Mechanisms to provide QoS guarantees

Abstract

We give a brief introduction to scheduling, link sharing, admission control and resource reservation mechanisms, used to provide QoS guarantees to applications in integrated-services networks. We give an overview of some of the latest research results and point out some open problems which we believe are valuable starting points for future research.

1 Scheduling

Scheduling algorithms operate at the smallest timescale (packet transmission time) of the overall traffic management framework. The basic function of a scheduling algorithm is to pick the next packet to transmit from a set of queued packets in accordance with some pre-determined policy. A "good" scheduling algorithm should exhibit the following features:

- isolation - The algorithm should provide QoS guarantees for a session in the presence of traffic fluctuations and (possibly) misbehaving sessions.
- utilization - The algorithm should utilize the bandwidth efficiently.
- fairness - The available bandwidth should be divided among the active sessions in a fair manner.
- low complexity - The algorithm must have a simple implementation. This is critical in high-speed network switches, such as ATM, where the transmission time for a cell is in the order of μs.

There are a large number of scheduling algorithms proposed in literature, each of which offers some combination of the above desired properties. Broadly, scheduling algorithms can be classified as 1) work-conserving—the scheduler is never idle when there is a packet to transmit and 2) non work-conserving—the scheduler transmits only those packets that are considered *eligible* in some sense. Delay Earliest Due Date (Delay-EDD), Weighted Fair Queueing (WFQ) and Virtual Clock (VC) are examples of work-conserving algorithms, while Stop-and-Go, Jitter-EDD and Rate Controlled Static Priority are examples of non work-conserving algorithms.

Since in the IETF specifications of both guaranteed QoS [12] and committed rate QoS [1], the proposed scheduling policy at switches is based on the class of fair-queueing algorithms, in the remainder of this section we concentrate mainly on discussing the algorithms in this class. However the above mentioned service disciplines could be implemented using other scheduling algorithms as well.

1.1 The Fluid Flow Model

We assume that each session is characterized by a *weight* which determines the *relative* share of the bandwidth that the session should receive. In the idealized model (known as fluid-flow system) sessions are assumed to be serviced in arbitrarily *small* increments. In this way, a session is guaranteed to receive its share over any interval of time dt.

Since in practice this is not possible (one of the reasons is that the packet transmission is usually assumed to be non-preemptive), a large number of approximation algorithms have been developed. Demers, Keshav, and Shenker were the first to propose the Packet Fair Queueing (PFQ) discipline, according to which *the packets are transmitted in the order in which they would finish in the corresponding fluid-flow system [5]*. By using the concept of *virtual time*, Parekh and Gallager have analyzed the PFQ[1] algorithm when the input traffic stream conforms to the *leaky-bucket* constraints [10, 11]. In particular, they have shown that no packet is serviced θ_{max} latter than it would have been serviced in the fluid-flow system, where θ_{max} represents the time to transmit a packet of maximum size.

Note: In many papers the session weight (denoted ρ) also represents the *minimum* amount of bandwidth that a session is *guaranteed* to receive. Thus, for a switch with capacity C we have:

$$\sum_{i=1}^{n} \rho_i \leq C \qquad (1)$$

[1] In their work PFQ is known as Packet-by-Packet Generalized Processor Sharing (PGPS for short).

In this model the "excess bandwidth" (i.e., $C - \sum_{i=1}^{n} \rho_i$) is allocated among the competing sessions in proportion their weights. This introduces a coupling between the guaranteed bandwidth and the allocation of the "excess" bandwidth, which may be undesirable.

1.2 Implementation Issues

The complexity in implementing fair-queueing algorithms arises basically from the need to sort packets (at high-speed) according to their virtual finish times and from the need to keep track of the ideal fluid-flow system. It is possible that in a short interval of time as much as $O(n)$ events could occur in the fluid-flow system. Various solutions have been proposed to reduce the implementation complexity of fair queueing systems [7, 3, 8, 13].

1.3 CPU Allocation

In the context of networks with value added services, scheduling algorithms need to be applied to allocate the processing resources inside the network (similar to link bandwidth). However, we note two main differences that makes the problem harder in this case:

- Unlike packet scheduling where the length of the packet and therefore the service time is assumed to be known in advance at the packet arrival, in the processor case it is hard (sometimes impossible) to predict *exactly* how much service time a process will actually use. This makes the computation of the virtual time difficult, which has direct impact on the accuracy of the previous algorithms.
- A general solution should take into account the synchronization (i.e., due to I/O operations). So far, we are not aware of any fair-queueing based algorithm to address this problem.

Recently, numerous algorithms have been developed for processor scheduling [16, 17, 14, 8].

1.4 Delay-Bandwidth Coupling

The main drawback of fair-queueing algorithms is that the introduction of a coupling between the bandwidth allocation and delay. Thus, a session requesting a small delay must allocate a large bandwidth irrespective of its throughput. This makes itdifficult to support low-latency/low-throughput sessions efficiently.

A promising approach to address this problem is to use the concepts of *arrival* and *service* curves [4]. The arrival curve provides an upper bound for the incoming traffic over any interval of time, while the service curve provides a lower bound for the the service time that a session is guaranteed to receive over any interval of time. By using the arrival curve to characterize the session incoming traffic and the service curve to characterize the service time, the delay is bounded by the maximum horizontal distance between the arrival and the service curves, while the session backlog is bounded by the maximum vertical distance between these two curves. By using an EDF algorithm in which the deadlines are assigned as if the the session receives *exactly* its guaranteed service time, it can be shown that both the bounds for delay and session's backlog are guaranteed as long as, for any time interval Δt the sum of the guaranteed service time over all sessions is smaller than Δt (i.e., the CPU is not over-committed). An open question here is how the concept of fairness can be integrated in this model. Another open issue is how the arrival curve can be efficiently computed in practice.

2 Link Sharing

The main purpose of link-sharing is to share the bandwidth on a link between different "entities". An entity can be anything from a single session to a set of sessions that request similar QoS, or the set of sessions belonging to a certain organizations. [6] proposes an hierarchical link-sharing model, where at the higher level the bandwidth is distributed between multiple organizations, and at the lower level(s) the share of each organization is distributed between different QoS classes, such as real-time and best-effort. At each level an entity is guaranteed to receive its share during congestion. Moreover, if one or more entities are passive (i.e., do not have any packet to send) their share is redistributed between the active entities at the same level.

Since link-sharing requires isolation among different entities and flexibility in redistributing the "excess" bandwidth, fair-queueing based algorithms were the natural choice for implementing these mechanisms. In a hierarchical fair queueing system, the delay bounds are strongly influenced by the accuracy with which the fluid system is approximated, which is not the case for a single-level fair queueing system [3]. Finally, we note that it is possible for the lower levels(s) to use other scheduling disciplines. For example in the best-effort class, the sessions may be simply scheduled according to the FCFS or SP disciplines.

Table 1: Comparison of resource reservation setup protocols

	Scalability	Resource sharing	Multicast model	QoS routing	State management	Setup
RSVP	Yes	Yes	Receiver-init	No	soft-state	one-pass
ST-II	Yes	Yes	Receiver/Sender-init	No	hard-state	two-pass
Tenet-2	Yes	Yes	Receiver/Sender-init	Yes	hard-state	two-pass
PNNI/UNI	Yes	No	Receiver/Sender-init	Yes	hard-state	two-pass

3 Admission Control

The main objective of an admission control test is to admit as many sessions as possible, while guaranteeing their QoS requirements. In general, the number of admitted sessions is determined by:

- traffic characterization - As shown in [9], increasing the accuracy of characterizing sessions' incoming traffic, also increases the number of accepted sessions and the link utilization. However, a more accurate characterization requires a larger number of parameters. This dictates a trade-off between the characterization accuracy and the computation complexity of the admission test. Another potential problem is that it is hard to provide an accurate characterization when the traffic profile is not known in advance (e.g., the traffic of a video-conferencing application). Finally, we note that although it is possible to predict the future traffic based on previous data, this will not ensure strict guarantees.
- scheduling algorithm - Usually, there is a trade-off between the algorithm complexity and the number of admitted sessions. While more sophisticated algorithms, such as Delay-EDD, are able to admit a larger number of sessions, other more simple algorithms, such as FCFS and Static Priorities (SP), involve a lower computation complexity.

4 Resource Reservation

In order to provide a contracted, quantitative QoS to a particular flow, the network usually needs to reserve resources such as bandwidth/buffers for that flow. An important component in the integrated services network architecture is the reservation setup protocol that is responsible for setting up and maintaining reservations on each link along the end-to-end path through the network. Some of the desirable properties of a reservation setup protocol are:

- Low connection setup time.
- Scalability (should deal with large multicast groups).
- Support for Sender and Receiver-initiated multicast.
- Resource sharing between senders in the same multicast group.
- Dynamic re-negotiation of QoS.
- Transparent support for re-routing.
- Consistent multipoint-to-multipoint state management (MxN).

RSVP [19] is a reservation setup protocol for the integrated services Internet, which has some of the above properties. It uses receiver initiated reservations to deal with heterogeneous receivers and provides different reservation styles to support shared/distinct reservations for a multicast group. By using "soft state" which is periodically refreshed by senders/receivers in a multicast group, RSVP is able to adapt dynamically to group membership and route changes.

RSVP is not a routing protocol, but relies on existing unicast/multicast routing algorithms. Since the multicast distribution tree is already assumed to exist when the receiver initiates reservation requests, RSVP could potentially try to establish reservations along sub-optimal routes. Another drawback of RSVP is the use of periodically refreshed "soft-state". This does not guarantee consistency of reservation state and could lead to trashing [20]. Moreover the soft-state protocol allows occasional disruptions in service (when the reservations are not refreshed in a timely manner) which makes it difficult to provide guaranteed services.

Other examples of reservation setup protocols are the ATM Forum PNNI/UNI specifications [21, 22], the Tenet-1/2 schemes [23, 24] and the Internet Stream Protocol (ST-II) [25]. Table 1 compares some of the interesting characteristics of these protocols.

In an application aware network framework, it should be noted that reservation protocols (signalling in general) would need to allocate and manage more resources such as compute servers and invoke appropriate value added services, than the traditional network resources such as bandwidth and buffers. The signalling protocol should also be able to deal with multiple connections of different QoS within the context of a single multiparty call. New interfaces are needed between applications and the signalling protocols and between signalling protocols and service providers to allocate and manage services and resources in the network.

References

[1] F. Baker, R. Guerin, and D. Kandlur, "Specification of Committed Rate Quality of Service,", *IETF Draft*, August 1996.

[2] J. C. R. Bennett and H. Zhang, "WF^2Q : Worst-case Fair Queueing", *Proc. of IEEE INFOCOM'96*, San-Francisco, March 1996.

[3] J. C. R. Bennett and H. Zhang, "Hierarchical Packet Fair Queueing Algorithms", *Proc. of ACM SIGCOMM'96*, Stanford University, August 1996, pp. 143–156.

[4] R. L. Cruz, "Quality of service guarantees in virtual circuit switched networks," *IEEE Journal of Selected Areas in Communication*, vol. 13, no. 6, Aug. 1995.

[5] A. Demers, S. Keshav, and S. Shenkar, "Analysis and Simulation of a Fair Queueing Algorithm", *Journal of Internetworking Research & Experience*, October 1990, pp. 3–12.

[6] S. Floyd, and V. Jacobson, "Link-Sharing and Resource Management Models for Packet Networks", *IEEE ACM Transactions on Networking*, vol. 3, no. 4, August 1995, pp. 365–386.

[7] S. J. Golestani, "A Self-Clocked Fair Queueing Scheme for Broadband Applications", *Proc. of IEEE INFOCOM'94*, April 1994, pp. 636–646.

[8] P. Goyal, X. Guo and H. M. Vin, "A Hierarchical CPU Scheduler for Multimedia Operating Systems", to appear in *Proc. of the 2nd OSDI Symposium*, October 1996.

[9] E. W. Knightly, D. E. Wrege, J. Liebeherr, and H. Zhang, "Fundamental Limits and Tradeoffs of Providing Deterministic Guarantees to VBR Video Traffic", *Proc. of ACM SIGMETRICS'95*, Ottawa, Canada, May, 1995.

[10] A. K. Parekh and R. G. Gallager, "A Generalized Processor Sharing Approach To Flow Control in Integrated Services Networks-The Single Node Case", *ACM/IEEE Trans. on Networking*, Vol. 1, No. 3, 1992, pp. 344–357.

[11] A. K. Parekh, "A Generalized Processor Sharing Approach To Flow Control in Integrated Services Networks", *Ph.D Thesis*, Department of EE and CS, MIT, 1992.

[12] S. Shenker, C. Partridge, and R. Guerin, "Specification of Guaranteed Quality of Service,", *IETF Draft*, August 1996.

[13] D. Stiliadis and A. Varma, "Frame-based Fair Queueing: A New Traffic Scheduling Algorithm for Packet-Switched Networks," Technical Report UCSC-CRL-95-39, July 1995.

[14] I. Stoica, H. Abdel-Wahab, and K. Jeffay, "A Proportional Share Resource Allocation for Real-Time, Time-Shared Systems", to appear in *Proc of RTSS'96*, December 1996.

[15] S. Suri, G. Varhese, and G. Chandranmenon, "Leap Forward Virtual Clock", accepted to *INFOCOMM'97*.

[16] C. A. Waldspurger and W. E. Weihl. "Lottery Scheduling: Flexible Proportional-Share Resource Management," *Proc. of the 1st OSDI Symposium*, November 1994, pp. 1–12.

[17] C. A. Waldspurger. "Lottery and Stride Scheduling: Flexible Proportional-Share Resource Management," *PhD Thesis*, Technical Report, MIT/LCS/TR-667, Laboratory for CS, MIT, Sept. 1995.

[18] L. Zhang, "VirtualClock: A New Traffic Control Algorithm for Packet-Switched Networks", *ACM Trans. on Comp. Systems*, vol. 9, no. 2, May 1991, pp. 101–124.

[19] R. Braden, L. Zhang, S. Berson, S. Herzog and S. Jamin. "Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification," *Internet Draft, draft-ietf-rsvp-spec-13.ps*, August 1996.

[20] Danny J. Mitzel, Deborah Estrin, Scott Shenker and Lixia Zhang. "A Study of Reservation Dynamics in Integrated Services Packet Networks," *Proc. of INFOCOM '96*, March 1996, San Fransisco, USA.

[21] The ATM Forum. *Private Network-Network Interface Specification Version 1.0*, March 1996.

[22] The ATM Forum. *ATM User-Network Interface Specification Version 3.1*, September 1994.

[23] A. Banerjea, D. Ferrari, B. A. Mah, M. Moran, D. C. Verma and H. Zhang. "The Tenet Real-Time Protocol Suite: Design, Implementation, and Experiences," *IEEE/ACM Transactions on Networking*, vol. 4, no. 1, pp. 1-11, February 1996.

[24] R. Bettati, D. Ferrari, A. Gupta, W. Heffner, W. Howe, M. Moran, Q. Nguyen and R. Yavatkar. "Connection Establishment for Multi-Party Real-Time Communication," *Proc. of NOSSDAV '95* New Hampshire, April 1995, pp. 255-266.

[25] L. Delgrossi and L. Berger. "Internet Stream Protocol Version 2 (ST2) Protocol Specification — Version ST2+," *Request for Comments 1819*, August 1995.