth: measurement methodology, dynamics, and relation with tcp throughput. *IEEE/ACM Trans. Netw.*, 11(4):537–549, 2003.

[JD04] M. Jain and C. Dovrolis. Ten fallacies and pitfalls on end-to-end available bandwidth estimation. In *IMC '04: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pages 272–277, 2004.

[KCL$^+$04] R. Kapoor, L.-J. Chen, L. Lao, M. Gerla, and M.Y. Sanadidi. CapProbe: A Simple and Accurate Capacity Estimation Technique . *SIGCOMM Comput. Commun. Rev.*, 34(4):67–78, 2004.

[Kes95] S. Keshav. A Control-Theoretic Approach to Flow Control. *SIGCOMM Comput. Commun. Rev.*, 25(1):188–201, 1995.

[KF06] E. Kohler and S. Floyd. Datagram Congestion Control Protocol (DCCP) Congestion Control ID 2: TCP-like Congestion Control . RFC 4341, Internet Society, March 2006.

[KFS08] E. Kohler, S. Floyd, and A. Sathiaseelan. Faster Restart for TCP-Friendly Rate Control (TFRC) . IETF Internet-Draft "draft-ietf-dccp-tfrc-faster-restart-06.txt", July 2008. Work in progress.

[KHF06a] E. Kohler, M. Handley, and S. Floyd. Datagram congestion control protocol (DCCP). RFC 4340, Internet Society, March 2006.

[KHF06b] E. Kohler, M. Handley, and S. Floyd. Designing DCCP: congestion control without reliability. *SIGCOMM Comput. Commun. Rev.*, 36(4):27–38, 2006.

[KKK04] Y.-G. Kim, JW. Kim, and C.-C. Jay Kuo. TCP-friendly internet video with smooth and fast rate adaptation and network-aware error control. *IEEE Transactions on Circuits and Systems for Video Technology*, 14(2):256–268, February 2004.

[KPF06] E. Kohler, J. Padhye, and S. Floyd. Datagram Congestion Control Protocol (DCCP) Congestion Control ID 3: TCP-Friendly Rate Control (TFRC) . RFC 4342, Internet Society, March 2006.

[LCK08] M. Li, M. Claypool, and R. Kinicki. WBest: a Bandwidth Estimation Tool for IEEE 802.11 Wireless Networks. In *LCN '08: Proceedings of the 33rd Annual IEEE Conference on Local Computer Networks*, pages 374–381, nov 2008.

[LDP$^+$04] L-A. Larzon, M. Degermark, S. Pink, L-E. Jonsson, and G. Fairhurst. The lightweight user datagram protocol (UDP-Lite). RFC 3828, Internet Society, July 2004.

[LDS06] L. Lao, C. Dovrolis, and M.Y. Sanadidi. The probe gap model can underestimate the available bandwidth of multihop paths. *SIGCOMM Comput. Commun. Rev.*, 36(5):29–34, 2006.

[LHY$^+$07] H.K. Lee, V. Hall, K.H. Yum, K.I. Kim, and E.J. Kim. Bandwidth Estimation in Wireless LANs for Multimedia Streaming Services. *Adv. MultiMedia*, 2007(1), 2007.

[LKW$^+$03] Y. Lin, G. Kuo, H. Wu, Y. Peng, and S. Cheng. MBTFRC: a TFRC enhancement for heterogeneous mobile networks. *Global Telecommunications Conference (GLOBECOM '03). IEEE*, 5:2880–2884, Dec. 2003.

[LPP04] K. Lakshminarayanan, V.N. Padmanabhan, and J. Padhye. Bandwidth Estimation in Broadband Access Networks. In *IMC '04: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pages 314–321, 2004.

[MPLC02] K. Mayer-Patel, L. Le, and G. Carle. An MPEG performance model and its application to adaptive forward error correction. In *MULTIMEDIA '02: Proceedings of the tenth ACM international conference on Multimedia*, pages 1–10, New York, NY, USA, 2002. ACM.

[NS03] H. Ningning and P. Steenkiste. Evaluation and characterization of available bandwidth probing techniques. *Selected Areas in Communications, IEEE Journal on*, 21(6):879–894, Aug. 2003.

[PFTK98] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP throughput: a simple model and its empirical validation. *SIGCOMM Comput. Commun. Rev.*, 28(4):303–314, 1998.

[Phe07] T. Phelan. Strategies for Streaming Media Applications Using TCP-Friendly Rate Control. IETF Internet-Draft "draft-ietf-dccp-tfrc-media-02.txt", July 2007. Work in progress.

[PMDC03] R.S. Prasad, M. Murray, C. Dovrolis, and K. Claffy. Bandwidth estimation: metrics, measurement techniques, and tools . *Network, IEEE*, 17(6):27–35, 2003.

[Pos80] J. Postel. User datagram protocol. RFC 768, Internet Society, August 1980.

[Pos81] J. Postel. Transmission control protocol. RFC 793, Internet Society, September 1981.

[RHE99] R. Rejaie, M. Handley, and D. Estrin. RAP: An end-to-end rate-based congestion control mechanism for realtime streams in the internet. *Proceedings of the Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies.(INFOCOM '99)*, 3:1337–1345, March 1999.

[ROY00] I. Rhee, V. Ozdemir, and Y. Yi. TEAR: TCP emulation at receivers - flow control for multimedia streaming. Technical report, NCSU, April 2000.

[RRB$^+$03] V.J. Ribeiro, R.H. Riedi, R.G. Baraniuk, J. Navratil, and L. Cottrell. pathChirp: Efficient Available Bandwidth Estimation for Network Paths. Passive and Active Measurement Workshop, PAM2003, april 2003.

[RX05] I. Rhee and L. Xu. Limitations of equation-based congestion control. In *SIGCOMM '05: Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 49–60, New York, NY, USA, 2005. ACM.

[SA03] M. Singh and N. Ahuja. Regression based bandwidth selection for segmentation using parzen windows. volume 1, pages 2–9, oct 2003.

[SCFJ03] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A transport protocol for real-time applications. RFC 3550, Internet Society, July 2003.

[SGG02] S. Saroiu, P.K. Gummadi, and S.D. Gribble. Sprobe: A Fast Technique for Measuring Bottleneck Bandwidth in Uncooperative Environments. IEEE Infocom Submission, 2002.

[SKK03] J. Strauss, D. Katabi, and F. Kaashoek. A measurement study of available bandwidth estimation tools. In *IMC '03: Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, pages 39–44, Miami Beach, FL, USA, 2003.

[SMW07] H. Schwarz, D. Marpe, and T. Wiegand. Overview of the Scalable Video Coding Extension of the H.264/AVC Standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(9), 2007.

[SRX$^+$04] R. Stewart, M. Ramalho, Q. Xie, M. Tuexen, and P. Conrad. Stream control transmission protocol (SCTP) partial reliability extension. RFC 3758, Internet Society, May 2004.

[Ste07] R. Stewart. Stream Control Transmission Protocol. RFC 4960, Internet Society, September 2007.

[TGH04] X. Tong, W. Gao, and Q. Huang. A novel rate control scheme for video streaming over wireless networks. In *ICIG '04: Proceedings of the Third International Conference on Image and Graphics*, pages 369–372, Washington, DC, USA, 2004. IEEE Computer Society.

[TH04] X. Tong and Q. Huang. MULTFRC-LERD: An improved rate control scheme for video streaming over wireless. In *Proceedings of the 5th Pacific Rim Conference on Multimedia*, pages 282–289, Tokyo, Japan, nov 2004.

[TLL07] S.-C. Tsao, Y.-C. Lai, and Y.-D. Lin. Taxonomy and Evaluation of TCP-Friendly Congestion-Control Schemes on Fairness, Aggressiveness, and Responsiveness. *IEEE Network*, 21(6):6–15, November-December 2007.

[TMAJ01] 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. *Proceedings of the Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2001)*, 1:114–122, 2001.

[TTM$^+$00] Y. Tobe, Y. Tamura, A. Molano, S. Ghosh, and H. Tokuda. Achieving moderate fairness for udp flows by path-status classification. In *LCN '00: Proceedings of the 25th Annual IEEE Conference on Local Computer Networks*, pages 252–261, Tampa, FL, USA, November 2000. IEEE Computer Society.

[VB05] M. Vojnović and J-Y. Le Boudec. On the long-run behavior of equation-based rate control. *IEEE/ACM Trans. Netw.*, 13(3):568–581, 2005.

[WBJ04] Z. Wang, S. Banerjee, and S. Jamin. Media-friendliness of a slowly-responsive congestion control protocol. In *NOSSDAV '04: Proceedings of the 14th international workshop on Network and operating systems support for digital audio and video*, pages 82–87, New York, NY, USA, 2004. ACM.

[XHR04] L. Xu, K. Harfoush, and I. Rhee. Binary increase congestion control (BIC) for fast long-distance networks. *Proceedings of the Twenty-third Annual Joint Conference of the*

IEEE Computer and Communications Societies (INFOCOM 2004), 4:2514–2524, March 2004.

[Xu07] L. Xu. Extending equation-based congestion control to high-speed and long-distance networks. *Comput. Netw.*, 51(7):1847–1859, 2007.

[YKL03] Y. R. Yang, M. S. Kim, and S. S. Lam. Transient behaviors of TCP-friendly congestion control protocols. *Comput. Netw.*, 41(2):193–210, 2003.

[YL00] Y.R Yang and S.S. Lam. General AIMD congestion control. Technical report, Austin, TX, USA, 2000.

[YSG$^+$06] G. Yang, T. Sun, M. Gerla, M. Y. Sanadidi, and L.-J. Chen. Smooth and efficient real-time video transport in the presence of wireless errors. *ACM Trans. Multimedia Comput. Commun. Appl.*, 2(2):109–126, 2006.

[YZZZ04] F. Yang, Q. Zhang, W. Zhu, and Y.-Q. Zhang. End-to-end TCP-friendly streaming protocol and bit allocation for scalable video over wireless Internet. *IEEE Journal on Selected Areas in Communications*, 22(4):777–790, May 2004.

[ZL04] Y. Zhang and D. Loguinov. Oscillations and buffer overflows in video streaming under non-negligible queuing delay. In *NOSSDAV '04: Proceedings of the 14th international workshop on Network and operating systems support for digital audio and video*, pages 88–93, New York, NY, USA, 2004. ACM.

[ZZL07] P. Zhu, W. Zeng, and C. Li. Cross-layer design of source rate control and congestion control for wireless video streaming. *Advances in MultiMedia*, 2007(1):3–3, 2007.