Edge-based Differentiated Services

Henrik Lundqvist[1], Ignacio Más Ivars[1], and Gunnar Karlsson[1]

[1] Laboratory for Communication Networks
KTH, the Royal Institute of Technology, Electrum 229
164 40 Kista, Sweden
{henrik.lundqvist,nacho,gk}@imit.kth.se

Abstract. Network quality of service is traditionally thought to be provided by a combination of scheduling in the network nodes to enforce a capacity sharing policy and traffic controls to prevent congestion that could annihilate that policy. The work presented herein is instead based on an end-to-end argument: A capacity sharing policy is enforced by traffic controls in the hosts at the edges of the network, without any scheduling support in the network. Our proposal is to add a feed-forward control at the transport layer to provide a service that is better suited to conversational and streaming applications than the batch-oriented transfer mode provided by TCP. The paper presents the control and its evaluation: We compare the sharing of capacity between traffic classes and study the loss rate seen by admitted streams. The outcome is that the new control adds a distinctly different service to the service offered by TCP for the Internet.

1 Introduction

Conversational and streaming services need quality assurances in the Internet. Most of these services convey audio-visual data that have inherent rates, determined by rate-distortion tradeoffs in the encoding of signals from the different information sources. Human perception places a limit on the amount of delay that is acceptable for communication. For conversational services, it is the well-established mouth-to-ear delay of 150 to 200 ms that need be respected as well as the adjoining lip-to-mouth synchronization of moving images that roughly lies within the range of ±100 ms [12]. Furthermore, human perception favors consistency. The network should hence allow a session to complete when started without noticeable changes in quality which could annoy the user or which might render the session useless.

Since TCP congestion control is not adequate for streaming and conversational services, we propose to add a second congestion control to the transport layer of the IP protocols to provide a consistent quality. The throughput should with high probability be at a level that exceeds the bit rate of the stream. Thereby, we make the quality of service assurance into a delay-loss tradeoff that can be made outside the network [11].

The congestion control we propose is a probe-based admission control, which the authors have proposed in several prior publications [3][4][5]. However, it has hitherto been combined in the classical manner with network scheduling for providing isolation between traffic classes as well as between probes (flow establishment attempts) and ongoing flows. Section 2 in this paper describes how this probe-based admission control works and how it can be used to provide resource sharing with TCP along a network path without any differentiation in the network. The two controls are supposed to provide different types of services, and it is important to note that the two service classes have incomparable characteristics and can therefore not be judged better or worse in any general sense: Only for a given application can it be said that one or the other class is the best.

Section 3 contains a description of how FEC can be added to separate the loss requirement of the applications from the loss rate in the network, and how the parameters of the FEC and the admission control should be set. In Section 4 simulations are used to evaluate the scheme in terms of fairness and the provided quality. Finally, the conclusion of the evaluation of the proposal is given in Section 5.

The work presented herein is an extension and evaluation of the initial proposal presented in [1]. The proposal by Roberts and Oueslati-Boulahia for flow aware networking gives the rationales for the classification of traffic into a stream class and a batch class, and it gives the general properties of the classes [2]; the basic idea of providing two different classes that cannot be ranked in goodness is akin to the alternative best effort proposal by Hurley et alii [9]. The suggested implementations of these two proposals are however entirely network centric. Our proposal is the antithesis of the TCP friendly rate control of Floyd et alii [7] in the sense that we do not promote rate adaptation per stream, but allow streams to be inelastic, when admitted into the network; the probe-based admission control ensures that the aggregate of admitted streams is responsive to congestion in a way that is fair to TCP. There is a time-scale separation that need be recognized: TCP reacts fast to congestion but will also quickly capture capacity that becomes available, while the stream class will be slower to react both to congestion and to available capacity. So, the services provided by the TCP congestion control and the probe-based congestion control are clearly different.

2 Probe-based Admission Control

Probe-based admission control can be performed without any support from the network. A new flow can only be established after probing the path across the network to the receiver and determining that the network state is acceptable. A probe is a stream of packets that is sent at a constant rate, R_{UDP}, which is equal to the peak rate of the variable-rate flow. The contents of the probe packets may be control data for the flow, such as encoding and flow parameters. The receiver may furthermore use the constant-rate probe-packet stream for clock synchronization and for allowing the jitter removal control system to settle into steady state. The details of the probe-based admission control are described in [4].

The receiver acknowledges received probe packets, which allows the sender to estimate the loss probability for the path, denoted by p. The important criterion is when to accept a flow. Our policy bases the decision on the estimation of an equivalent TCP sending rate:

$$r_{TCP} = \frac{MSS}{RTT}\sqrt{\frac{C}{p}}, \qquad (1)$$

where C is a constant related to the throughput of TCP, RTT is the round trip time and MSS is the maximum segment size, measured in bits. The constant C is often set to 1.5, but the exact value depends on details in the TCP parameters, such as whether delayed acknowledgements are used. In this context C can be used to tune the sharing between UDP and TCP, and it has been chosen to 1.0 in the simulations in this paper based on some experimental evaluation. The flow may be established when the probe rate is below the equivalent TCP rate, $R_{UDP} < r_{TCP}$, and rejected otherwise. (An admission policy based on comparing the probe loss to a fixed threshold is included in the evaluation of ref. [1].)

3 Parameter Setting

In this section we investigate how the loss requirement of an application can be met when the loss rate in the network for an accepted flow would exceed the tolerance of the application. By using forward-error correction, it is possible to achieve a separation between the loss probability seen by the application and the loss probability of the network. We assume that the application has a certain requirement on the data rate, the maximum tolerable delay and loss. For example, an audiovisual application may use the rate-distortion and loss-distortion functions to determine the total distortion at a given loss probability and data rate. Depending on the desired quality, the data rate and the maximum loss requirement are determined. Using knowledge about the admission policy it is also possible to find good combinations of data rate and loss requirement, p_{req}, for given distortion requirements.

The parameters of the error correction are set statically for a flow to values that give the highest chance of admittance while not being unfair to the TCP traffic. In order to achieve this we make use of the TCP throughput equation again. From (1) the loss probability, p_{eq}, that corresponds to a certain bit rate, R_{UDP}, can be found to be:

$$p_{eq} = \frac{MSS^2 C}{RTT^2 R_{UDP}^2} \qquad (2)$$

Equation (2) shows how the admission threshold for the probe based admission control can be set when the MSS and the RTT are already known. By comparing p_{eq} and p_{req} we can deduce which one sets a lower requirement on the loss probability and use that as admission threshold. The only information required for this is the RTT and the MSS, which therefore must have been estimated beforehand, for example using ICMP echo requests and path-MTU discovery. As it turns out, measurement of the RTT may in fact not be necessary: In Section 4.6 we will introduce a compensation for the delay that will be used in the admission policy. It means that the parameters of the FEC will not depend on the RTT. For now, assume that the RTT and MSS are known, however.

The probe-based admission control can only give probabilistic guarantees about the quality of a flow, therefore it is important to include margins that ensures that the loss rate will be sufficiently good. In principle it would be desirable to have a guarantee that an admitted flow should experience a lower loss rate than the specification, with a given probability. However, since that depends on the future development of the loss rate and the length of the flow it is not feasible. A better solution is to ensure that the loss rate on the probed path does not exceed the required loss rate with a certain probability. We base the loss-probability estimate on the assumption that the measured loss is normally distributed for the probes, and use a 95 percent confidence level [3]. The assumption of normal distribution is motivated by central limit arguments, which are valid since we choose the probe long enough for the admission threshold. If FEC is not taken into account, the loss threshold for the admission process including margin can be written as:

$$p_{m\arg} = p_{req} - z_R \sqrt{\frac{p_{req}(1-p_{req})}{N_{probe}}}\ , \tag{3}$$

where z_R is the R-percentile for a normal distribution and N_{probe} is the number of packets in the probe. The resulting p_{marg} will hence be significantly lower than the loss requirement p_{req}. Without FEC the admission threshold would have to be set to the minimum of p_{marg} and p_{eq}.

To evaluate the gain of FEC the packet losses are assumed to be uncorrelated. Even though the real loss process is correlated the simulations will show that this assumption is sufficiently good for our purposes. We consider block codes, such as Reed-Solomon codes, with erasure decoding so that the number of recoverable losses per block is the same as the number of added redundant packets. The number of lost packets in a block is geometrically distributed and the loss rate after FEC can be calculated for a given block length of *N* data packets and *M* redundant packets:

$$p_{packetloss} = \sum_{i=M+1}^{N+M} \binom{N+M}{i} p^i (1-p)^{N+M-i} \frac{i}{N+M}\ . \tag{4}$$

The last factor in (4) is the fraction of lost packets; it converts the block loss rate to packet loss rate. The block length is determined by the delay tolerance of the application as dR_{UDP}/P_{size}, where d is the delay and P_{size} is the size for UDP packets (we assume equal packet size).

After recovering losses with FEC the remaining loss rate from (4) should be lower than p_{req} with 95 percent probability. Therefore the 95 percent confidence interval is added to the admission threshold level p_{eq}.

$$p_{FECm\arg} = p_{eq} + z_R \sqrt{\frac{p_{eq}(1-p_{eq})}{N_{probe}}}\ . \tag{5}$$

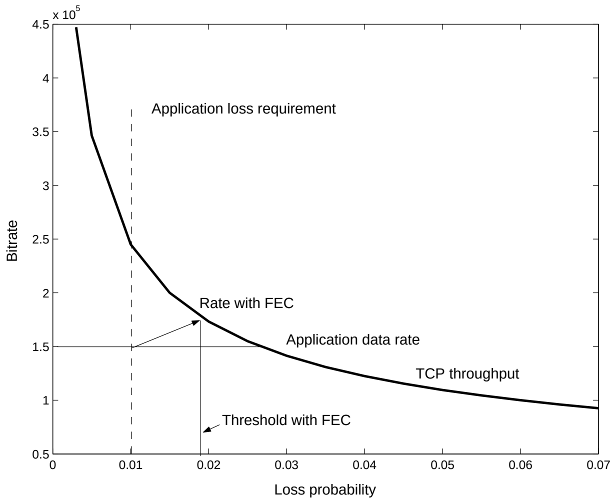

Fig. 1. When FEC is added the total rate is increased until it corresponds to a rate on the TCP throughput curve where the application requirement is met.

When $p_{FECmarg}$ is used to determine the residual loss rate after FEC from (4), the probability that the actual loss rate is too high to provide low enough residual loss rate is lower than five percent.

The procedure for setting the amount of redundancy and the admission threshold is illustrated in Fig. 1. The source starts by using equation (2) to calculate the loss rate that corresponds to the peak data rate. If p_{eq} from (2) is higher than p_{marg} from (3), then the flow must use FEC. The source may first count on adding one redundant packet to each block; the corresponding increase in sending rate leads to a new value for p_{eq} which is calculated from (2). Using p_{eq} the loss rate including margin, $p_{FECmarg}$, is calculated from (5), and the packet loss rate after decoding can be calculated from (4). If this is higher than the loss requirement, another redundant packet is added and the calculations are repeated. This is iterated until the loss rate after decoding is lower than what the application demands. In Fig. 1 this means that the admission threshold p_{eq} will be on the TCP throughput curve and the angle of the arrow will be determined by the block length, i.e. the delay tolerance of the application.

4 Simulation Results

All results presented in the paper are from simulations with NS-2. The experimental settings have been chosen to provide insight into the characteristics of the scheme without adding unnecessary complexity.

4.1 TCP Fairness

A first simulation experiment aims at evaluating the fairness between TCP and streaming UDP traffic. For this purpose a single bottleneck topology with a capacity of 20 Mb/s and a one-way propagation delay of 50 ms is simulated. The first traffic scenario consists of 30 long-lived TCP Newreno flows and a varying number of constant bit rate UDP flows. The UDP flows use a probe length of one second, have a loss requirement of one percent and a net throughput of 200 kb/s. Following the procedure described in Section 3, FEC is added with three redundant packets per 20 data packets, hence resulting in a 230 kb/s total rate. The UDP flows arrive as a Poisson process and have an exponentially distributed flow length with average 50 s. In Fig. 2 the expected throughput per flow for TCP and UDP is plotted as a function of the loss rate in the network with 95 percent confidence intervals indicated. The expected goodput per UDP flow is calculated as the net sending rate times the admission probability which decreases at high loss rates due to the high probability of blocking, whereas the TCP throughput for each flow is decreased as the congestion control reacts to the packet losses. Despite of these differences in mechanisms the result is that the expected throughput is similar for both traffic types. The main difference is that TCP gets a higher throughput when the load on the network is low, since TCP will be using up the extra capacity, whereas admitted UDP flows have no need to use a higher rate than the peak rate R_{UDP}.

Note that the sharing of TCP and UDP can be changed by modifying the parameter C in (1), for example a higher value for C results in lower blocking probability for

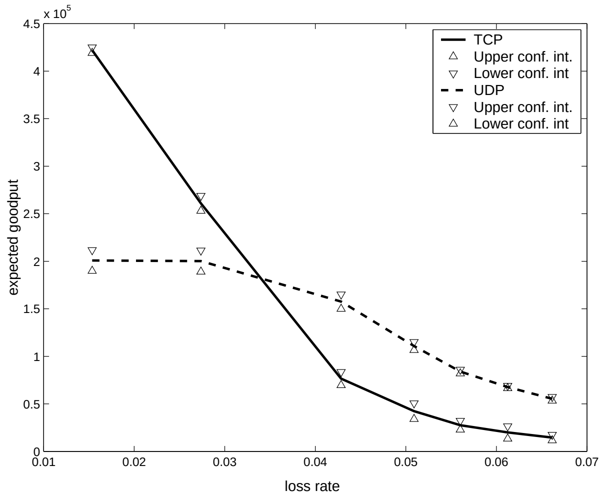

Fig. 2. The expected throughput of each TCP and UDP session decrease at similar rates as the number of UDP sessions increase. The decrease in expected UDP rate is due to higher blocking probability.

UDP traffic. Note also that the sending rate for UDP is actually 15 percent higher when redundancy is included. It can also be seen from Fig. 2 that the shape of the curves are different, the TCP throughput is essentially convex as opposed to the expected UDP throughput, hence no perfect fairness can be defined. Furthermore, the sharing between UDP and TCP flows also depends on the traffic mix and the properties of the loss process [15]. Therefore, the targeted fairness between TCP and UDP should be such that neither of the services always gets a higher throughput over a wide range of parameters. Hence, for a specific application it should be favorable to use the intended service class.

4.2 FEC Gain

To evaluate the effect of FEC we simulate a single link topology with different offered traffic loads and evaluate the loss rate with and without FEC. The scenario is essentially the same as in the first simulation with 30 TCP flows and an increasing number of UDP flows that can tolerate a packet loss rate of maximum one percent. The offered UDP load is varied from 25 percent to 300 percent of the link capacity by increasing the arrival rate of UDP flows, however, some of the flows will be rejected. Fig. 3 shows that the loss rate of the flows can be reduced so that they achieve a sufficient quality as long as the offered UDP load is not higher than 150 percent of the link capacity. Without FEC the loss rate would on average be higher than the acceptable level. To simplify the comparison of UDP with and without FEC there is no 95 percent confidence level here, otherwise an even higher load would be tolerable without causing too high loss for the applications. This also has the effect of changing the FEC

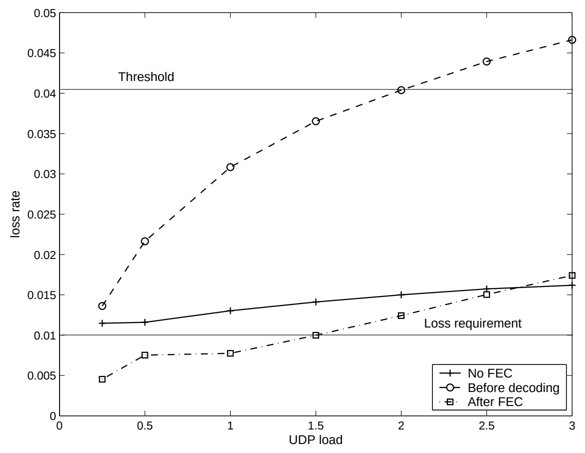

Fig. 3. The loss rate can be reduced to a level that is acceptable to the application by adding FEC. The curve with the highest loss rate is with FEC before decoding, i.e. the loss rate in the network, and the lowest is with FEC after decoding. Without FEC more sessions are rejected, hence the loss rate in the network is lower than with FEC.

parameters so that only 10 percent redundancy is added, rather than 15 in the previous section. With the 95 percent confidence level the problem is that the acceptance probability without FEC is very low. Fig. 3 also shows that at high load some flows will be admitted even though the loss rate is actually higher than the threshold, this is due to the estimation inaccuracy in the probing process. Note that the loss rate after FEC does not increase significantly between 50 and 100 percent offered UDP load. This is not an effect of simulation inaccuracy; the efficiency of FEC depends on the correlation of the loss process and the degree of multiplexing, which in turn depends on the traffic. As the share of UDP traffic increase at the expense of TCP the losses are less correlated and FEC becomes more efficient.

4.3 Fairness between Applications with Different Requirements

The choice of FEC parameters and threshold described in Section 3 does not only define the fairness between TCP and UDP flows. It also provides a way of defining fairness between real-time flows with differing requirements on loss rate, data rate and delay. The method described results in an admission threshold that determines the admission probability of a flow. To illustrate this, a traffic scenario with eight different UDP classes has been investigated. In Table 1 the rate, loss requirement and the FEC block length are given for the different classes. The FEC block length follows directly from the delay requirements of each application. Class 3 has the largest delay budget allocated for FEC in this example, the delay is 200 ms for a packet size of 188 bytes. From the three given parameters, the number of redundancy packets per block and the admission threshold are calculated. Note that due to the 95 percent confidence level, the threshold is lower than the required loss rate also for class 2 where no redundancy is added. The blocking probability for each of the classes is found from the simulation. As expected the blocking probability follows the admission thresholds so that the most demanding flows are least likely to be admitted. The last column in Table 1 shows the percentage of the admitted flows that experience a higher loss rate than their requirement. Clearly, almost all the flows get their required quality.

Table 1. Parameters and simulation results for UDP flows with differing requirements. Failed corresponds to admitted flows that get a higher loss probability than the requirement. The 95 percent confidence intervals are less than 3% for blocking rate and less than 1% for failed flows.

Class	Rate (kb/s)	Loss req. %	Block lgth.	Thr. %	Red. pkts	Block prob. %	Failed %
0	500	0.5	40	0.8	2	98	0
1	500	2.0	-	1.4	0	80	0
2	300	0.5	10	1.8	2	55	0
3	300	0.5	40	2.2	3	29	0.4
4	300	2.0	10	2.1	1	34	0
5	300	2.0	40	2.3	2	24	0
6	100	0.5	10	10.2	5	0	0.1
7	100	2.0	10	11.7	4	0	0

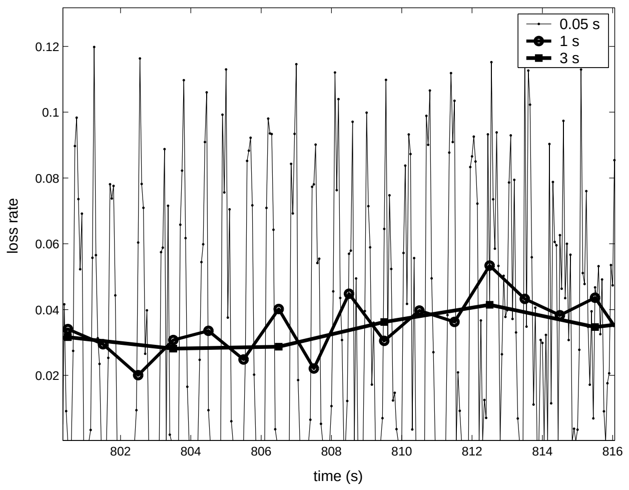

Fig. 4. The loss process has a very bursty behavior on short time scales and a probe length of at least one second is required to get a reasonable estimate of the loss rate.

4.4 Time Dynamics

As has already been noted in previous sections, the properties of the loss process have an impact both on the TCP throughput and on the FEC efficiency. Furthermore, the time dynamics of the channel impacts both the estimation of the loss rate and how well the loss estimate of the probe works as a predictor for the loss rate during the flow. In Fig. 4 the loss process for a typical simulation has been plotted, measured as loss average over 50 ms, one and three seconds respectively. The simulated scenario is the same as in Section 4.1 with 30 UDP flows and an average offered UDP load of 20 Mb/s. The first observation is that the average time between loss epochs is around 0.5 seconds; this depends on both the RTTs of the TCP flows, the buffer size and the number of TCP flows. Hence the typical time between loss epochs can vary significantly in more heterogeneous scenarios, but a first conclusion is that a probe time shorter than one second is inappropriate. With a probe length of one second the loss estimate can vary significantly, as can be seen from the one second mean. The three second mean gives a better estimate of the long term loss rate in the network as can be seen from the smoother curve. However, the experienced probe loss is used to make a decision regarding the whole duration of the flow and it might not be meaningful to make a very accurate estimation of the momentary loss rate. For flows with a long life time it can be expected that the loss rate will vary due to fluctuations in the load. Hence, to limit the probing delay at the start of a new session it would be reasonable to choose a probe length between one and three seconds. Further investigations of the effect of probe length and session length can be found in [15].

4.5 The Importance of Delay

One objective of the service differentiation is that the chances of getting a flow accepted should mainly depend on the desired rate and loss levels of the application and on the loss rate of the path. To evaluate this we use a topology with two bottlenecks where the capacity of the links is 20 Mb/s. Although it is not a complex network, it serves the purpose of illustrating the service differentiation in a more realistic way than a ‘dumb-bell’ topology. The traffic in the network consists of combinations of long-lived TCP flows and TCP controlled smaller file transfers (mice).

To investigate the effect of the delay we consider a topology with six different paths: P1-P6. Paths P1 and P2 both have two bottleneck links, the difference between them is that P1 has a one-way propagation delay of 50 ms and P2 has 100 ms. The first bottleneck of P1 and P2 is shared with P3 and P5, whereas the second bottleneck is shared with P4 and P6. P3 and P4 both have a one-way propagation delay of 50 ms while P5 and P6 have 100 ms. On each of the six paths there are ten persistent TCP flows, and on the paths P3 to P6 there are TCP mice contributing 0.5 Mb/s on each of the paths. The offered UDP load corresponds to four Mb/s per path, or equivalently to a total of 40 percent of the link capacity, consisting of flows of 200 kb/s with a loss requirement of one percent. The results in Table 2 reveals that the blocking rate is higher for the path with longer delay but lower loss rate, than for the path with two bottlenecks and low delay. This is due to the comparison with TCP, since TCP throughput depends heavily on the RTT. With a delay of 50 ms the UDP source can add 3 redundancy packets to a block of 20 data packets and get a resulting admission threshold of 4 %. The paths that experience a 100 ms one-way delay can only add one redundant packet and get an admission threshold of 1.2 %. This is an undesirable effect that follows from the definition of fairness by comparison with TCP. This can be seen from the fact that the throughput of the TCP flows and the blocking probability of UDP on the different paths follow the same pattern. For TCP the effect is a consequence of the window based congestion control, and should not necessarily be considered as a problem. However, feed-forward admission control does not have the same issues regarding stability as the feedback congestion control of TCP, therefore the effect is inappropriate for UDP.

Table 2. The results for paths with different delays show that the UDP flows are not affected by the delay. The 95 percent confidence intervals are smaller than 3% for the blocking rate, 0.05% for the path loss, 1% for failed flows and less than 10 kb/s for the TCP throughput.

	P1, 50 ms	P2, 100 ms	P3/P4, 50ms	P5/P6, 100ms
Blocking probability	4 %	100 %	0 %	30 %
Path loss	2.5 %	-	1.1 %	1.1 %
Loss rate (after FEC)	2.3 %	-	0 %	0 %
TCP throughput per flow	380 kb/s	180 kb/s	590 kb/s	250 kb/s

4.6 Delay Compensation

To avoid the undesirable RTT dependence we can consider a hypothetical TCP flow using segment sizes proportional to the round-trip delay. This would cancel the RTT dependence in the TCP throughput equation (1). Of course, if we modify the admission threshold accordingly, there is no guarantee that the policy is fair to TCP anymore. As could be seen in Section 4.1 it is not possible to guarantee perfect fairness between TCP and UDP, therefore it makes sense not to let the TCP comparison impair the fairness criterion between different UDP flows.

With this modification, equation (2) would change to

$$p_{eq} = \frac{T}{R_{UDP}^2}, \qquad \textbf{(6)}$$

where *T* is a constant. For example, if we would use a hypothetical segment size of 500 bytes for the path with 50 ms one-way delay and 1000 bytes for the path with one-way delay 100 ms, T would be equal to $1.6\text{x}10^9$. With this modification the service received by UDP sessions does not depend on the RTT [15].

This new admission policy is independent of the RTT and the MSS of TCP. Therefore, there is no need to estimate the RTT before choosing the FEC parameters, as mentioned in the Section 3.

There are of course other possible criteria that can be used, which do not have to be related to TCP fairness at all. In that case the admission threshold does not have to be inversely proportional to the square of the sending rate.

5 Conclusions

We have presented an entirely edge-based scheme for providing service differentiation to streaming and elastic traffic. Probe-based admission control is used to make streaming traffic TCP friendly without a need for per-flow rate control. Even though the two traffic controls work on different time scales simulations show that there is a reasonable fairness between TCP and streaming traffic. Furthermore, we address two problems with TCP fairness: The loss rate in the network may not be acceptable to the application at equilibrium and the TCP fairness depends on parameters that are not relevant to the streaming traffic.

The first of these problems is addressed by FEC, which is added in a way that does not discriminate against TCP traffic.

The second problem is addressed by modifying the admission policy not to take the round-trip time into account, but still to maintain TCP fairness on the average. Since the policies not only define fairness between TCP and UDP, but also between UDP flows with differing requirements, it is important to use a policy that is not too closely tied to the TCP throughput equation when it impairs the fairness between UDP flows.

In our future work we intend to analyze and optimize the parameters of the probing process to further improve the stability, fairness and isolation between flows. Furthermore, the scheme will be evaluated on larger scale networks and over longer time scales to provide more realistic conclusions about the performance.

We conclude from our evaluation that the proposed scheme can offer useful differentiated services for a wide range of network scenarios, as exemplified in the paper. It is our belief that this is an appropriate first step towards quality service for conversational and streaming applications over the Internet.

References

1. G. Karlsson, H. Lundqvist and I. Más Ivars, ”Single-Service Quality Differentiation,” Proc. IEEE IWQOS, Montreal, Canada, June 7-9, 2004.
2. J. W. Roberts and S. Oueslati-Boulahia, “Quality of Service by Flow Aware Networking,” in Phil. Trans. of The Royal Society of London, series A, vol. 358, no. 1773, August 2000.
3. V. Fodor (née Elek), G. Karlsson, and R. Rönngren, "Admission Control Based on End-to-End Measurements," Proc. IEEE INFOCOM, Tel-Aviv, Israel, March 26-30, 2000.
4. I. Más Ivars and G. Karlsson. “PBAC: Probe–Based Admission Control”, Proc. QoFIS, Coimbra, Portugal, Sept. 2001.
5. I. Más Ivars, V. Fodor and G. Karlsson, “Probe-Based Admission Control for Multicast,” Proc. IWQOS, Miami Beach, May 2002.
6. P. Key, L. Massoulié, A. Bain and F. Kelly, “Fair Internet traffic integration: network flow models and analysis,” to appear in Annals of Telecommunications, special issue on teletraffic. Available at URL http://research.microsoft.com/users/lmassoul/annals-telecom.ps
7. S. Floyd, M. Handley, J. Padhye and J. Widmer, “Equation-based congestion control for unicast applications,” Proc. ACM SIGCOMM, Stockholm, Sweden, 2000.
8. E. Kohler and S. Floyd, “Datagram congestion control protocol (DCCP) overview,” URL http://www.icir.org/kohler/dcp/summary.pdf.
9. P. Hurley, M. Kara, J. Y. Le Boudec and P. Thiran, “ABE: Providing a Low-Delay Service within Best Effort" IEEE Network Magazine, Vol. 15, No. 3, May 2001.
10. J. Padhye, V. Firoiu, D. Towsley and J. Kurose, “Modeling TCP Reno performance: a simple model and its empirical validation,” IEEE/ACM Trans. on Networking, Vol. 8, Issue 2, April 2000.
11. G. Dán, V. Fodor, “Quality Differentiation with Source Shaping and Forward Error Correction,” in Proc. MIPS, Naples, Italy, Nov. 18-21, 2003.
12. R. Steinmetz, “Human perception of jitter and media synchronization” IEEE Journal on Selected Areas in Communications, Vol. 14, No. 1, Jan. 1996.
13. M. Roughan, A. Erramilli, D. Veitch, “Network Performance for TCP Networks Part 1: Persistent Sources”, International Teletraffic Congress, ITC 17, 2001.
14. P. Frossard, “FEC Performances in Multimedia Streaming”, IEEE Comm. Letters, Vol. 5, No 3, March 2001.
15. H. Lundqvist, I. Mas, G. Karlsson, “Edge-based Differentiated Services”, Technical Report, KTH, April 2005. URL: http://web.it.kth.se/~hen/EBDS_TR.pdf