# The DIY Approach to QoS

***Gunnar Karlsson, Fredrik Orava***
KTH Teleinformatics, Electrum 204
164 40 Kista, Sweden
{gk, fredrik}@it.kth.se

*There is much attention given today to the provisioning of service quality in the Internet. In this position paper we advocate the use of an architecture that improves the integrated-services architecture and allows it to be brought into a differentiated-services context. The main idea in our work is to move functionality and per-session state from the network nodes out to the hosts. Such decentralization improves scalability and simplifies QoS for multicast sessions.*

## *Introduction*

The need for quality of service beyond best effort in the Internet might not be a matter of dispute: There clearly exist communication contexts for which users only can tolerate limited amounts of delay and information loss. Although an application may be designed to conceal losses and delay variations, there is nevertheless a limit to how much it is able to improve an unreliable network service. Delay and error control functions are vital in networked applications and, yet, some form of traffic control is needed to limit the degradations in the network. This position paper proposes an architecture for quality of service provisioning that owes much to the two general structures proposed for the Internet: the integrated-services (intserv) [2] and the differentiated-services (diffserv) architectures [1]. Our aim has been to remove the need for a general-purpose signaling protocol such as RSVP and to move all connection states out of the network nodes and into the end-systems. This is the do-it-yourself approach to QoS provisioning. In a sense, we wish to retain the original Internet belief that the network is responsible for forwarding packets while the end systems are responsible for the quality of the transfer.

Our architecture has borrowed the set of service classes from intserv: a guaranteed service based on deterministic multiplexing, a controlled-load service with measurement-based admission control, and a best-effort service. The best effort would be similar to today's service with the possible option of separating UDP and TCP into separate classes, as well as separating present-day TCP (Reno and its ilk) from a future TCP with better congestion and error controls. The separations eliminate potential fairness conflicts between the rate-based congestion control that would be suitable for UDP-based services and the window-based controls of present and future TCP versions (best effort is not further considered in this paper). The differences in our actual design of the guaranteed and controlled-load services are substantial when compared to the original proposals (see [6, 7]). We will discuss them briefly in the following sections.

## *Guaranteed service*

The service laid down in RFC 2212 is based on deterministic bit rate limits on the flows [6]. Each flow should receive sufficient buffer space and service rate in a node to guarantee a maximum queuing delay and absence of packet loss due to contention. Token-bucket descriptors that consist of three values describe the flows: peak rate (possibly infinite), token rate which bounds the average rate, and the burst size which limits how much a flow may exceed the token rate. The guaranteed service requires some form of signaling to setup the reservation; it has been assumed to be provided by RSVP [3] which however appears unwarrantedly complex for this simple task.

The implementation of the service normally requires scheduling in the network nodes to ensure that each flow receives a share of the link capacity that is sufficient for achieving its delay bound. A router consequently needs to maintain states for all flows that pass it and would also need buffers to prevent losses of traffic bursts.

It is important to note that the sender's permitted ability to exceed the token rate for bounded periods of time does not help reducing the maximum delay (albeit the average delay goes down which however leads to larger delay jitter). This can be formally shown by network calculus but the rationale is simple: the maximum delay is based on the case where a flow only receives its reserved rate on a particular link. The node feeding that link has to smooth out the bursts. It is not known which link will become the bottleneck, and all nodes must therefore reserve buffer space in anticipation of smoothing the bursts of a flow. A simpler and better idea would be to limit the flow to the token rate at the access: the smoothing would then occur before the flow enters the network and the network nodes could thereby avoid buffers for the source-generated bursts. The delay bound would still be the same for the given token rate. For this reason we require the bit rate of a flow to be limited by an unsurpassable upper bound [5].

Per-flow scheduling may be too complex to implement in fast routers, and it would be better to

schedule all flows of the service class together. If this is done by means of a work-conserving scheduler then packets may clump together and thus the nodes must provide buffer space for the clump. The amount depends on the number of hops a flow traverses, which complicates the dimensioning. Clumping can only be prevented by a non-work conserving scheduling that reshapes the aggregate flow. One such scheduler is proposed in [5].

Now, given that we have upper-bounded flows at the network ingress and reshaping in all network nodes it follows that the reservation state for the service is simply the sum of the peak rates of the flows on the link. If the reservation state is held in a shared variable that all senders may modify, then a sender simply needs to add its desired rate to the state variable in order to make a reservation. Analogously, the rate is subtracted when the reservation is rescinded. Attempts to write out of the shared variable's range are detected and the corresponding reservation is thereby blocked. This simple procedure is the basis for our service-specific signaling that we believe makes a guaranteed service simpler and more scalable.

## Controlled-load service

The purpose of this service is to prevent congestion from occurring [7]. This is in contrast to best-effort service where congestion is resolved rather than prevented. The original CLS is based on a peak-rate description of a flow with measurement-based admission control in all network nodes. This type of admission control adds complexity to the network nodes for the measurements, their analyses and the decision process. The service also requires signaling between the network nodes. It has never been clear to us what decision criterion a node would use for the admission since the desired operational level is not specified (i.e., what is considered a non-congested operational level?).

Our goal has been to provide an admission control in order to protect ongoing sessions from new sessions [4]. However, we do not want to burden the routers with the possibly complex measurements and analyses, and we do not want signaling more than between the host and the access gateway. We accomplish this by requiring each host to probe the network to see if there is capacity available for a new session. It is only allowed to use the service if the probing is successful, otherwise it is blocked and will have to wait a random time before probing again (or it may proceed with the session at best effort).

The service requires its own capacity partition on the links in the network and it gives an unsurpassable limit for the traffic. This is needed to ensure that the probes only estimate unused capacity that is available to the controlled-load service and should not include the remaining capacity on the link used by guaranteed and best-effort services. Unused capacity is however fully available to best-effort traffic so there is no waste incurred by the allocation. The limit can be provided by the same non-work conservative scheduling that we use for our guaranteed service [5]. The network nodes must recognize two priority levels within the service: a high priority for the established sessions and a low priority for the probes. The probe packets can therefore not disturb ongoing sessions, and they are only forwarded if there is capacity available within the service's capacity partition. The allocation of capacity to the service class is done by the network management and determines the blocking probability for calls.

A probe is a stream of packets sent at the maximal bit rate that the sender wishes to use. The maximally allowed rate for a session should be a small fraction of the capacity allocated to the service class on each link to ensure that statistical multiplexing works well. The receiver measures the loss rate of the probe and decides whether the session should be established. It notifies the sender about the decision. The service requires only small buffers in the network for packet-scale queuing. This has the advantages that the delay is kept at a minimum and the admission decision can be based on the loss rate solely (there is virtually no information in the delay variations). It also simplifies the analysis of the service since one may rely on a bufferless assumption. There is only one service level offered and the end systems must match that to the application's need by means of forward (or backward) error correction.

The sender is allowed to establish a session if the packet loss ratio of the probe is below a prescribed level. All parties sharing the service must use the same level. The procedure relies on trust between network and host. One means of establishing it would be the use of certified and tamper-proof implementations of the protocol. This remains a research issue, however (the measurement could be conducted by the exit gateway if the end system is not trusted).

The admission control works well for multicast sessions. It might also suit mobile hosts since there is no per-session state in the network that needs to migrate due to changes in the call routing.

## The DIY approach and diffserv

Intserv is not the main focus any longer for the work towards introduction of QoS in the Internet. Instead diffserv has been proposed as a lightweight architecture for proving QoS [1]. It relies on the newly defined 6-bit DS-field in the packets to indicate the service class to which a packet belongs. A risk with diffserv is that the number of service classes will grow: It looks innocuous to assign a value of the DS-field to a particular service without regard to the ensuing complexity incurred in the routers. If it should be possible to build routers for the operational range of 10 Gb/s per port, then care has to be given to the service offering and the mechanisms needed in the routers to implement them. By nailing down three classes we may avoid further definitions of classes that bring marginal advantages but add substantial complexity.

We believe that three is the necessary and sufficient number of classes.

The load in a particular diffserv class is either unregulated or controlled by service-level agreements, which specify the bit rate that the sender has access to for a specific set of destinations. Adding a new service or a new destination to the existing service set requires the negotiation of a new SLA with the network operator. This is a cumbersome process that is acceptable only for semi-permanent communication patterns, e.g., between offices in a corporation, or from home to office for telecommuters. Support for on-demand establishment of sessions is currently lacking. Our DIY proposal may therefore be seen as a means to provide session establishment for diffserv. The guaranteed service we suggest has a sleek form of signaling while the controlled-load service manages the setup without explicit signaling.

In order to bring these two classes into a diffserv context, we would need the following allocation of per-hop behaviors. The guaranteed service requires two code points in the DS-field: one to mark the packets carrying user data and one for packets that carry signaling information. Similarly, the CLS requires one code point for user-data packets and one for probe packets. The state information kept in the network nodes consists of the allocated amounts of capacity, and for GS the amount of reserved capacity within its capacity partition. This means a total of three state variables per output port of which network management sets two and the users set one through the signaling protocol.

## Conclusion

We believe that both intserv and diffserv have merits and seek to define a service architecture that builds on the strengths of both. We have been able to define a measurement-based admission control that runs between sender and receiver without intervention (or even awareness) of the intermediate routers. The end system determines whether it is sensible to start a session. We also have a reservation scheme for which a router only needs to hold one state variable per output link. The value of the variable is the amount of reserved capacity on the link. Senders are allowed to add and subtract to this variable in order to reserve or rescind capacity. All state information associated with ongoing sessions is in both the GS and CLS cases kept in the end systems.

During the course of our work we have come to disregard many of the ideas that are customarily assumed in QoS discussions. First we fully ignore RSVP, which might not be particularly controversial. Second, we believe that the token-bucket traffic descriptor is simply a bad choice and thus we favor a fixed-rate bound. Third, weighted-fair queuing is inappropriate for per-class scheduling since it does not prevent packet clumping. Also, its weighted sharing of surplus capacity between classes is not useful for a guaranteed service since the delay bounds are based solely on the reserved rate. Fourth, diffserv is not a better protection against complex QoS solutions than intserv. Diffserv runs a risk of having overly many service classes defined. This means that nodal equipment must implement all of them at the expense of complexity or a selected subset at the risk of partial incompatibility. Fifth, the mantra in the diffserv community is that it will work if "the network is sufficiently provisioned". Yet there is no basis in the proposals for properly dimensioning a network. In fact, there isn't even clear how an operator will handle the SLAs, how it can decide to grant or decline a request for a service agreement.

Sixth, we do not believe in buffering within network nodes for smoothing of traffic bursts. Buffers are needed only to resolve packet-scale contention due to the asynchrony of the multiplexing. Routers with small buffers have less queuing delays and less need for complex buffer management (like random early discard). The network is thereby approximately a pure loss system and can be dimensioned on that assumption. Seventh, charging should be based on simple parameters on the demand-side, like the peak rate and the session duration (but preferably not the distance: why letting the charging introduce an awareness of distance that the service is supposed to eliminate?). The supply-side parameter influencing the price is the amount of capacity the operator must allocate to a service in order to meet a desirable blocking probability. Eighth, quality of service is not best provided through traffic control solely; it is better provided through a combination of traffic and error controls, supported by applications using robust source coding. So when you want QoS, don't expect it all from the network: DIY!

## References


1. S. Blake, *et al.*, "An Architecture for Differentiated Services," IETF RFC 2475, Dec. 1998.
2. R. Braden, D. Clarke, and S. Shenker, "Integrated Services in the Internet Architecture: an Overview," IETF RFC 1633, June 1994.
3. R. Braden, Ed., "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification," IETF RFC 2205, Sept. 1997.
4. G. Karlsson, "Providing quality for Internet video services,'" in *Proc. CNIT/IEEE 10th Int'l Tyrrhenian Workshop on Dig. Comm.*, Ischia, Italy, Sept. 15-18, 1998.
5. M. Mowbray, G. Karlsson, and T. Köhler, "Capacity reservation for multimedia traffics," Distr. Syst. Eng., Vol. 5, 1998, pp. 12-18.
6. S. Shenker, C. Partridge, and R. Guerin, "Specification of Guaranteed Quality of Service," IETF RFC 2212, Sept. 1997.
7. J. Wroclawski, "Specification of the Controlled-Load Network Element Service," IETF RFC 2211, Sept. 1997.