A Simplified Guaranteed Service for the Internet

Evgueni Ossipov and Gunnar Karlsson

Department of Microelectronics and Information Technology,
KTH, Royal Institute of Technology, Sweden,
{eosipov,gk} @imit.kth.se

Abstract. An expected growth of real-time traffic in the Internet will place stricter requirements on network performance. We are developing a new simplified service architecture that combines the strengths of the integrated and differentiated services architectures. In this paper we focus on the issues related to providing a guaranteed service in a high-speed network. We give a description of the service, which includes a lightweight signaling protocol and a non-work-conserving scheduling algorithm, and describe the system requirements and the performance evaluation. Our implementation of the protocol allows processing of 30 million signaling messages per second.

1 Introduction

Much effort today in the Internet research community is aimed at providing services for applications that were not under consideration when the Internet was originally designed. Nowadays the network has to support real-time communication services that allow clients to transport information with expectations on network performance in terms of loss rate, maximum end-to-end delay, and maximum delay jitter. A number of solutions to accomplish these needs have been proposed and implemented. The two most well-known approaches are the integrated services (*intserv*) [1] and differentiated services (*diffserv*) [2], [6] architectures. The intserv architecture classifies network traffic into three classes: *guaranteed service*, *controlled load service* and *best effort*. In diffserv, the traffic is assigned to specific behavior aggregates.

For traffic with guarantees, intserv provides reservation of bandwidth and buffers by using signaling between network nodes. An example of such signaling is RSVP [3]. Intserv keeps per-flow soft state in each network node. In comparison, diffserv avoids per-flow states in the routers, and instead ingress nodes perform traffic metering and admission control on the flows. The services offered to customers are statically described by service level agreements.

From the point of view of performance and scalability, intserv appeared to be a too cumbersome architecture for high-speed IP networks. Basically, the performance is limited by a router's ability to process and maintain the set of per-connection states. The diffserv architecture, however, is free of these limitations since a router serves all flows of a behavior aggregate together. The high scalability of diffserv is accomplished by separating the operations performed at the borders of the network

G. Carle and M. Zitterbart (Eds.): PfHSN 2002, LNCS 2334, pp. 147–163, 2002.

from those performed in the core. There is alas an inflexibility in this approach, namely a service level agreement (SLA) does not allow the end user to dynamically change the service requirements, for example maximum allowed bit rate.

The limited scalability of the intserv approach and lack of dynamic resource allocation in the diffserv architecture have motivated researchers to work on the combination of the two QoS approaches, rather than to consider them separately. That work is basically going on along two tracks. Some groups try to make the two approaches interoperable, for example by classifying traffic into the intserv traffic classes at the network ingress and then mapping aggregated flows of the same class to certain behavior aggregates of the diffserv model [4], [5], [7]. Other groups are working on simplifications of the signaling protocol [8-13]. The recently organized IETF working group on Next Steps in Signaling (NSIS) [30] works on a specification of the next generation signaling protocol that will overcome problems of RSVP and diffserv's static SLAs; however, there is no solution proposed yet.

The work presented in this paper develops the ideas of a QoS architecture proposed by Karlsson and Orava in [14]. It introduces a new simplified service architecture that combines the strengths of the intserv and diffserv models. Our contribution is as follows. We provide a description of the guaranteed service model, which includes a lightweight signaling protocol. We suggest a simple way of flow description based on a fixed rate. We further evaluate the non-work-conserving scheduling introduced in [15], and provide formal and experimental justification of its suitability for our service architecture.

The remainder of the paper is organized as follows. Section 2 describes the service architecture and its assumptions. In Section 3 the description of the simplified signaling is presented. Section 4 describes the scheduling algorithm with its system requirements, performance parameters and simulation results. We state open issues and summarize our work in Section 5.

2 Description of the Service Architecture

We are developing a service architecture consisting of three quality classes, as illustrated in Fig. 1. The *guaranteed service* (GS) class provides deterministic guarantees, namely constant throughput, absence of packet loss due to queue overflows in routers, and tightly bounded delay. Flows of the *controlled load service* (CLS) class have bounded packet loss and limited delay. The third class of service is the customary *best effort* (BE). The flows of this class have no guarantees, they obtain leftover capacity in the routers, and may be lost due to congestion. Both GS and CLS are allocated restricted shares of the link capacity so that BE traffic always has a possibility to get through. Best effort traffic is also using all unused portions of capacity reserved for CLS and GS traffic. Since GS and CLS are isolated from each other, we consider only the interaction between GS and BE in this paper. The work in our group that is related to CLS is described in [16], [17].

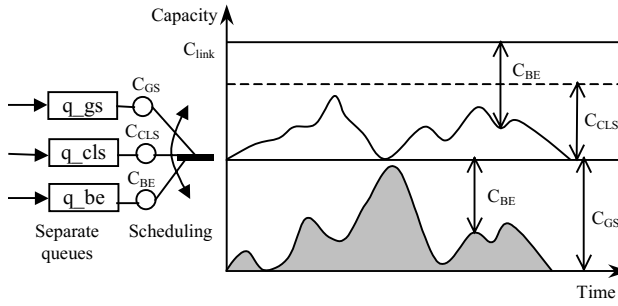


Fig. 1. The service architecture.

2.1 Assumptions Used in This Paper

For the purpose of our presentation we define the following actors in the network:

End systems (ES). These are client systems, which use certain services in the network, and the servers, which offer the services to the clients. We refer to data of a session between two end systems as a *GS flow*. Sufficient capacity for a particular GS flow is provided by explicit reservation in all nodes along a path. The ES's use a simple signaling protocol to make reservations. It is described below.

Access gateways (or ingress nodes) (AG). These are routers situated on the border between the backbone and the access networks of an ISP. An AG is the first IP-level node of a GS flow. In our architecture AG's keep per-connection soft state, perform policing and billing.

Core nodes (or routers). These are all routers apart from AG's. They provide IP-level packet forwarding, and do not keep per-connection soft state. Packets belonging to the GS class are queued in a separate GS queue at each node in the network, as shown in Fig. 1. The total traffic of all GS *flows* that share a particular link is referred to as an *aggregated GS flow*.

We base this work on the following assumptions.

- All nodes in the network are output-buffered devices that implement the scheduling proposed later in the paper (we are working on the issues related to the scheduling inside the switching fabric).
- Each GS connection is described solely by a peak rate as traffic specification. The signaling protocol assumes the reservation state to be purely additive: The reserved rate of an outgoing link of a router equals to the sum of the incoming rates for the link. This is enforced by the scheduling.
- We also assume that re-routing does not occur in the network. Although we are aware of the problem that routes in the real Internet may change, we leave this problem for our future work.
- We consider only unicast communications.
- Security, multicast, policing and billing are out of scope for this paper and will be considered in our future work.

3 Signaling Protocol

The signaling protocol serves the purpose of establishing a connection between two end systems by reserving capacity in the routers along a path. Related ideas have been proposed in [3], [8-13], [28]. Our protocol differs from these approaches in the following ways: it uses simpler traffic descriptor (we describe a connection solely by a peak rate); the protocol does not use per-flow soft state in core routers and relies instead on garbage collection (see Section 3.3). Further, it uses explicit reservations in comparison with [9]. In contrast to [11], which is similar to our approach in that it uses hard state, our protocol requires fewer messages to establish and maintain reservations; thus, our protocol introduces less overhead. The comprehensive work in [13] as well as efforts of ATM Forum [28] deal with signaling for QoS routing, while our protocol uses existing routing algorithms. Another approach to resource provisioning deals with centralized schemes (bandwidth brokers), like in [6], which is not considered in this paper.

We have the following design goals for our signaling protocol: It should not require per-connection soft-states in the routers; the number of signaling messages should be low, and the time to process a signaling message should be small. The protocol should be robust and tolerate loss of certain types of messages.

The idea behind the signaling is simple and uses the additivity of the reservation state. We keep track of available capacity for each network link in a variable C_{GS}, and have two public state variables for *requested* and *confirmed* capacity, R_r and R_c. All users are able to modify these variables by means of the signaling protocol. To make a reservation the router simply adds the requested rate in the signaling message to the requested capacity variable. This permits the requestor to send data with this rate. The router does the reverse operation to release capacity when the end system terminates the connection by an explicit tear down message. Let us define the requested rate r as a multiple of Δ, where Δ bits per second is a minimum value (quantum) of reservation. The following messages are needed for the signaling:

- Reservation message *mR(r)* for the amount of capacity r. Messages of this kind may be discarded by the routers.
- Confirmation message *mC(r)* from the initiating party for the reserved capacity. Messages of this type may not be lost.
- Tear down message *mT(r)*, which indicates that the initiating party finishes the connection and releases the reserved capacity. Messages of this type may not be lost.

The continued work on the protocol specification includes finding the best possible way of encoding the signaling messages in the fields of the IP header. We believe that the length of a signaling message could be as small as an IP header, 20 bytes, considering that we only need to represent three distinct messages and one rate value.

The service specifies that all signaling messages are carried in the GS part of the link capacity, and they are therefore not subjected to congestion-induced loss.

3.1 End System Behavior

In order to establish a guaranteed service connection, the initiator sends *mR(r)* towards the destination. Each reservation message has a unique identifier. After issuing this message a waiting timer is started with a value *T*. This is the allowed waiting time for an acknowledgment at the sender and it is a constant specified for the service. By defining *T* as maximum *RTT* we ensure correct handling of delayed acknowledgments: If no acknowledgment arrives during *T* seconds, the initiator sends a new reservation request.

An acknowledgment message is issued by the receiver when it receives a reservation request, and it is encoded as a zero-rate reservation (i.e. *mR(0)*) with the same identifier as in the reservation message. For a sender an acknowledgment indicates a successful reservation along the path. The source then sends a confirmation message towards the destination, and may immediately start to send data at a rate below *r* b/s. When the initiating party wishes to end the session, it sends a teardown message, freeing the reserved capacity on the path. The diagram in Fig. 2 represents the behaviors of the sender and the receiver.

The establishment of a connection is repeated until an acknowledgment arrives or a sender decides to stop. If an acknowledgement was delayed in the network and the sender times out before receiving this message, it will issue a new reservation message. The acknowledgment for the old reservation will be ignored. The core routers will handle this case as a lost reservation, and will clean up the reserved capacity during the garbage collection (see Section 3.3).

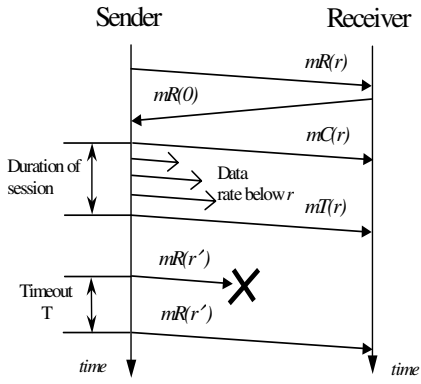


Fig. 2. Setup and tear down of reservations.

```
void server() {
 //types of messages: 0- mR, 1- mC, 2-mT
 r=0; //requested capacity
 Rr=0;//reserved capacity
 Rc=0; //confirmed capacity
 while (1) {
   int type=receive();
   switch type {
     0: if(r>Cgs-Rr) //Admission control
          drop(message);
        else Rr +=r
     1: Rc +=r;
     2: Rr-=r; Rc-=r;
    } //switch type
   if (Rr == Cgs) {
     wait(W);
     /* function wait process only messages
of types 1 and 2 for the duration of W
seconds */
     if (Rr!=Rc) garb_collect();
    } //if (Rr == Cgs)
  } //while (1)
} //void server()
```

Fig. 3. Algorithm for processing signaling messages in a router

3.2 Router Behavior

First when a router receives *mR(r)* it performs an admission control based on the requested rate. The router will drop a reservation if it exceeds the remaining capacity ($r>C_{GS}-R_r$). Otherwise it increases R_r by the bit rate requested in the reservation message. When getting a confirmation message, the router increases the value of R_c

by the rate *r*. When the router gets a teardown message, it decreases both R_r and R_c by the rate in the message. The summary of a router's behavior is presented by the pseudo-code in Fig. 3.

As one could notice from the algorithm, if a router gets an acknowledgment message in the form of *mR(0)*, it will not change the state variables. However, acknowledgments are carried in the unallocated part of the GS capacity, and might therefore be lost due to buffer overflow if that part is nil. Fig. 4 shows the capacity allocation for the signaling messages and the dynamic behavior of R_r and R_c. The message *mR(r)* is carried in the available capacity from which it reserves a share; *mC(r)* and *mT(r)* use the established reservation. Thus, the messages are not subjected to congestion.

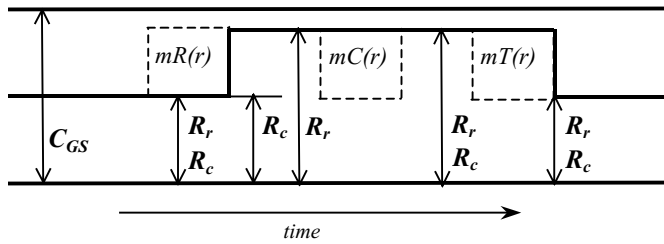


Fig. 4. Dynamic of R_r and R_c.

3.3 Garbage Collection

The reservation message is discarded when a router blocks a reservation request. This means that there will be pending reservations upstream along the path of the message. Thus, we need some garbage collection process to re-establish the reservation levels in the routers that the message has successfully passed.

Since a message of the type *mC()* will not be lost, all successfully acknowledged reservations are expected to be confirmed. The garbage collection starts when all GS capacity is reserved $Rr=C_{GS}$. The router will therefore expect a confirmation for the reserved capacity within a waiting time $W>T$. If some of the requested capacity is not confirmed after *W*, the output port will free the capacity *Rr-Rc*.

Example. Consider the following case. The capacity variable of the outgoing link is $C_{GS}=5$ *Mb/s*. There was no reservation request before T0, so $Rr=Rc=0$.

Table 1 shows the dynamic behavior of C_{GS}, R_r and R_c. Steps T1 to T3 show a successful cycle of a reservation for the amount of 1 Mb/s. The router gets a confirmation message at T2 that makes R_r and R_c equal. This indicates success of the reservation. A tear down message arrives at time T3. At T4 we have a 4 Mb/s requested reservation, and so far have only 2 Mb/s been confirmed. At time T5 the router accepts a 1 Mb/s reservation. Since the amount of requested capacity becomes equal to the available capacity C_{GS}, the garbage collection is started. During the time from T5 to T6, which equals *W* seconds, the router accepts only confirmation and tear down messages; all reservation messages will be discarded. There are several possibilities of what messages may arrive to the router during the garbage collection. One of the cases will be if the router gets confirmations for all unconfirmed capacity.

This indicates that all the capacity on the link is in use and the garbage collection is ended without action. Another case is when some or no confirmations will arrive. In this case the garbage collection procedure at time T6 will realize that the amount of capacity R_r-R_c= 5-2 =3 Mb/s is not confirmed and will free it. Then the router will start accepting new reservations again.

Table 1. Bookkeeping of parameters for reservations.

Time Point	Event	C_{GS}	R_r	R_c
T0	Initial settings	5	0	0
T1	mR(1) arrives	5	1	0
T2	mC(1) arrives	5	1	1
T3	mT(1) arrives	5	0	0
.....				
T4	Settings after some time	5	4	2
T5	mR(1) arrives	5	5	2
T5'	Garbage Collection starts [Accept only mC() and mT()]			
T6	Settings after Garbage Collection [Accept all messages]	5	2	2

The interesting case is when a teardown message will arrive during the garbage collection. Suppose during T5 to T6 the router has processed two 1Mb/s tear down messages.

Table 2. Arrival of tear down during garbage collection.

Time Point	Event	C_{GS}	R_r	R_c
T4	Settings after some time	5	4	2
T5	mR(1) arrives	5	5	2
T5'	Garbage Collection started [Accept only mC() and mT()]			
T1	mT(1) arrives	5	4	1
T2	mT(1) arrives	5	3	0
T2'				
T6	Settings after Garbage collection [Accept all messages]	5	0	0

In Table 2 the changes are reflected. We will denote the time points within this interval with small letters. At t2' we have 2 Mb/s of capacity which can be used by new reservations. One possibility is to stop the garbage collection and accept new

reservations. However it is not efficient: Potentially it may result in the situation where the garbage collection restarts every time a new reservation arrives. Thus we have chosen to let the garbage collection end before accepting new reservations. Note that this choice affects only the duration of the blocking periods, but since it releases more capacity could result in fewer activations of the garbage collection.

3.4 System Requirements and Performance Parameters

The routers need to provide enough buffer space for the signaling messages that cannot be lost. We have assumed that there is a minimum reservation quantum. The worst case is when all capacity available for the guaranteed service will be reserved by the smallest quantum. In this case the total number of supported connections will be equal to C_{GS}/Δ. Since tear down and confirmation messages can be issued only for successful reservations, the total number of loss-intolerant messages will be twice the number of connections. If we assume that the call processing is equally fast in all routers then it is easy to show that a router with n ports requires $Bmax$ packets of buffering per output port:

$$B_{\max} = \frac{2C_{GS}(n-2)}{\Delta(n-1)}. \tag{1}$$

(The buffer size can be reduced in half if the end systems do not send confirmation and tear down messages back to back. This is accomplished by enforcing a minimum duration of a GS connection.)

We have implemented the signaling protocol in the network simulator ns-2 [29]. Our goal was to observe the performance of our protocol in a real router; therefore, we measured the actual time the processor spends on processing every signaling message and not the local ns-2 time, since it is independent of the computer power. In a series of simulations on a processor Intel Pentium III 500 MHz we obtain the following results. The time of processing one reservation message is 35 microseconds, a tear down message takes 32 microseconds, and a confirmation message 17 microseconds. Although the operations of reservation and tear down are similar, recall that the router performs admission control for reservations. This explains the slightly different processing times for these types of messages. The average rate of message processing is therefore 30×10^6 messages per second. Assuming for example a link capacity equal to 100 Mb/s and a reservation quantum of 10 kb/s, then the maximum number of supported connections will be 10000 and therefore a buffer for 20000 messages is needed to hold loss-intolerant signaling messages. This corresponds to 400 kB of memory if each message is 20 bytes.

Summarizing the description of our signaling protocol, we have developed a new simple signaling protocol for the guaranteed service. This protocol has the following properties.

- It is sender oriented.
- It has three types of messages *mR()*, *mC()* and *mT()* and uses four messages to setup, acknowledge, confirm and tear down a connection (with soft state, the number of messages depends on the connection's duration).

- The protocol does not use per-flow state in core routers and the simple operations allow fast message processing.

Further work on this signaling protocol will focus on the following problems:

- Encoding of signaling messages in standard IP header.
- Change of routes in the core network.
- Failure of a connection without explicit tear down.
- Loss of teardown or confirmation messages due to link failure or bit errors.

4. Packet Scheduling in Routers

Many researchers work in the field of scheduling algorithms. A good overview of the existing approaches is given in [26]. Work-conserving algorithms like those in [20], [21] have been extensively studied in [22-25]. Non-work-conserving schemes are described in [18], [19], [27]. In our architecture we use the non-work-conserving scheme described in [15]. Our contribution described in this section is a further evaluation of the chosen scheme, with experimental and formal justification of its suitability in our service architecture.

In order to satisfy the properties of our service model, a router needs a special scheduling algorithm to enforce additivity of the reservation state and to ensure absence of packet loss for the GS traffic in the network. It should work with variable length packets up to some maximum size *L*. The algorithm should not prevent best effort traffic from being sent for long periods. The buffer space for the GS traffic in the routers should be enough to avoid packet loss, taking into account the worst arrival pattern.

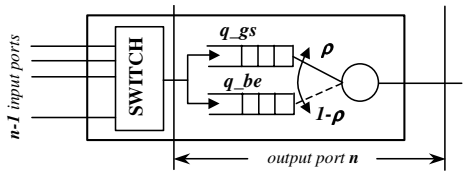


Fig. 5. An output port of a router.

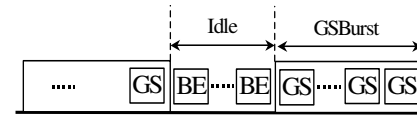


Fig. 6. Scheduling.

4.1 The Algorithm

Consider a router in the network as shown in Fig. 5. The router has *n* ports. Traffic for output *n* enters the router through *n-1* input ports. After the route lookup, packets come to the switching fabric and are switched to the appropriate output ports. At an output port, all incoming packets are sorted according to the service class and placed into the corresponding queue: one for guaranteed service (q_gs) and one for best effort service (q_be).

The scheduler will serve one or more packets from the GS queue (a *GSBurst*), then it will schedule an idle period long enough to preserve the outgoing reserved rate of GS traffic. During the idle it will serve packets from the best effort queue. Note that

proposed scheduling scheme is non-work-conserving only with respect to GS packets. Best effort traffic gets all the capacity that is not used by GS. The schematic in Fig. 6 shows this scheduling. Thus we define the reservation ratio of an outgoing link as

$$\rho = \frac{GSBurst}{Idle + GSBurst}. \tag{2}$$

For a GS burst the duration of the idle period is

$$Idle = \frac{1-\rho}{\rho} GSBurst \ . \tag{3}$$

Note that with increasing ρ up to 1 the idle period tends to 0. Under such circumstances BE traffic will be blocked. Thus, we compute the smallest length of a *GSBurst* so that ρ is maintained and the following idle period is sufficiently long to transmit one BE packet of the maximum size *L*. Thus, the burst size is at most $\frac{\rho}{1-\rho} L$ bytes long. The pseudo-code of the scheduling algorithm is presented in [15].

The system in Fig. 5 has a drawback; namely, having our scheduling only at an output port, we cannot control the number of packets that might bunch together due to the multiplexing of flows from the input ports. The outgoing aggregate is smooth but the output scheduling does not control the behavior of each subaggregate.

Although the aggregate flow from a router does not contain bursts, we do not have information about how individual flows will be routed in the next router

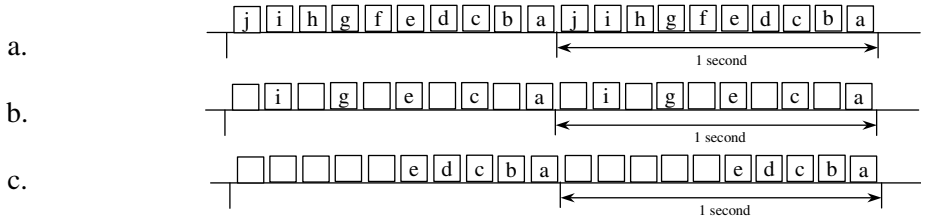


Fig. 7. Burst of a subaggregate.

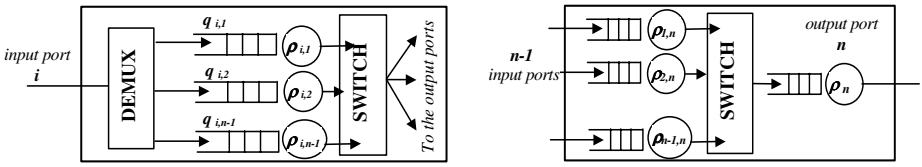
In Fig. 7a we have an aggregate flow of 10 packets per second (pps), observed at the output of a router during 2 seconds. The aggregate consists of 10 individual connections, each with rate 1 pps. All the connections are smooth. Now, consider this aggregate arriving to the demultiplexor of the next router. Assume that 5 pps of this aggregate should be forwarded to output port 1. The best case for the corresponding queue will be when the subaggregate of 5 pps will be smooth. This will happen if connections *a,c,e,g,i* are destined to port 1 (Fig. 7b). But it could also happen that the 5pps subaggregate will consists of flows *a,b,c,d,e* (Fig. 7c). The worst-case subaggregate is formed when the aggregate consists of connections with the minimum reservation rate.

We can eliminate this problem by adding similar schedulers at each input port. Their purpose is to smooth out the distortion of the subaggregates introduced by the

upstream router. This will allow us to dimension the buffers needed in the routers for zero packet loss.

4.2 Buffer Size Evaluation

We add (n-1) rate controllers at each input port as shown in Fig. 8a. Each controller is responsible for smoothing out the subaggregate directed to a particular output port. Denote the controller at the input port *i* for the output port *j* as the *(i,j)*-controller. The rate controller here is the same scheduler as described in the previous section. At each output port there will be a scheduler for the purpose of shaping the multiplexed flows from all the input ports, and for providing a fair service to the best effort traffic. Hence from the point of view of an output port the architecture looks as in Fig. 8b.



a. An input port and the switching fabric. b. The input port, the fabric and an output port.

Fig. 8. Router architecture (GS-aware devices).

In this architecture we need a small buffer at the output, only *(n-2)L* bytes long, in order to keep packets arriving simultaneously from the input ports. By having the rate controllers at the input we will never have bursty arrival to the queue of the output port. Now we have obtained the following:

- There is no clumping of packets from a particular input port.
- An aggregated output flow is smooth.
- While certain subaggregates of the output flow can be bursty, the burst is finite and easy to compute.

Having a rate controller at the input ensures the first property. Although a subaggregate can be distorted at the output due to multiplexing with flows from other input ports, it will be reshaped at the input of the downstream router. The presence of a non-work-conserving scheduler at the output ensures the second property. The observations about the third property were made in section 4.1.

We will compute the needed buffer for one *(i,j)*-controller at the input. Denote the reservation level for the flows originated from the port *i* on the outgoing link *j* as $\rho_{i,j}$. The rate of the *(i,j)*-controller is $\rho_{i,j}C_{GS}$, the minimum reservation rate is Δ, and the capacity available for GS is C_{GS}. Recall that in the worst case all the connections in the subaggregate will have minimum reservation rates. The number of connections will then be $N=\rho_{i,j}C_{GS}/\Delta$. The worst-case burst of the subaggregate arriving to $q_{i,j}$ is N packets with the rate of the aggregate (as in Fig. 7c). For the maximum size packet L, the needed buffer is:

$$B(\rho_{i,j}) = L\frac{C_{GS}\rho_{i,j}}{\Delta}(1-\rho_{i,j}) \ . \tag{4}$$

As an example we have computed the size of one controller queue in the router with the following parameters. In the attached links 100 Mb/s is available for the guaranteed service (C_{GS}=100Mb/s), the minimum reservation rate Δ is 10 kb/s, the maximum packet size L is 576 bytes. In Fig. 9 the size is shown of an input buffer depending on the level of reservation $\rho_{i,j}$.

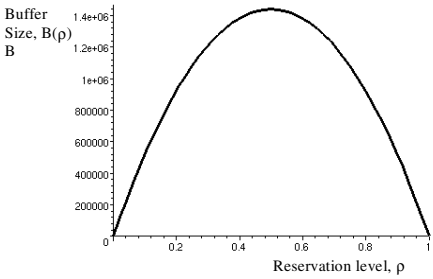


Fig. 9. Buffer size as a function of reservation level.

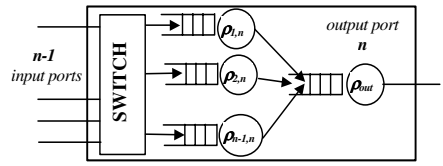


Fig. 10. Router architecture (implementation).

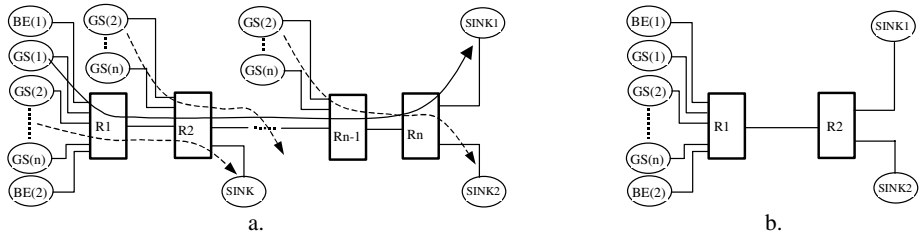


Fig. 11. Network topology for the simulations.

In the implementation we have moved all the rate controllers from the input ports to the corresponding output port. Fig. 10 shows this structure.

The scheme shown in Fig. 8 is functionally equivalent to the one in Fig. 10. In this work we are not concerned with the scheduling for the switching fabric. The only purpose of the controllers at the input is to prevent bursty arrivals to the buffer at the output port. From Fig. 9 one can see that the maximum buffer size is needed when the rate controller is setup to 0.5 of the outgoing link. In our example we need at most 1.4 MB of buffer in this controller. Assume the router has 50 input ports. The maximum size buffer for output port n will be when the output rate will be equally distributed between all input ports. In this case the total memory needed for buffers at the output port is $(n-1)B(\frac{\rho_n}{n-1}) + (n-1)L - L$, where n is the number of ports in the router and $B(\ )$ is given in Eq. 4. In our case it will be 5.7MB.

4.3 Simulation and Worst Case Performance Analysis

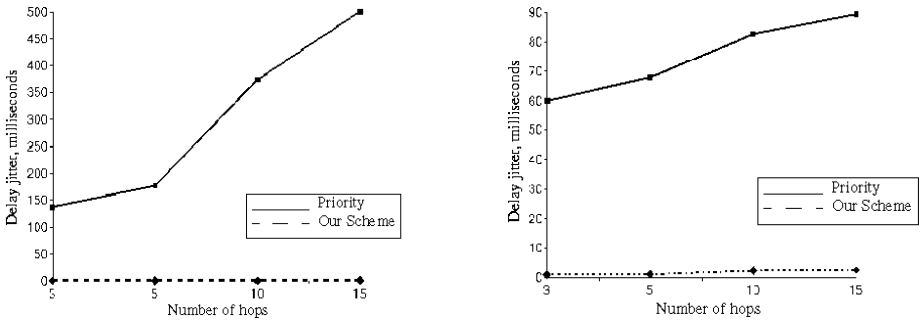
In this section we present simulations to evaluate the delay and fairness properties of our scheduling. Our scheme provides a fair service for best effort traffic even when the reservation level is higher than 50 percent. We compare the performance of our scheduler to a strict priority scheduler. Both schemes were implemented in the network simulator NS-2 (we are working on an implementation of the proposed architecture in a Linux-based router, and will report the results in our future publications).

We used the topology in Fig. 11a to study the delay properties of our scheme. The goal was to measure the end-to-end delay of one GS flow. The following setup was used: R(1) to R(N) are the routers, BE(1) and BE(2) are two best effort sources, GS(1) is the monitored GS source, GS(2) up to GS(N) are background GS sources. All links in the network are 100Mb/s. A flow from GS(1) traverses the path towards SINK1. The best effort flows from BE(1) and BE(2) follow the path towards SINK2. The background GS traffic follows the path towards a SINK connected to the next downstream router. We ran simulations with both VBR and CBR traffic. In the case of CBR traffic, the sources have the following characteristics: GS(1) is set to 4Mb/s; the rest of the capacity available for the guaranteed flows is equally distributed between sources GS(2) up to GS(N); the best effort flow from BE(1) is 10 Mb/s; the rest of the capacity available for the best effort flows is assigned to BE(2). In the case of VBR traffic, all GS sources are exponential on-off sources. Their peak rates are the same as in the CBR case. The mean duration of on/off periods is 500 milliseconds.

Initially experiments were done with VBR traffic. The number of hops between GS(1) and SINK1 was 3, 5, 10, and 15.We used 10 background GS sources in this experiment, feeding each router. The monitored flow of 4Mb/s from GS(1) was subjected to the same kind of background traffic in every router along the path. We conducted two experiments, firstly with a reservation of 80 percent on all links, and second with 20 percent. The goal was to measure the end-to-end delay of the monitored flow. Then we repeated the simulations for the CBR traffic.

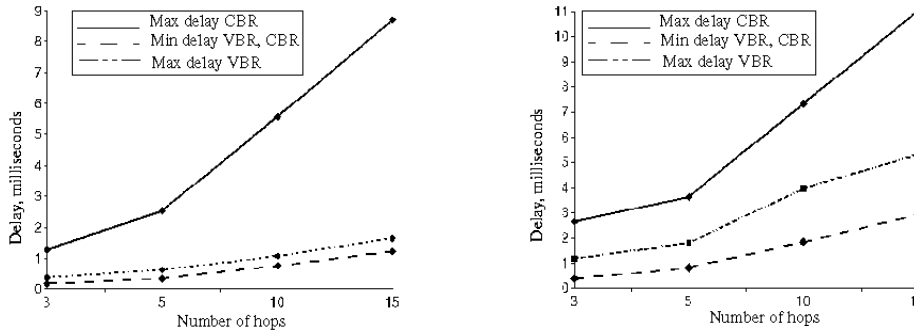
Fig. 12 shows the delay jitter (the difference between the maximum and minimum delays) of the monitored GS flow for the VBR case. Simulations show for both values of the reservation level that our scheme introduces much smaller jitter than strict priority scheduling. This is due to the difference between the two schemes. Using strict priority scheduling a burst of packets can be created due to the effect of flow multiplexing. This phenomenon may increase with number of hops. While in our scheme the multiplexed flows from the input ports in a router are reshaped and smooth. By looking at the maximum and minimum delays for our scheme we have seen that they grow almost linearly with the number of hops. Fig. 13 shows this. This explains the small variations in the delay jitter. In the case of ρ=0.8 the maximum jitter is 0.414 milliseconds after 15 hops, in the case where ρ=0.2 this value is 2.3 milliseconds. Reservation values of less than 0.5 (in our case 0.2), lead to packets being delayed for one or more idle periods. This accounts for on higher jitter value. We have studied the maximum delay under the worst case arrival of the GS traffic in CBR experiments. The maximum delay of the 4Mb/s after 15 routers with 10 input ports each is 11 milliseconds (excluding the propagation time on the links).

We decided to continue our comparison of the two schemes using a different topology. In this case we did not want to include the hop count. We studied the impact of the two schemes on a monitored best effort flow. For this purpose we used the topology in Fig. 11b. The description of the sources, routers, links as well as traffic specification is the same as in the previous setup. The monitored BE flow from BE(1) follows the path towards SINK1, and other flows towards SINK2. The number of GS sources was varied from 10 to 100. The results of the experiments with ρ=0.8 and ρ=0.2 are shown in Fig. 14. The following observation was drawn from the simulations: For both values of the reservation level our scheme performs better with regard to best effort flows. This is due to the difference between the two scheduling schemes: Under strict priority scheduling GS traffic gets full priority over best effort traffic, therefore the BE packets can only be transmitted when there are no packets in the GS queue, while in our scheme, the best effort traffic gets certain guarantees to be transmitted even in case of simultaneous arrivals of GS packets.



ρ=0.8 ρ=0.2

Fig. 12. End-to-end jitter of the monitored GS flow for the VBR case.



ρ=0.8 ρ=0.2

Fig. 13. Min and max delays for GS flow for VBR and CBR.

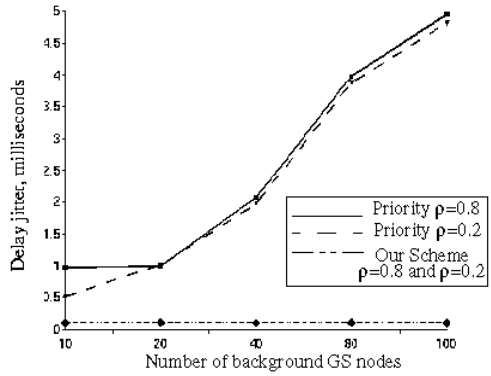


Fig. 14. Delay jitter of the monitored best effort flow.

5 Open Issues and Summary

The ongoing work on the architecture continues in the following ways. Current specification of the signaling protocol does not support loss of other kinds of messages than reservation messages. Since a teardown message might be lost, due to link failure for example, all the nodes after the point of failure will never receive the message and therefore the amount of capacity, which was supposed to be deallocated, will remain in use. This situation can potentially lead to a situation in which all the capacity in a router is unavailable for new reservations. To eliminate this we are developing a more advanced garbage collection, to increase the robustness of the protocol. We also develop a mechanism in the signaling protocol which will handle re-routing. Obviously forwarding rerouted GS traffic without reservation is unacceptable.

The design in this paper was concerned only with unicast communication. Another direction of our work is to consider multicast GS flows. We are working on an implementation of the described architecture in a Linux-based platform. We intend to consider the details of the administration policy, security and pricing issues of our service.

We have presented a simplified guaranteed service for the Internet. Our contribution is as follows. We suggest a simpler way of flow description based on a fixed rate. This along with a new signaling protocol and non-work-conserving scheduling gives a basis for capacity reservation and admission control. The simulation results show good performance of the service. The simplicity of the overall architecture certainly adds to the possibility of implementing it in high-speed routers.

References

1. R. Braden, D. Clark, S. Shenker, "Integrated services in the Internet architecture: an overview", RFC1633, IETF, June 1994.
2. S. Blake, D.Black, M.Carlson, E.Davies, Z. Wang, W. Weiss, "An architecture for differentiated services", RFC 2475, IETF, December 1998
3. R. Braden, L. Zhang, S. Berson, S. Herzog, S. Jamin, "Resource reservation protocol (RSVP)", RFC 2205, IETF, September 1997
4. Y. Bernet, P. Ford, R. Yavatkar, F. Baker, L. Zhang, M. Speer, R. Braden, B. Davie, J. Wroclawski, E. Felstaine, "A framework for integrated services operation over diffserv networks", RFC 2998, IETF, November 2000
5. V. Jacobson, K. Poduri, "An expedited forwarding PHB", RFC 2598, IETF, June 1999
6. K.Nichols, V. Jacobson, L.Zhang, "A two-bit differentiated service architecture for the Internet", RFC 2638, IETF, July 1999.
7. J. Harju, P. Kivimaki, "Co-operation and comparison of diffServ and intserv: performance measurements", *in Proc. of 25th Annual IEEE Conference on Local Computer Networks*, 2000, pp. 177 – 186
8. P. Pan, H. Schulzrinne, "YESSIR: a simple reservation mechanism for the Internet", *Computer Communication Review*, vol. 29, No. 2, April 1999
9. W. Almesberger, T. Ferrari, J.-Y. Le Boudec, "SRP: a scalable resource reservation protocol for the Internet", *Sixth International Workshop on Quality of Service (IWQoS 98)*, 1998, pp. 107 –116
10. G. Feher, K. Nemeth, M. Maliosz, I. Cselenyi, J. Bergkvist, D. Ahlard, T. Engborg, "Boomerang - a simple protocol for resource reservation in IP networks", *IEEE Workshop on QoS Support for Real-Time Internet Application*, June 1999.
11. A. Eriksson, C.Gehrman, "Robust and secure light-weight resource reservation for unicast IP traffic", *Sixth International Workshop on Quality of Service (IWQoS 98)*, 1998, pp. 168-170
12. T.W.K. Chung, H.C.B. Chang, V.C.M. Leung, "Flow initiation and reservation tree (FIRST)", *IEEE Conference on Communications, Computers, and Signal Processing*, pp. 361-364, 1999.
13. R.A. Guerin, A. Orda, "Networks with advance reservations: the routing prespective", *in Proc. of INFOCOM 2000*, pp. 118-127.
14. G. Karlsson, F. Orava, "The DIY approach to QoS", *Seventh International Workshop on Quality of Service (IWQoS 99)*, 1999, pp. 6 –8
15. M. Mowbray, G. Karlsson, T. Köhler, "Capacity reservation for multimedia traffics", *Distributed Systems Engineering*, vol. 5, 1998, pp. 12-18
16. V.Elek, G.Karlsson, R. Rönngren, "Admission control based on end-to-end measurements", *in Proc. of INFOCOM 2000*, pp. 623-630.
17. I. Mas Ivars, G. Karlsson, "PBAC: probe based admission control", *In Proc.of Second International Workshop on Quality of Future Internet Services (QoFIS 2001)*, pp.97-109, 2001.
18. S.J. Golestani, "A framing strategy for congestion management", *IEEE Journal on Selected Areas in Communications*, vol. 97, September 1991, pp. 1064 -1077.
19. R. Brown, "Calendar queues: a fast O(1) priority queue implementation for the simulation event set problem", *Communications of the ACM*, 31 (10), pp. 1220-1227, Oct. 1988.
20. J.C.R Bennett, H. Zhang, "WF^2Q: Worst-case fair weighted queueing", *In Proc. of INFOCOM 1996*, pp. 120-128.
21. S. Golestani, "A self-clocked fair queueing scheme for broadband applications", *In Proc. of INFOCOM 1994*, pp. 636-646.
22. R.L. Cruz, "A calculus for network delay. II. Network analysis", *IEEE Transactions on Information Theory*, Vol. 37, Issue 1, Jan. 1991, pp. 132-141.

23. R.L. Cruz.,"A calculus for network delay. I. Network elements in isolation", *IEEE Transactions on Information Theory*, Vol. 37, Issue 1, Jan. 1991, pp. 114-131.
24. L. Georgiadis, R.A. Guerin, A. Parekh, "Optimal multiplexing on a single link: delay and buffer requirements", *IEEE Transactions on Information Theory*, Vol. 43, No. 5, pp. 1518-1535, Sept 1997.
25. J.Y. Le Boudec, P.Thiran, *Network Calculus*, Springer Verlag LNCS 2050, June 2001.
26. H. Zhang, "Service disciplines for guaranteed performance service in packet-switching networks", *Proceedings of IEEE*, Vol. 83(10), pp. 1374-1396, Oct.1995.
27. H. Zhang, "Providing end-to-end performance guarantees using non-work-conserving disciplines", *Computer Communications: Special Issue on System Support for Multimedia Computing*, 18 (10), Oct. 1995.
28. ATM Forum, http://www.atmforum.com
29. Network Simulator ns-2, http://www.isi.edu/nsnam/ns/
30. Next Steps in Signaling (NSIS), IETF working group, http://www.ietf.org/html.charters/nsis-charter.html