Quality of Service and the End-to-End Argument

Gunnar Karlsson and Ignacio Más

Abstract

Audio-visual services are now commonly used on the Internet. Many of them are based on batch downloading of contents for later replay. Real-time interactive and streaming services are now rapidly becoming popular. These services would benefit from quality of service if it were widely provided. The purpose of this article is to show how QoS solutions may be introduced incrementally. Quality is obtained by means of a probe-based admission control that can be exerted outside the network. The introduction of QoS starts from self-admission control in the application layer, followed by transport layer service differentiation. These two steps do not require any change to the network. If motivated, scheduling for service class separation in the network may be added. We show by proof of concept how QoS may be provided in agreement with the end-to-end argument. The three steps of our proposal are compared and discussed with respect to the possibility of deployment.

The last 20 years of networking research have provided the world with a multitude of quality of service (QoS) solutions as well as three major service architectures: the traffic classes defined for asynchronous transfer mode (ATM) within the International Telecommunication Union (ITU) and the ATM Forum, and the integrated (IntServ) and differentiated services (DiffServ) architectures defined by the Internet Engineering Task Force (IETF). Most proposed QoS solutions relate to various problems within one of these three QoS architectures. Despite the large research and standardization efforts, there are only limited QoS offerings from Internet service providers, and most, if not all, pertain to DiffServ for virtual private networks [1].

There have been many viewpoints on the lack of deployed QoS solutions. Bell points to the little attention QoS researchers have given the culture of network operations to avoid complexity and anticipate failure [2]. He states that the addition of QoS mechanisms to the network would introduce new potential failure modes that might be difficult to debug. This has to be weighed against the uncertain benefit of providing QoS support in the network; this view is shared by Davie in [1].

Quality of service functions have traditionally been associated with the network layer (e.g., admission control, scheduling, and policing). So the question arises whether it is possible to provide some form of QoS without changes to the network or network operations. Lifting the QoS mechanisms out of the network to higher protocol layers would have the following advantages:

- QoS is provided from host to host.
- The QoS mechanisms may develop independent of the network.
- The management and operation of a network remains unaffected.

The design of the Internet has relied on a powerful argument in favor of assigning functions to the layers of a protocol architecture, the so-called end-to-end argument [3]: "*Functions placed at low levels of a system may be redundant or of little value when compared with the cost of providing them at that low level.*" The contribution of this article is to show how QoS may be provided from hosts in agreement with the end-to-end argument.

We present how to introduce QoS solutions incrementally, first at the application layer, then at the transport layer, and finally at the network layer. The application layer solution can be introduced by application developers without standardization, but it also gives the least benefit: it prevents users from setting up sessions when the network is congested. The transport layer solution would require standardization, and provides differentiation into two distinct traffic classes; the differentiation may be reinforced by scheduling at the network layer. The necessity of QoS support in the network layer is left open, but the solution is presented to show that it is possible to add such support if the quality gain should justify it. We discuss the implications it would have on network operation. We build on our previous work and the work of others to show that the solutions are practicable.

We structure our presentation as follows. The definition of and need for QoS is presented. We present the concept of probe-based admission control used in all three case studies; these studies are presented, starting from the application layer, followed by the transport and network layers. The QoS solutions at the three layers are compared. We discuss the network operator roles for the alternatives in the proposal and offer some deployment perspectives. We present related work and then conclude the article.

Quality of Service

A definition of QoS is given in ITU Recommendation E.800: "*The QoS is the collective effect of service performances which determine the degree of satisfaction of a user of the service.*" Satisfaction might be guaranteed, but is not implicit in the definition. It is not trivial to guarantee a level of satisfaction to a user. There are many performance aspects that affect user sat-

0890-8044/07/$20.00 © 2007 IEEE

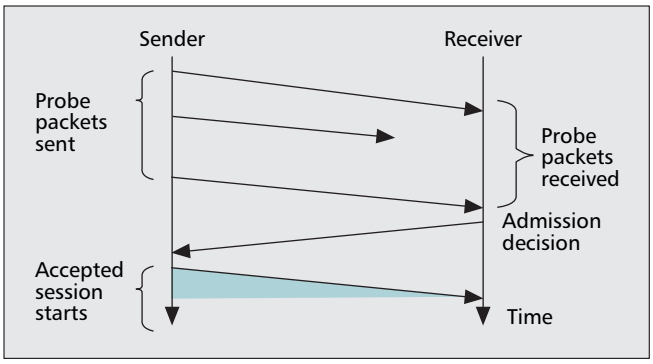


Figure 1. *The steps of probe-based admission control. The decision may be made by the sender if the receiver feeds back the results of the probing. Blocked sessions back off a random time before trying anew.*

isfaction even when we limit ourselves to network performance: traffic control behavior, route changes, sensitivity to service attacks, and latency in fault handling, for instance. The concept of QoS is often associated with traffic control behavior, and the associated performance metrics are packet loss and delay; this is also the restricted scope of QoS herein. A guarantee solely with respect to traffic control is of limited value to a user since it is the collective effect of all performance aspects that determine user satisfaction. A guarantee spanning all performance aspects might, however, be unattainable. We do not consider QoS guarantees for this reason.

We refer to the communication between two or more users as a *session*; a real-time service session may, for example, carry a voice conversation, a videoconference, or the streaming data of a film or an audio track. The quality aspect we are interested in pertains to the throughput variations to which a data stream is exposed in the network. For instance, TCP attempts to share the rate of a bottleneck link fairly over all ongoing TCP sessions, which leads to throughput variations when sessions enter or leave and when the total data rate of the sessions changes due to additive-increase multiplicative-decrease congestion control. TCP implements only one possible control policy for reacting to congestion. The complementary policy we propose herein is to block new real-time sessions when there is congestion, and to allow admitted real-time sessions to remain unresponsive to congestion. This policy results in a time-wise sharing of a bottleneck by blocking some sessions in order to ensure that the sessions admitted have sufficient throughput to sustain the data rates for the sessions.

Our motivation for this policy is based on human perception of real-time streaming and interactive services with audio-visual signals. When audio and video signals are encoded, there is a trade-off between the distortion in the signal and the resulting bit rate: the lower the bit rate, the higher the distortion. The throughput of the session with the encoded data should match the data rate of the signal; packets will otherwise be delayed and lost in the network. An alternative to packet loss and delay is to quench the sender based on feedback from the network. This results in either higher distortion in the encoding if the bit rate is reduced, or higher delay and interruption of the playout at the receiver if the data is queued up. The signal at the receiver will in all cases vary in quality due to throughput variations.

It is important to recognize that human perception favors consistency. Noticeable quality variations are distracting and reduce user satisfaction even in cases where the average quality is high. This is the rationale for our interest in limiting the throughput variations for audio-visual sessions. We do this by means of preventive congestion control in the form of probe-based admission control.

Probe-Based Admission Control

Probe-based admission control is based on the transmission of a probe, which consists of a stream of probe packets that are used to infer the state of a network path (Fig. 1) [4]. The probe lasts a few seconds and typically constitutes only a small addition to the total session length. The probe rate (denoted r_{pr}) is constant, and the session will not be allowed to exceed it when admitted. The idea is to use the probe to estimate the effect the session would have on the network if it were allowed to start. One might argue that probing at the highest rate of the session is pessimistic. However, neither the distribution nor the mean of the data rate of the session can be known in advance when the rate is variable. A peak rate limit can more readily be established for a streaming data source and is easily enforced by a token bucket.

The receiver measures the loss and possibly the delay for the probe. A new session is permitted to continue after the probing if the result of the measurement is found to be acceptable according to a prescribed admission criterion; the session is blocked otherwise. Depending on implementation, the decision may be made by the receiver or the sender after it has received feedback on the result of probing. The choice of admission criteria depends on the protocol layer where the admission control is used. This is presented later.

The transmission of a new probe after a blocked session setup attempt is controlled by backoff and eventually by a limit on the number of attempts that can be made. The backoff randomizes new setup attempts if the failure was due to too many simultaneous probes. The probe packets may contain both session and control data. The control data is included in several probe packets for reliability and can include parameters for the configuration of the receiving application as well as information regarding the session, such as the identity of the calling party, whether it is bidirectional, and the intended subject of the session. The session data can be a greeting or ring tone, for instance, which is played out when the session has been established. Since the probe packet stream is sent at a constant rate, it may be used for clock synchronization and allowing the jitter removal system to settle into steady state.

Probe-based admission control also works for multicast sessions. The operation is identical to that described above if all receivers are waiting for the source to start transmitting; the receivers will first receive probe packets and then make an admission decision. Those that cannot receive the data with sufficient quality leave the multicast group. It is of course necessary that the decision be based on a criterion that allows it to be made at the receiver. If a receiver joins an ongoing multicast session, it would not receive a probe. Its prediction would be based on the data rate of the session, which might be lower than the probe rate. This means that a decision to remain in the group is less reliable than if it were based on the probe, but a decision to leave the group would be more reliable.

Both unicast and multicast routes can change during an ongoing session. The change is not noticeable if the new path also supports the necessary quality for the session. The new path might be congested, or become congested due to the rerouting. The receiver would in this case terminate the session (and notify the users), and request the sender to re-establish the session with a new probing phase. Rerouting caused by traffic engineering typically results in lower congestion throughout the network. Hence, the main risk for having sessions terminate due to route changes is associated with network faults, for which the termination of some sessions might be necessary.

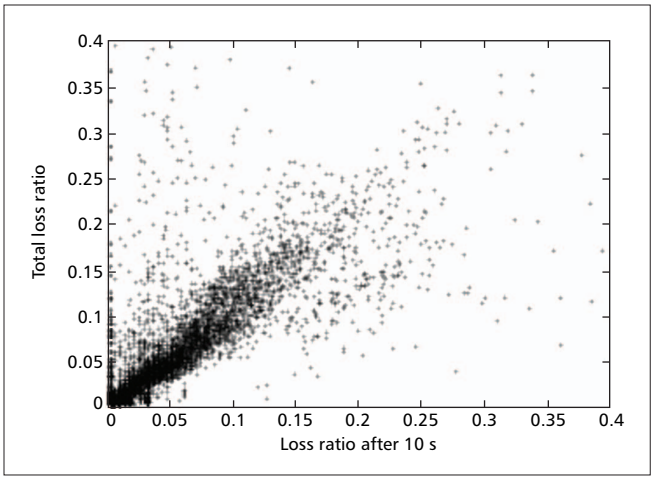


Figure 2. *The loss ratio of a call measured over the initial 10 s and over the whole duration (70 s). Each cross corresponds to one call.*

Application Layer QoS Support

The purpose of the first step to support interactive and streaming services is simply to avoid establishing sessions when there is some indication that the quality would not meet the user's expectation.

Self-Admission Control

Measurements of the quality for voice over IP (VoIP) calls in the Internet show that it varies with time [5]. The temporal variations turn out to be quite slow with respect to the timescale of the calls. Figure 2 shows the loss rate for the first 10 s and for the entire call of 70 s for each of approximately 9700 calls that had at least one packet lost. The estimated correlation between the initial phase and the entire call are 0.8 after 1 s and 0.9 after 4 s, with only minor increases up to 10 s (Fig. 3).

This strong correlation allows the quality for a session to be predicted with reasonable certainty after a few seconds. Hence, it is possible to base an admission decision on a short probe [5]. The advantages of this are:

- The network will not be loaded by the new session when it is already congested.
- The called party in an interactive session would not be disturbed by a session that risks being aborted due to poor quality.
- For streaming, the receiving party may avoid starting to listen to or view a session that is likely to be rendered useless by poor throughput.

Self-admission control is realized as follows. The sender probes the path to the receiver, as explained above (Fig. 1). The receiver measures the probe loss and estimates the resulting quality the receiving party would experience. This could be, for instance, the perceived quality for the given application [6]. The receiver will block the session if the quality does not meet a desired target value; it will accept the session otherwise and alert the user.

Two-way sessions are initiated by the first received probe packet and carried out by the called application, which probes the return path in the same manner.

Remarks

Application layer self-admission control only prevents network congestion if all applications use some form of congestion control (either self-admission control or any other form of congestion control). If all applications were to use self-admission control, it is the application type with the lowest quality target that can drive the network closest to saturation. This means that ongoing sessions with higher quality expectations could be disturbed, maybe to the point of becoming useless; the sessions would then be aborted by users. As stated above, admission control still prevents new sessions from being established in highly loaded conditions.

All design choices for self-admission control are application-specific, and the application designer is the one to implement it. Although it is much simpler to implement than feedback congestion control, it might be desirable to provide probe-based admission control at the transport layer as a generic service to all applications that might benefit from it.

Self-admission control is implemented in the open source VoIP software Sphone [7].

Transport Layer Service Differentiation

Probe-based admission control at the transport layer allows quality of service to be provided in two generic traffic classes: *batch* and *streaming*. They are qualitatively different due to the control behaviors associated with the classes. The batch class is controlled by TCP, and the streaming class uses probe-based admission control. The requirement for the service differentiation to work is that all sessions to enter the network are congestion controlled.

Although admitted streaming sessions are allowed to be unresponsive to load variations, probe-based admission control makes the *aggregate* of streaming sessions responsive to load variations by blocking new sessions to prevent the load from increasing; the load is reduced when sessions eventually terminate. This is illustrated in Fig. 4, where new sessions start by probing: the first three sessions are successful and continue with the data transfer; the fourth and fifth probes are unsuccessful, and the sessions are blocked. Finally, the sixth session is successful since one session has terminated and made capacity available.

Fair sharing of bottleneck capacity between sessions of the two classes may be achieved without scheduling in the network by choosing the admission criterion appropriately, as explained below. Service differentiation is then entirely implemented in the hosts.

General Idea

The sender issues a probe, and the receiver acknowledges all received probe packets. This allows the sender to estimate both the round-trip time (RTT) and loss probability for the path. The estimates of the RTT and packet loss probability allow computation of an equivalent TCP throughput, rTCP, for instance, using the formula in [8],

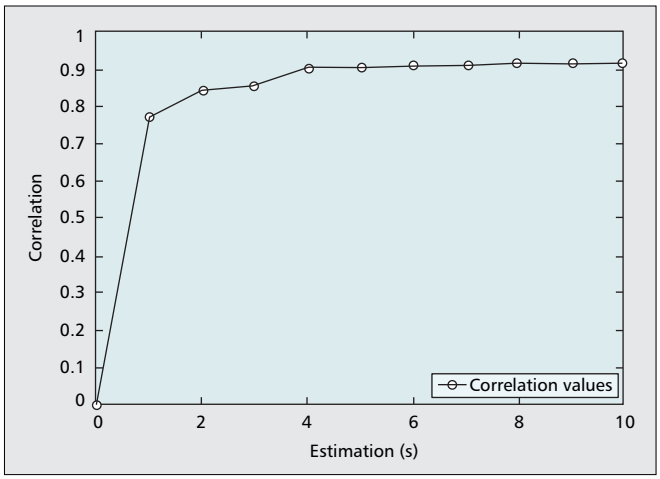


Figure 3. *Correlation for the estimate of the loss for an entire call, based on the first 1 to 10 seconds of the call.*

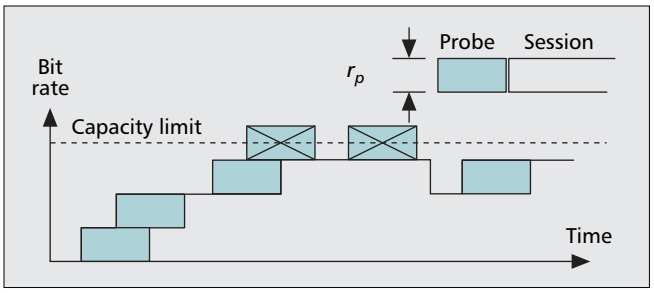


Figure 4. *Illustration of aggregate rate control by means of probe-based admission control. The marked probes are not successful in establishing new sessions. The probe is sent at the peak rate of the session,* r_{pr}. *The probes and sessions are shown as constant rate flows.*

$$r_{TCP} = \min\left(\frac{W_{\max}}{RTT}, \left(\begin{array}{c} RTT\sqrt{2bp/3} + T0 \times \\ \min(1, 3\sqrt{3bp/8})\, p(1+32p^2) \end{array}\right)^{-1}\right),$$

where p is the packet loss probability, W_{max} the maximum window size, RTT the round-trip time, b the number of packets acknowledged by a received ACK (usually 2), and $T0$ is the timeout value. Probe-based admission control ensures that the *aggregate* of admitted streams is responsive to congestion in a way that is fair to TCP by the decision policy to accept the session if $r_{pr} \le r_{TCP}$ and block it otherwise.

A remaining issue concerns the quality admitted streaming sessions could expect. Sessions will be disturbed by new streaming sessions that probe the network for admission, and by old and new TCP sessions that probe for available capacity (by additive increase and slow start, respectively). The admitted sessions should therefore be protected by forward error correction with error control codes that are strong enough to withstand the loss caused by the disturbing sessions. The loss estimate from the probing can be used to calculate the appropriate level of redundancy necessary for insulating the sessions in this way [7]. The redundancy rate is included in the decision of whether or not to admit the session.

Probe-based admission control can also be used for multicast sessions. An admission policy based on estimation of RTTs would require each receiver to estimate it, for instance, by means of *pings* to a rendezvous point. This solution might be acceptable for a low arrival rate of new receivers (note that group size is not important); an admission policy independent of RTTs is preferable, however, since the receivers could then make the decision.

Remarks

Note that the controls for the batch and streaming classes are qualitatively different and thereby provide distinct services to applications; it cannot be argued that one traffic class or the other is better. TCP provides a lossless service where throughput variations and retransmissions lead to uncontrolled delay; probe-based admission control allows the sender to use a fixed amount of network capacity, and thereby controls delay and loss while introducing uncontrollable blocking. One service class cannot starve the other, and they share capacity in a reasonably fair manner [9].

Probe-based admission control could be implemented within RTP/UDP or incorporated in the datagram congestion control protocol [10]. Compliance with probe-based admission control may be verified by passive measurements close to the receivers [11]. TCP compliance could be monitored in a similar manner.

Network Support for Service Differentiation

Isolation of the batch and streaming classes may be provided by scheduling in the network in order to further differentiate services. The scheduling would allow an operator to control the blocking probability for the streaming class and the average throughput for the batch class. The queuing system for this is illustrated in Fig. 5, where the two classes are allocated partitions C_b and C_s, respectively, of the link capacity, denoted C_l. Probe packets are sent at low priority within C_s, and packets of admitted sessions are sent at high priority [4]. The scheduling limits the rate of the streaming class to C_s. The elastic batch class is guaranteed a capacity share $C_l - C_s$ and may in addition use all of the unused capacity within C_s. The necessary rate-limited priority service can be configured on many QoS enabled routers. The three buffers may be dimensioned for their particular purposes: the batch data buffer for low loss and the high-priority buffer of C_s for low delay; while the low-priority buffer should be minimal, simply providing storage of a few simultaneously arriving probe packets.

A streaming session is admitted if there is enough capacity available for its probe within link partition C_s [4]. This is measured by the estimated loss probability for the end-to-end path. There must be a single target loss level for the admission of all hosts using the service class. There is no need to consider TCP fairness in the admission decision since the classes have been separated by scheduling.

The advantages of network support are:

- Ongoing streaming sessions are disturbed by neither batch transfers nor probes.
- The operator can control the blocking probability for streaming sessions by adjusting the capacity allocation, C_s.
- The delay may be reduced by configuring short buffers for the streaming class.

Multicast admission control follows the earlier description with one modification. The probe is sent on one multicast group that has low priority, and the admitted session is sent on a group with high priority. A new receiver first joins the probe group. If admitted, it leaves the probe group and joins the high-priority group; if blocked, it leaves the probe group and makes a new attempt after a random backoff time [12].

Comparison of the Proposals

Simulations

We have shown how admission control may be implemented in the application and transport layers, and how limited network support may be added in order to isolate the service classes from one another. We now show a comparison of the three alternatives

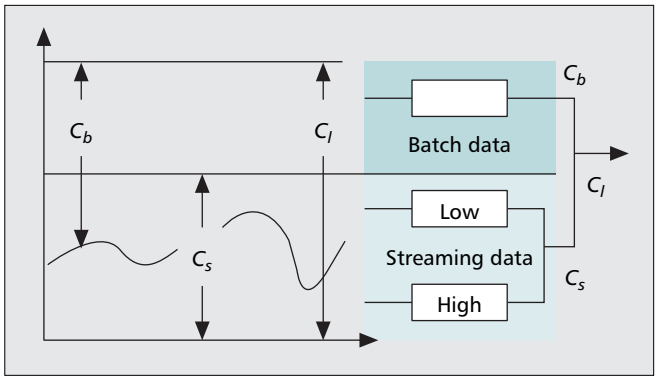
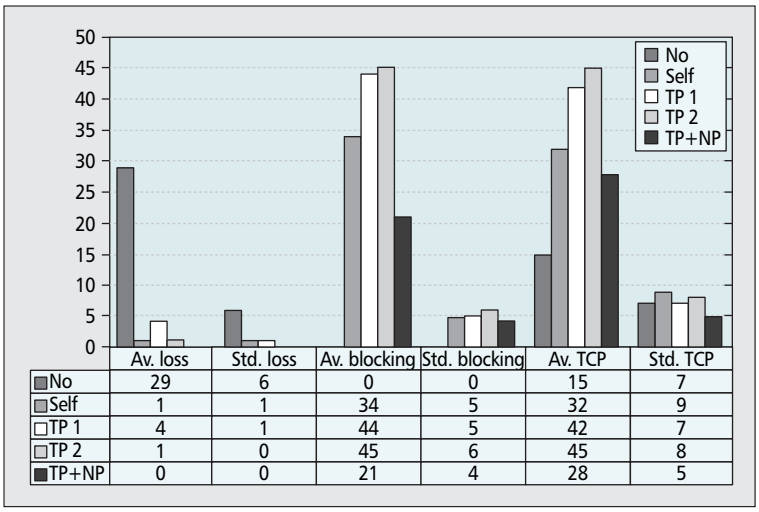


Figure 5. *The link capacity* C_l *is split into two parts:* C_s *for admission-controlled streams and* C_b *for elastic batch transfers. Probe packets are sent at low priority and forwarded if there is capacity available within the partition* C_s. *Batch sessions are guaranteed a minimum capacity share of* $C_l - C_s$ *and may in addition use all capacity not used by streaming sessions.*



	Av. loss	Std. loss	Av. blocking	Std. blocking	Av. TCP	Std. TCP
No	29	6	0	0	15	7
Self	1	1	34	5	32	9
TP 1	4	1	44	5	42	7
TP 2	1	0	45	6	45	8
TP+NP	0	0	21	4	28	5

Figure 6. *Comparison of loss, blocking, and TCP throughput for the three scenarios, with two policies for the transport layer case. The averages and standard deviations are computed over 30 simulation runs for each case. All measures are given in percentage.*

for a limited setting. A bottleneck link with 10 Mb/s capacity is shared between batch and streaming sessions. There at 10 batch sessions, which use TCP. Each batch session is peak rate limited to 500 kb/s (as through a residential access link); they enter the link one by one in the first 10 s of the simulation. They repeatedly download a 5 Mbyte file, creating a load of up to 5 Mb/s.

The source model for the streaming sessions consists of a two-state Markov chain with 250 kb/s rate in the on state, with an average rate of 100 kb/s; the average on and off times are 200 ms and 300 ms, respectively. Probing time is 2 s at the peak rate, corresponding to 976 probe packets of 64 bytes each. Each session lasts for 30 s and then restarts by probing. There are 80 sources offering an average load of 8 Mb/s. The simulations were run for 1000 s each; the first 100 s, which allowed the load to settle, has been excluded in the measurements. Each case has been run 30 times (Fig. 6).

There are five simulated cases: no admission control (No); self-admission control (Self) with a 1 percent loss threshold for admission; two versions of transport layer admission control, the first with the TCP fair policy (TP 1) and the second with two admission criteria: TCP fairness combined with a loss threshold of 1 percent; the call is admitted if both criteria are met (TP 2). Network scheduling is included in the last case (TP+NP) for which the admission criterion is again 1 percent loss, and C_s is set to 10 Mb/s. The columns represent, from left to right, the average and standard deviations of loss, blocking, and TCP throughput (i.e., the TCP share of the link). The fair share of capacity for TCP is 5/13 (38 percent).

Results

The uncontrolled situation simply illustrates severe overload: streaming sessions are freely admitted (no blocking) and experience severe packet loss, while they unfairly out-compete TCP sessions. With the addition of self-admission control, the situation improves. The loss in streaming sessions has been reduced at the expense of blocking; TCP sessions get closer to their fair share.

The admission policies for the transport layer differentiation affect the quality of streaming sessions: the TCP fair policy only bounds the loss through the formula for equivalent TCP throughput, and the resulting loss turns out to be higher than in the self-admission case; TCP fares well. The combined fairness and explicit loss threshold for admission provides a loss target, which is met, but blocking goes up and TCP is favored. Finally, the inclusion of network support allows the blocking to be controlled vis-à-vis the TCP share of the link capacity.

The conclusion is that self-admission control can be helpful since the admitted streaming sessions mostly meet the target of at most 1 percent loss. Depending on the streaming load and the particular decision level, the balance between the streams and the batch classes may tip in either direction (here batches are disadvantaged). However, the important point is that both classes see improved quality with the introduction of self-admission control. A balance may be achieved at the transport layer by including TCP fairness as part of the decision criteria for probe-based admission control. This is the only advantage in terms of performance over self-admission control; from a system's point of view, it might be advantageous to also have the two generic services provided to applications, rather than expecting each streaming application to have its own admission control. The network support adds a control mechanism with respect to blocking of streaming sessions. It is up to an operator to decide whether it is a motivated part of QoS provisioning.

Deployment and Operator Role

A desirable property that follows from the end-to-end argument when applied to QoS is the separation of roles between network operation and the host functions: the operator is incognizant of the differentiation at the transport layer and the admission control at the application layer. Self-admission at the application layer may readily be incorporated into software for streaming and conversational applications; it does not depend on standardization. This freedom is also the solution's weakness: it cannot be mandated and reinforced by network providers in the Internet, who might value the congestion control aspect of it. Admission control protects users of the application from establishing sessions during bouts of congestion. This advantage to the user does not depend on a general consensus on using it. Competitive pressure in the market could result in developers including the function in new software releases for the sake of their customers.

Transport layer differentiation provides two generic service classes to applications. There is no fixed configuration necessary for the coupling of application to service class: the transfer of a large batch of data might be efficiently made at a fixed rate after probing and admission control; for a small batch this could be inefficient since the probe phase might be a substantial part of the total transfer time. Admission control requires a decision policy for sharing with TCP (hence it is not value neutral, in the phrasing of [13]). If policies are well defined, they can be monitored [11]. Introduction of this solution requires standardization, for instance, by including it in the datagram congestion control protocol [10]; deployment thereafter may occur with updates of operating systems. However, operators have one role: to enforce that the traffic entering their networks is congestion controlled by the hosts, unless they can trust that it is (e.g., the hosts may belong to and be under control of the network operator).

The operators' role is further affected by QoS when they offer service class differentiation in the network. In addition to the aforementioned enforcement of congestion control in the hosts, operators have to manage the capacity allocation for the streaming class. There is still a separation of roles between hosts and networks. Hosts manage the admission decisions of sessions. These decisions are time critical, but also fully distributed, so there is good scaling. The network manages capacity allocation to control the blocking probabilities of the links, which occurs less frequently than admission decisions. The allocation might therefore be centralized without scaling problems.

If there is a clear and explicit target by the network operator to keep the blocking probability below a declared value, users ought to pay for that guarantee since it elevates the streaming class over the batch class. However, the operator might equally well manage capacity allocation to balance the throughputs of the two classes. In that case there is no reason to charge the streaming class unless the batch class is also charged.

Network support for QoS affects peering between operators: they have to agree on the policy for resource allocation, whether it is a guaranteed upper limit for blocking or fairness between classes, and they must trust each other with respect to enforcement of congestion controls. This might lead to disputes that impede deployment [13]. The possibility of offering low delay for streaming, isolating the classes well, and explicitly being able to control capacity sharing between them might provide enough incentive to introduce the complexity, especially when the QoS solutions in the higher layers have become widely used and streaming services thrive. Network support forms the last step in the introduction of QoS.

Related Work

Our design of probe-based admission control is not the only one possible. We base the admission decision on a step response, but other forms of probing are conceivable that also include network support (scheduling or ECN); see [14] for a comparative survey. Work related to the one we present is the TCP friendly rate control [15]. It does not attempt to avoid or limit rate fluctuations, only to smooth them temporally. The idea of having two distinct services that cannot be ranked as better and worse comes from the alternative of best effort [16]. End-to-end arguments have been discussed by Moors, who rejects them with respect to QoS and congestion control [17]. His arguments are that congestion is a network phenomenon; thus, the network is responsible for isolating endpoints that offer excessive traffic, and endpoints cannot be expected to act altruistically to limit congestion. Our counterargument to the first point is that congestion is only a network manifestation of how endpoints inject traffic into the network, and probe-based admission control and TCP indeed reduce the risk of congestion. We agree with Moors that the endpoints might not be trustworthy and have therefore studied how their behaviors can be monitored [11]. The "tussle in cyberspace" discussion is highly relevant to QoS provisioning [13]; the conflicts are potentially lessened by QoS provided from the endpoints of the communication sessions.

Final Remarks and Conclusion

A common view is that quality of service with respect to statistical multiplexing is solved by overprovisioning network capacity. Capacity planning is, of course, a necessary long-term task for operating a network in order to keep capacity above demand. However, the cycle is long: traffic volumes have to be measured and future demands forecast; necessary capacity increases in the network must be determined; additional capacity must be leased or procured and installed. Mistakes in forecasting make the network susceptible to long-term congestion on one hand or make the operation uneconomical due to unnecessary investments on the other hand. It is not easy to properly define what constitutes *overprovisioning*, and as sole control strategy for QoS, it is coarse and slow.

Admission control is hence a vital complement to TCP since it suits real-time services and protects the network from overload in the short term by blocking load surges; it remains almost transparent if the network can meet all demands for establishing sessions. The cost of controlled admission is the probing phase.

The well established design principle, the end-to-end argument, is applicable to the traffic control issues that need to be resolved to provide QoS in the Internet. This article has presented how solutions may be introduced: first by application developers implementing self-admission control; second by the standardization of congestion control in addition to TCP at the transport layer and its inclusion in operating systems' protocol stacks; and third by service class differentiation in the network. Network support is scaleble since time-critical admission control is distributed out to the hosts, while capacity allocation is done by the network on a longer timescale.

It is our hope that this presented strategy may end the impasse in providing quality of service.

Acknowledgment

We would like to thank Drs. Viktória Fodor, Olof Hagsand, and Henrik Lundqvist as well as Mr. Ian Marsh for their collaboration on the work described herein. Gunnar Karlsson is also grateful to Prof. Bernhard Plattner of ETH Zurich for hosting during a sabbatical when this article was written. Dr. Ulrich Fiedler of ETH Zurich and the anonymous reviewers provided comments that greatly helped to improve the original manuscript.

References

[1] B. Davie, "Deployment Experience with Differentiated Services," *Proc. ACM Wksp.: Revisiting IP QoS*, Karlsruhe, Germany, Aug. 25–27, 2003, pp. 131–36.
[2] G. Bell, "Failure to Thrive: QoS and the Culture of Operational Networking," ibid, pp. 115–20.
[3] J. H. Saltzer, D. P. Reed, and D. D. Clark, "End-to-End Arguments in System Design," *ACM Trans. Comp. Sys.*, 1984.
[4] I. Más and G. Karlsson, "Probe-based Admission Control for a Differentiated-Services Internet," *Comp. Networks*, Apr. 2007.
[5] O. Hagsand *et al.*, "Self-Admission Control for IP Telephony Using Early Quality Estimation," *Proc. IFIP Networking*, Athens, Greece, May 2004, Springer LNCS vol. 3042.
[6] V. Grancharov *et al.*, "Non-Intrusive Speech Quality Assessment with Low Computational Complexity," *Proc. Int'l. Conf. Spoken Language Processing*, Sept. 2006.
[7] O. Hagsand, I. Marsh, and K. Hansson, "Sicsophone: A Low-Delay Internet Telephony Tool," *Proc. 29th Euromicro Conf.*, Belek-Anatalya, Turkey, Sept. 2003, pp 189–97, http://www.nada.kth.se/~olofh/sphone/.
[9] H. Lundqvist, I. Más, and G. Karlsson, "Edge-Based Differentiated Services," *Proc. IWQoS*, Passau, Germany, June 2005.
[8] J. Padhye *et al.*, "Modeling TCP Reno Performance: A Simple Model and Its Empirical Validation," *IEEE/ACM Trans. Net.*, vol. 8, issue 2, Apr. 2000.
[10] E. Kohler, M. Handley, and S. Floyd, "Datagram Congestion Control Protocol (DCCP)," IETF RFC 4340, Mar. 2006.
[11] I. Más, J. Brage, and G. Karlsson, "Lightweight Monitoring of Edge-Based Admission Control," *Proc. IZS*, Feb. 2006.
[12] I. Más, V. Fodor, and G. Karlsson, "Probe-based Admission Control for Multicast," *Proc. IWQoS*, Miami Beach, FL, May 2002.
[13] D. D. Clark *et al.*, "Tussle in Cyberspace: Defining Tomorrow's Internet," *IEEE/ACM Trans. Net.*, vol. 13, issue 3, June 2005, pp. 462–75.
[14] L. Breslau *et al.*, "Endpoint Admission Control: Architectural Issues and Performance," *Proc. ACM SIGCOMM*, Stockholm Sweden, Sept. 2000.
[15] S. Floyd *et al.*, "Equation-based Congestion Control for Unicast Applications," *Proc. ACM SIGCOMM*, Stockholm, Sweden, 2000.
[16] P. Hurley *et al.*, "ABE: Providing a Low-Delay Service within Best Effort," *IEEE Network*, vol. 15, no. 3, May 2001.
[17] T. Moors, "A Critical Review of End-to-End Arguments in Systems Design," *Proc. IEEE ICC*, 2002.

Biographies

GUNNAR KARLSSON (gk@ee.kth.se) has been a professor at KTH, the Royal Institute of Technology, since 1998. He previously worked for IBM Zurich Research Laboratory and the Swedish Institute of Computer Science. He holds a Ph.D. from Columbia University. He has been a visiting professor at EPFL and ETH Zurich in Switzerland, and TKK in Finland. His current research relates to quality of service and wireless content distribution.

IGNACIO MAS has been working on IPTV development at Ericsson Research since 2005. He is a doctoral student at KTH, and holds an M.Sc. in telecommunications from Polytechnical University of Madrid (1999). His research interests are in multimedia networking, quality of service, and operating systems.