Capacity reservation for multimedia traffic

Miranda Mowbray, Gunnar Karlsson and Torsten Koehler

Addresses:
Miranda Mowbray, Hewlett Packard Laboratories Bristol, Bristol BS12 6QT, UK.
Gunnar Karlsson and Torsten Koehler, Swedish Institute of Computer Science, Isafjordsgatan 22, S-164 28 Kista, Sweden.

Abstract

We introduce a capacity reservation scheme for multimedia traffic. We compare it to other algorithms in the literature. It has been implemented and its worst-case performance has been analysed. It appears to give a noticeably improved quality of service to delay-sensitive traffic.

1 Introduction

Networked multimedia applications are becoming increasingly important to the business community, and an increasing proportion of the traffic of data networks is used by such applications. Multimedia traffic has different quality of service needs than those provided by most routers, which were designed with regular data traffic in mind. The main feature of multimedia traffic which makes it different from regular data traffic is that it is much more sensitive to delay. Current routing technology gives "best-effort" service, aimed at maximizing throughput, with minor consideration to delay and loss. The result of this is that some delay-sensitive packets are not delivered in time. We therefore believe that it is worth using a router scheduling algorithm specially designed to give good performance for multimedia traffic.

In this paper we introduce a new scheduling algorithm for routers which is designed to solve this problem by minimizing the worst-case delay to delay-sensitive traffic without dropping these packets in the routers, whilst providing a best-effort type service to other traffic. The delay-sensitive traffic need not necessarily come entirely from multimedia applications – it might include interactive data traffic, which is also sensitive to delay. The algorithm works with a reservation scheme; roughly speaking, a proportion of the capacity is reserved for the use of delay-sensitive traffic. Provided that delay-sensitive traffic arriving at the router does not exceed its reservation, a bound on the

delay can be given so that no delay-sensitive packet ever has an end-to-end delay exceeding the bound. The reserved capacity carries packets loss-free through the network with regard to contention (bit-errors might still occur). The method is based on a pacing mechanism, which controls the departure instances of the reserved packet stream. The leftover non-reserved capacity can be used by anyone at best effort.

Much research on traffic control has focused on the provision of delay and loss guarantees in conjunction with statistical multiplexing (see, for instance, [9, 5]). It may therefore seem a step backwards to propose a capacity reservation which enforces a deterministic service rate for a source. In effect we have a circuit (a TDM channel in telephony networks), albeit without the constant delay since it still is asynchronous. First, it should be kept in mind that the reservations do not preclude statistical multiplexing, which is used for best effort traffic. Second, statistical multiplexing is not an end in itself. Its prime motivation has been to efficiently use the network capacity, a resource which appears less and less scarce. Third, statistical multiplexing with guarantees has not yet been proven viable for larger scale networks in daily operating. Consider instead some of the advantages with deterministic capacity reservation over statistical multiplexing with quality guarantees:

- Implementation with low complexity
- Simple source traffic descriptor (an upper limit on the bit rate)

- Straightforward call-acceptance procedure
- Loss-free operation with bounded delay
- Sound basis for charging

We have strived for simplicity and robustness (as recommended by eg. [11]) and have also attempted to give the non-reserved traffic a smooth and truly best-effort service, without delays due to reserved traffic.

We see the algorithm being used for routing within enterprise networks. These are networks owned by large or medium-sized enterprises, typically spanning several separate sites but with at most 4 or 5 hops between any source and destination. Many companies started out with all their IT infrastructure in the IT department at the company's central headquarters, and then acquired more equipment at branch offices. There has been a tendency throughout the US and Europe for the different branches to become networked together, decentralizing the use of IT within the company, and creating an enterprise network. These networks are large enough to support heavy use of multimedia and interactive traffic, but have few enough hops so that it is possible to give useful bounds on the worst-case end-to-end delay.

Comparisons between this and other available schemes are made elsewhere in this paper.

2 The capacity reservation scheme

Fig. 1 Separation of traffic classes into different FIFO queues

Connections which require reserved capacity are treated separately from traffic without reserved capacity (so called "best effort" traffic). The reserved capacity could be a part or the full (maximum) capacity needed by the call. Each of these two traffic classes has its own buffers in the network nodes, as shown in Fig. 1, which are denoted $q1$ and $q2$. The classes are separated by addresses or some other header information. The packets in the reserved class could be given either high or low loss-priority. The surplus traffic with low loss-priority is allowed on a connection to increase the flexibility – the reserved rate can be exceeded at its own risk – and to better the utilization (eg, when capacity is reserved but little used and there is no "best effort" traffic).

The reservation only covers the high-priority packets of a stream with reservation. The low-priority packets are forwarded if there is reserved capacity

left over. Low-priority packets can thus only use unused portions of the reserved capacity. Best-effort on the contrary may use all left over capacity (unused reserved as well as all unreserved). The best-effort traffic may have a different set of loss-priorities, it is not of consequence for the reservation scheme.

It is up to the user to decide how much capacity to reserve and how much to vie for when a connection is requested. For video under layered source coding, for example, packets can have different loss tolerance. You may not want to reorder these packets, however, so they are not assigned to different traffic classes. The scheme supports this scenario well: you make a reservation which is sufficient for what must absolutely be delivered and send the rest at a low-priority on the same connection. The low-priority packets which get across are delivered in the proper order. If order is not important, send the important part as reserved and the surplus as best effort. Again, the option to send the low-priority packets along with the reserved packet-stream is to soften the fixed reserved rate since it may be hard to estimate correctly the rate before the session has started. So, if it is a bit low the surplus may still be sent, albeit at its own risk, rather than being dropped at the sender.

We define the reserved rate of a connection as the number of bytes, R, that may be sent during a window of W bytes of transmission capacity. (Since packets are measured in bytes, W is expressed in bytes as well.) The re-

served rate can be any ratio R/W of the link's capacity (R and W can be arbitrarily large). The connections without reservations get full use of the remaining portion $(W - R)/W$. Packets are serviced from the best-effort queue exclusively if the other is empty. Reserved capacity is consequently not wasted.

Fig. 2 A multiplexer with N input links.

If we assume N incoming links to a multiplexer and that all incoming links have the same capacity, then

$$C_{in} \sum_{j=1}^{N} \frac{R_j}{W_j} = C_{out} \frac{R_{out}}{W_{out}} \quad (1)$$

C_{in} and C_{out} are the capacities of the incoming and the outgoing links, respectively. The outgoing reserved capacity equals the sum of the incoming, reserved capacity. The equation shows that $W_{out} \leq$ Least common multiple of (W_j). At each stage of multiplexing the values of W_{out} and R_{out} for the outgoing stream may consequently increase. The departures of the packets may well be clustered within the window. The buffer space needed for the reserved traffic to avoid packet loss will thereby increase. Also, if the connections with reservations are served for R bytes before the non-reserved traffic

is served, then the latter may be unduly delayed and receives a bursty service. This worsens its traffic characteristics and thereby also its performance, in terms of loss and delay.

The problem with clustering is that a constant R/W ratio can be $(aR)/(aW)$ and that the factor a may grow indefinitely. Thus, you can send aR bytes consequtively with in an aW window which is more clustered than R bytes in a row over a W-byte long window.

There are two immediate solutions to avoid aggregation of clusters. The first is to require $W_j = W_{out}$ for all j. If the links all have the same capacity, and the rate is specified for a fixed value of W, then W cannot become larger and clustering is not a problem. This is the approach of the TTT [10] and the Stop-and-Go queuing [4].

However, keeping W fixed is a rigid solution. If the value of W is low, then the resolution of the specified rates is limited (to W^{-1} bytes per second). When the value is high, best effort traffic may still be considerably delayed. We have instead chosen to alleviate the batch arrival problem by enforcing an even distribution of the packet departures within a window.

The issue is consequently how to pace the packet departures evenly to maintain the R/W ratio.

2.1 The Algorithm

The basic idea was to serve a packet from $q1$ and then to wait for an idle period I before serving the queue again. Best effort traffic is served during the idle period. If $L\{P_i\}$ is the length of a packet served from $q1$, then we get

$$I_i = L\{P_i\}(\rho^{-1} - 1) \quad (2)$$

Here $\rho = R/W$. That means that the reserved rate is maintained. Typically, one reserved packet is served, then a few best effort packets, and then another reserved packet. The problem with this calculation is that the length of the idle period gets short when the reserved rate is high – to be precise, the length of the idle period tends to zero as ρ tends to one. Even though there will be spare capacity available to best effort traffic, it may be fragmented into periods too short for a packet. We therefore allow several packets to depart back-to-back until $L\{P_1\} + L\{P_2\} + \ldots + L\{P_i\} \geq L\{P_1'\}\rho/(1-\rho)$ (where P_1 is the first packet of $q1$, and P_1' is the first packet of $q2$). The number of delay-sensitive packets served back-to-back is chosen so that the corresponding idle period is at least as long as the time to send the pending delay-insensitive packet.

The scheme does not tell which should be the next packet within its class to send. The simplest is to pick them in FIFO order.

The steps in the pacing are as follows (assumes packets in both $q1$ and $q2$).

I. Serve $q1$ for $i > 0$ packets, until

$$L\{P_1\} + L\{P_2\} + \ldots + L\{P_i\} \geq L\{P_1'\}\rho/(1-\rho) \quad (3)$$

II. Serve $q2$ for j packets, $j > 0$, until

$$L\{P_1'\} + \ldots + L\{P_j'\} + L\{P_{j+1}'\} > (L\{P_1\} + \ldots + L\{P_i\}).(\rho^{-1} - 1), \quad (4)$$

or until $q2$ is empty (whichever is shorter).

III. If

$$(L\{P_1\} + L\{P_2\} + \ldots + L\{P_i\}).(\rho^{-1} - 1) - (L\{P_1'\} + L\{P_2'\} + \ldots + L\{P_j'\})$$

is greater than zero, then schedule an idle (during which no traffic at all is sent) for this many bytes.

The pacing gives a known arrival process to the reserved traffic on each link so that it is possible to determine the buffer space needed to ensure a lossless operation.

2.2 Call acceptance

The goals of the proposed call-acceptance technique are to simplify the call acceptance, and to provide users with a loss-free information transfer. Packet loss is avoided by sufficient buffering in the nodes.

The acceptance procedure for calls that request some amount of reserved capacity can be sketched as follows. The network controller preallocates some portion of the capacity on the network links. This ratio could be chosen in accordance with the available buffer sizes. Notice that this capacity is not wasted when not used by connections with reservations since surplus capacity may be used by best-effort connections.

To accept a call, the network control has to find a route to the destination such that on each link

$$\rho_{req} + \rho_{ex} \leq \rho_{res} \quad (5)$$

(The ratio denoted 'res' is the reserved, preallocated portion of the link's capacity; 'ex' is the existing, used part of that capacity and 'req' is the requirement of the new call.) If the inequality holds for all links on the route, then the call is accepted and the used capacity is updated:

$$\rho_{ex} \leftarrow \rho_{ex} + \rho_{req} \quad (6)$$

The call could be blocked if no such route exists. The preallocated capacity could alternatively be updated on needed paths, according to

$$\rho_{ex} \leftarrow \rho_{ex} + \rho_{\Delta} \quad (7)$$

for some ratio ρ_{Δ}, so that Eq. (5) now can be met.

This increment of the preallocation can be done in anticipation of new connections to reduce the probability of blocking. The preallocation lowers the

need for updates triggered by the call acceptance procedure, since most new calls may be handled according to Eq. (5). Note that the preallocation does not lead to a full reservation of all the capacity over time since reservations on the incoming links to a multiplexer may not overload the output (see Eq. (1)). But if the used part of the existing reservation is low, the reservation can be reduced. The main idea is that the capacity allocation may be disassociated from the handling of connection requests.

2.3 Remarks

All connections with reserved capacity are paced at the network access. The amount of capacity to reserve for a connection could be chosed according to the "effective capacity" of the source for a given tolerable buffering delay (see [6] for details on "effective capacity").

The pacing function is performed per link (output port) and not per stream, as in schemes suggested for statistical multiplexing. The complexity of the implementation is thus comparatively lower for the capacity reservation. The dual buffers for the two traffic classes may be logical queues in shared memory.

The two buffers may be dimensioned differently. The buffer for traffic with reservations must have a size in accordance with the limits given for loss-free operation. Any additional space should be included only if the delay limits

allow it to hold more packets with low loss-priority. Best effort traffic could instead be given ample buffer space to minimize the probability of packet loss.

The reservation scheme allows "best effort" traffic to use all non-reserved capacity as well as the slack in the reserved capacity. The network may therefore be well utilized. It may also be operated to emulate a TDM network by requiring full reservation for all connections. The reservations can use all of the network's capacity, when needed. The reservation scheme has advantages also over a traditional TDM network: it does not require a fixed channel (framing) structure nor synchronicity within the nodes.

The charge for a call consisting of packets in a reserved traffic class may be based on the amount of reserved capacity, its duration and, possibly, the length of the route. Packets within a reserved traffic class with low priority that are sent in addition over the connection would not be counted. This basis is more attractive than the charge after behavior (such as peak-to-mean ratio) that may result from the statistical traffic control.

The capacity reservation is most suited to near isochronal sources which have predictable rates. The performance guarantees are also most valuable for such sources, which include most video and audio sources, since the needed real-time delivery makes retransmission infeasible. Batch data, on the contrary, is commonly offered a best effort service and the reliability is added by

retransmission at the transport layer. Note, however, that the simplified call acceptance procedure may allow sufficiently fast connection establishment to handle transfer of bulk data, such as images.

3 Worst-case performance analysis

The performance of this algorithm with respect to a particular traffic stream was analysed using techniques similar to those in [1, 2]. The traffic stream was that of delay-sensitive traffic in a route from a 100BaseVG LAN, to an FDDI ring, across a 64×6kb/s WAN to another FDDI ring, and from there to a 100BaseVG LAN, using four routers in all which were assumed to run the same scheduling algorithm. The delay-sensitive traffic was as bursty as possible, subject to realistic bounds on packet lengths, and there was enough delay-insensitive traffic to flood the WAN. Estimates of worst-case delays on the media and non-queueing delays within the routers were made based on real-life measurements. For this stream, when the routers run the algorithm described, the worst case end-to-end delay for a delay-sensitive packet was calculated to be around 162ms. All non-preemptive scheduling algorithms give a worst case end-to-end delay of over 150ms for this example. Golestani's Stop-and-Go [4] gives a worst case delay of over 165ms, and pure FIFO scheduling, which does not distinguish between delay-sensitive and delay-insensitive traffic (as in many current routers), gives a worst case delay

of over 500ms.

The ITU recommendation for delays for telephony (G.114) is that 150ms end-to-end delay is certainly acceptable, care is to be used above that, and 400ms delay is unacceptable.

An upper bound can be calculated on the amount of buffer space needed to ensure that no delay-sensitive packets are dropped: for this particular traffic stream, under the algorithm described, buffer space for 36 delay-sensitive packets will suffice. The net buffers in the router measured hold 32 packets. However, the bound is for the space needed for *delay-sensitive* packets alone. Some buffer space should also be provided for delay-insensitive packets. In contrast to pure FIFO scheduling, it is not necessary to have any constraints on the quantity of delay-insensitive traffic entering the system in order to ensure that delay-sensitive packets are not dropped.

4 Implementation

A prototype implementation of the algorithm was done for a Hewlett Packard workstation, connected over a Serial Line IP (SLIP) link to another workstation.

Here is the pseudocode on which the scheduling part of the implementation was based. This part was implemented within the SLIP server, which acts

as endpoint and router, so that hosts which do not have SLIP access can communicate with the SLIP router over a LAN and gain from the scheduling inside the router. The SLIP software was extended to integrate the scheduler and to deal with the two pseudo network interfaces simultaneously.

`LEN_LO` and `LEN_HI` return the length of the next waiting delay-insensitive and delay-sensitive packet respectiviely, `SCHED_LO/HI` schedules the specified packet for transmission and `SCHED_ID()` schedules an idle period.

```
while (1) {
    /* Initialize delay-insensitive length counter */
    lo_s = 0;
    /* Calculate length of delay-insensitive service */
    hi_s = hi_s * (1 - rho)/rho;
    do {
        /* If there is a delay-insensitive packet, schedule it
           and add its length to the counter */
        lo_s += lo;
        if (lo) SCHED_LO(lo);
    }
    /* If there is a no delay-insensitive packet, and service
       is to delay-insensitive traffic, schedule an idle */
    while((lo = LEN_LO()) && (lo_s + lo <= hi_s));
```

```
    if (hi_s - lo_s > 0) SCHED_ID(hi_s - lo_s);
    /* Initialize delay-sensitive length counter */
    hi_s = 0;

    /* Calculate length of delay-sensitive service */
    lo_s = lo * rho/(1-rho);
    while(hi = LEN_HI()) {
         /* If there is a delay-sensitive packet, schedule it
            and add its length to the counter */
         hi_s += hi;
         SCHED_HI(hi);
         if (hi_s >= lo_s) break;
    }
}
```

5 Review of proposals in the literature

There are three proposals for scheduling with deterministic performance guarantees in the literature that have received much attention: Virtual Clocks by Zhang [13], Packet-by-Packet Generalized Processor Sharing by Parekh and Gallager [7], and Stop-and-Go Queuing by Golestani [4]. The first two propose work-conserving, non-FIFO algorithms. (A *work-conserving* algo-

rithm is one for which the link is never left idle if there is a packet in the queue to be sent.) The latter as well as the algorithm proposed here are not work-conserving and could use any service order.

This section presents a brief review of the proposals that were rejected in favor of our own algorithm. (See [12] for a more extensive overview of scheduling algorithms.)

5.1 Virtual Clocks

Each connection (flow, stream, virtual circuit, call or any other denotation) has its own virtual clock. The clock is incremented each time a packet belonging to the connection arrives to the (multiplexing) buffer. The packet is stamped with the new time of the clock. Packets are then served from the buffer in order of increasing time stamps. The service order is thus FIFO per connection but not for the aggregation of connections.

The clock increment is the inverse of the connections reserved rate times the length of the packet (for example, at 3 Mb/s, a 375-byte packet would advance the clock by 1.0 ms). The rate of a connection exceeds the reservation when its virtual clock surpasses the network clock.

Zhang's paper explains a framework for traffic control. It could be used to provide performance guarantees by providing policing and guaranteed buffer

space. No such results are presented in the paper, but an upper delay bound has been derived recently, and it is identical to that of the PGPS algorithm [3].

The method requires one virtual clock and a logical queue per connection. This may cause scaling problems when implemented. The clock increment must be computed per packet.

5.2 Packet-by-Packet Generalized Processor Sharing (PGPS)

This scheme [7, 8] is akin to the previous one. Each arriving packet is stamped with the time it would complete service according to a Generalized Processor Sharing (GPS) scheme (cf. scheduling according to the time stamps of the virtual clocks). The packet which would complete service first is scheduled to go first.

GPS assumes traffic and service rates to be infinitely divisible. Each connection has a weight W_i which guarantees it a portion equal to $W_i/(\Sigma_j W_j)$ of the total capacity, where the denominator is the total of all weights. When a connection does not use its allocation, the slack may be shared by the backlogged connections according to their weights. A "best-effort" traffic class is assigned a weight and treated like any other connection.

The packet-based GPS closely approximates the GPS and performance guar-

antees are possible for connections controlled by leaky buckets. The paper gives the bounds for single-node case (the multinode case is covered in a sequel [8]). The bounds get weaker as the number of hops increases.

The implementation has the same drawback as the former scheme: the complexity scales with the number of connections. Another complication is that all weights must be recomputed when a new connection is accepted. The computation per packet of the completion time, used for the scheduling decision, is more complex than determining the clock increment in the previous method.

5.3 Stop-and-Go Queuing

The requirements on input connections in this method have a strong resemblance to those for the TTT specification method. A connection is restricted to send at most B bytes of data in a period of length T (a frame in the author's terminology). A packet arriving in frame i at a node will not be serviced until the start of the next frame, $i + 1$. The service is consequently not work-conserving. The delayed service ensures that a smooth connection does not get clustered. Any service discipline could be used to schedule which packets should go in a frame, including the ones discussed above.

The frame length T determines the worst-case delay through the network. A small T gives a low delay, but limits the granularity of the capacity al-

locations, which is one packet per frame. It may be alleviated by allowing multiple frame sizes at the cost of increased complexity. The minimum and maximum delays are no more than $N \times T$ seconds apart, for a route N hops long. (Here multiplexing hubs are counted in the hop count, as well as routers.) The jitter is consequently tightly bound. When the worst case delay is near the unacceptable, one cannot, however, hope to have acceptable performance on the average.

Best effort traffic may be serviced during the periods when controlled packets wait for the next frame to commence. The traffic will receive a service which clusters the packets at the end of each frame. This will worsen its loss performance.

The algorithm is simple to implement if the packets are served in FIFO order and a single frame length is used. The server either forwards the first packet of the queue, or waits until the next frame starts (best-effort traffic would be served during the waiting period).

5.4 Discussion

Both Virtual Clocks and PGPS concentrate on the service of a queue and attempt to divide the service evenly in time over the connections, in proportion to their allocations. The idea is to emulate a system where each connection has its own queue that is serviced at the allocated rate. GPS of-

fers in principle the most even division possible. PGPS closely approximates the unattainable GPS and is therefore closest to the goal.

Queuing in the node occurs when the incoming rate of a connection temporarily exceeds the service rate of the connection. For a route over more than one hop, the issue is whether the service discipline may propagate and even aggravate the burstiness of a connection. If so, the delay (and loss) may increase in each node along the route.

Queuing in the node occurs when the incoming rate of a connection temporarily exceeds the service rate of the connection. The worst-case bounds for Virtual Clocks and PGPS are derived under the assumption that all connections are shaped by leaky buckets at the network access. This makes the service for the end-to-end connection non-work conserving. The virtue of having conservation of work inside the network when the access is non-work conserving is therefore questionable. The algorithm we propose is not work conserving for reserved traffic but is approximately work conserving for best effort traffic (the idle period in step III of the algorithm constitute some lost service time since it is too short for sending the j + 1st best-effort packet).

The main difference between Stop-and-Go and the algorithm proposed in this paper is that the latter does not have a fixed frame size. In fact, one may view the window size W (in which R bytes are sent) as the frame size in our scheme. It is thus different for each link, depending on its total capacity and

proportion of reserved capacity. Another difference is that Stop-and-Go only restricts a connection to B bytes of data per frame; we require in addition that the data is as evenly spread over the window as possible considering the entirety of the packets. This restriction is added to give non-reserved connections a smooth and truly best-effort service.

6 Conclusion

Multimedia traffic has different quality requirements from those of standard data traffic – in particular, interactive voice and video traffic is more sensitive to delay than standard data traffic is.

In this paper we have introduced a new scheduling algorithm which is designed to give a good quality of service to delay-sensitive traffic. We have given an example of how it might be implemented, and compared it to other algorithms in the literature. The algorithm has been simulated and its worst-case performance has been analysed.

The new algorithm has the advantage that the quality of service to delay-insensitive traffic does not deteriorate if the amount of delay-sensitive traffic sent onto the network is more than the reserved capacity. Therefore with this algorithm it is not necessary to monitor the delay-sensitive streams to ensure that they conform to their specifications, in order to preserve the

quality of service of delay-insensitive traffic. (Such monitoring is necessary, however, if one wants to ensure the loss performance, and may also be useful in identifying misbehaving streams.) The algorithm attempts to give an even service to delay-insensitive traffic.

We conclude that the use of the new algorithm appears to give a noticeable improvement to the quality of service for delay-sensitive traffic.

References

[1] Rene L. Cruz, *A Calculus for Network Delay, Part I: Network Elements in Isolation*, IEEE Trans. Information Theory vol.37 no.1, Jan '91, pp. 114-131.

[2] Rene L. Cruz, *A Calculus for Network Delay, Part II: Network Analysis*, IEEE Trans. Information Theory vol.37 no.1, Jan. 1991, pp. 132-141.

[3] N. Figueira and J. Pasquale, *An upper bound on delay for the Virtual Clock service discipline*, IEEE/ACM Transactions on Networking, Vol. 3, No. 4, August 95, pp. 399-408.

[4] S. J. Golestani, *A Stop and Go Queueing Framework for Congestion Management*, Proc. ACM SIGCOMM'90, Philadelphia PA, September 1990, pp.8-18

[5] S. Jamin, et al., *A Measurement-based Admission Control Algorithm for Integrated Services Packet Network*, ACM Computer Communications Review, Vol. 25, No.4, October 1995, pp. 2-13.

[6] F. P. Kelly, *Notes on effective bandwidth*, in Stochastic Networks: Theory and Applications (Eds. F. Kelly, S. Zachary, I. Ziedins), Oxford University Press, 1996.

[7] Abhay K. Parekh, Robert G. Gallager, *A Generalized Processor Sharing approach to flow control in integrated services networks – the single node*

case, IEEE/ACM Transactions on Networking, Vol.1, No.3, June 1993, pp.344-357

[8] Abhay K. Parekh, Robert G. Gallager, *A Generalized Processor Sharing approach to flow control in integrated services networks – the multiple node case*, IEEE/ACM Transactions on Networking, Vol.2, No.2, April 1994, pp.137-150.

[9] H. G. Perros and K. M. Elsayed, *Call Admission Control Schemes: A Review*, IEEE Communications Magazine, Vol. 34 no. 11, November 1996, pp. 82-91.

[10] Michael Spratt, *The Target Transmission Time per Source Port (TTTPSP) Scheduling Algorithm and Derivatives*, HP Labs internal document

[11] G. Woodruff, R. Kositpaiboon, *Multimedia Traffic Management Principles for Guaranteed ATM Network Performance*, IEEE Journal on Selected Areas in Communications, Vol.8 no.3, April 1990, pp.437-446.

[12] H. Zhang, *Service Disciplines for Guaranteed Performance Service in Packet-Switching Networks*, Proceedings of the IEEE, Vol. 83, No. 10, October 1995, pp. 1374-1396.

[13] L. Zhang, *Virtual Clocks: A New Traffic Control Algorithm for Packet Switching Networks*, Proc. ACM SIGCOMM'90, Philadelphia PA, Sept. 1990, pp. 19-29.