SRP: a Scalable Resource Reservation Protocol for the Internet

Werner Almesberger[1], Tiziana Ferrari[2], and Jean-Yves Le Boudec[1]

[1] EPFL ICA, INN (Ecublens), CH-1015 Lausanne, Switzerland
[2] DEIS, University of Bologna, viale Risorgimento, 2, I-40136 Bologna, Italy; and Italian National Inst. for Nuclear Physics/CNAF, viale Berti Pichat, 6/2, I-40127 Bologna, Italy

Abstract. The Scalable Reservation Protocol (SRP) provides a light-weight reservation mechanism for adaptive multimedia applications. Our main focus is on good scalability to very large numbers of individual flows. End systems (i.e. senders and destinations) actively participate in maintaining reservations, but routers can still control their conformance. Routers aggregate flows and monitor the aggregate to estimate the local resources needed to support present and new reservations. There is neither explicit signaling of flow parameters, nor do routers maintain per-flow state.

1 Introduction

Many adaptive multimedia applications [1] require a well-defined fraction of their traffic to reach the destination and to do so in a timely way. We call this fraction the *minimum rate* these applications need in order to operate properly. SRP aims to allow such applications to make a dependable reservation of their minimum rate.

The sender can expect that, as long as it adheres to the agreed-upon profile, no *reserved* packets will be lost due to congestion. Furthermore, forwarding of *reserved* packets will have priority over best-effort traffic.

Traditional resource reservation architectures that have been proposed for integrated service networks (RSVP [2], ST-2 [3], Tenet [4], ATM [5,6], etc.) all have in common that intermediate systems (routers or switches) need to store per-flow state information. The more recently designed Differentiated Services architecture [7] offers improved scalability by aggregating flows and by maintaining state information only for such aggregates. SRP extends upon simple aggregation by providing a means for reserving network resources in routers along the paths flows take.

Recently, hybrid approaches combining RSVP and Differentiated Services have been proposed (e.g. [8]) to overcome the scalability problems of RSVP. Unlike SRP, which runs end-to-end, they require a mapping of the INTSERV services onto the underlying Differentiated Services network, and a means to tunnel RSVP signaling information through network regions where QoS is provided using Differentiated Services.

Reservation mechanism In short, our reservation model works as follows. A source that wishes to make a reservation starts by sending data packets marked as *request* packets to the destination. Packets marked as *request* are subject to packet admission control by routers, based on the following principle. Routers monitor the aggregate flows of *reserved* packets and maintain a running estimate of what level of resources is required to serve them with a good quality of service. The resources are bandwidth and buffer on outgoing links, plus any internal resources as required by the router architecture. Quality of service is loss ratio and delay, and is defined statically. When receiving a *request* packet, a router determines whether hypothetically adding this packet to the flow of *reserved* packets would yield an acceptable value of the estimator. If so, the *request* packet is accepted and forwarded towards the destination, while still keeping the status of a *request* packet; the router must also update the estimator as if the packet had been received as *reserved*. In the opposite case, the *request* packet is degraded and forwarded towards the destination, and the estimator is not updated. Degrading a *request* packet means assigning it a lower traffic class, such as best-effort. A packet sent as *request* will reach the destination as *request* only if all routers along the path have accepted the packet as *request*. Note that the choice of an estimation method is local to a router and actual estimators may differ in their principle of operation.

The destination periodically sends feedback to the source indicating the rate at which *request* and *reserved* packets have been received. This feedback does not receive any special treatment in the network (except possibly for policing, see below. Upon reception of the feedback, the source can send packets marked as *reserved* according to a profile derived from the rate indicated in the feedback. If necessary, the source may continue to send more *request* packets in an attempt to increase the rate that will be indicated in subsequent feedback messages.

Thus, in essence, a router accepting to forward a *request* packet as *request* allows the source to send more *reserved* packets in the future; it is thus a form of implicit reservation.

Aggregation Routers aggregate flows on output ports, and possibly on any contention point as required by their internal architecture. They use estimator algorithms for each aggregated flow to determine their current reservation levels and to predict the impact of accepting *request* packets. The exact definition of what constitutes an aggregated flow is local to a router.

Likewise, senders and sources treat all flows between each pair of them as a single aggregate and use estimator algorithms for characterizing them. The estimator algorithms in routers and hosts do not need to be the same. In fact, we expect hosts to implement a fairly simple algorithm, while estimator algorithms in routers may evolve independently over time.

Fairness and security Denial-of-service conditions may arise if flows can reserve disproportional amounts of resources or if flows can exceed their reservations. We presently consider fairness in accepting reservations a local policy issue (much like billing) which may be addressed at a future time.

Sources violating the agreed upon reservations are a real threat and need to be policed. A scalable policing mechanism to allow routers to identify non-conformant flows based on certain heuristics is the subject of ongoing research. Such a mechanism can be combined with more traditional approaches, e.g. policing of individual flows at locations where scalability is less important, e.g. at network edges.

The rest of this paper is organized as follows. Section 2 provides a more detailed protocol overview. Section 3 describes a simple algorithm for the implementation of the traffic estimator. Finally, protocol operation is illustrated with some simulation results in section 4 and the paper concludes with section 5.

2 Architecture overview

The proposed architecture uses two protocols to manage reservations: a reservation protocol to establish and maintain them, and a feedback protocol to inform the sender about the reservation status.

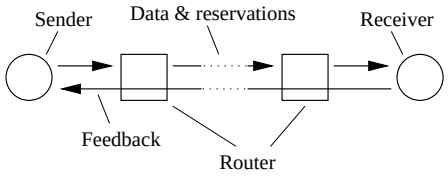

Fig. 1. Overview of the components in SRP.

Figure 1 illustrates the operation of the two protocols:

- Data packets with reservation information are sent from the sender to the receiver. The reservation information consists in a packet type which can take three values, one of them being ordinary best-effort (section 2.2). It is processed by routers, and may be modified by routers. Routers may also discard packets (section 2.1).
- The receiver sends feedback information back to the sender. Routers only forward this information; they don't need to process it (section 2.3).

Routers monitor the reserved traffic which is effectively present and adjust their global state information accordingly. Sections 2.1 to 2.3 illustrate the reservation and feedback protocol.

2.1 Reservation protocol

The reservation protocol is used in the direction from the sender to the receiver. It is implemented by the sender, the receiver, and the routers between them. As mentioned earlier, the reservation information is a packet type which may take three values:

Request This packet is part of a flow which is trying to gain *reserved* status. Routers may accept, degrade or reject such packets. When routers accept some *request* packets, then they commit to accept in the future a flow of reserved packets at the same rate. The exact definition of the rate is part of the estimator module.

Reserved This label identifies packets which are inside the source's profile and are allowed to make use of the reservation previously established by *request* packets. Given a correct estimation, routers should never discard *reserved* packets because of resource shortage.

Best effort No reservation is attempted by this packet.

Packet types are initially assigned by the sender, as shown in figure 2. A traffic source (i.e. the application) specifies for each packet if that packet needs a reservation. If no reservation is necessary, the packet is simply sent as *best-effort*. If a reservation is needed, the protocol entity checks if an already established reservation at the source covers the current packet. If so, the packet is sent as *reserved*, otherwise an additional reservation is requested by sending the packet as *request*.

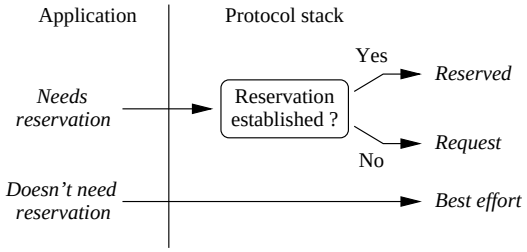

Fig. 2. Initial packet type assignment by sender.

Each router performs two processing steps (see also figure 3). First, for each *request* and *reserved* packet the estimator updates its current estimate of the resources used by the aggregate flows and decides whether to accept the packet (packet admission control). Then, packets are processed by various schedulers and queue managers inside the router.

- When a *reserved* packet is received, the estimator updates the resource estimation. The packet is automatically forwarded unchanged to the sched-

uler where it will have priority over best-effort traffic and normally is not discarded.

- When a *request* packet is received, then the estimator checks whether accepting the packet will not exceed the available resources. If the packet can be accepted, its *request* label is not modified. If the packet cannot be accepted, then it is degraded to best-effort
- If a scheduler or queue manager cannot accept a reserved or request packet, then the packet is either discarded or downgraded to *best-effort*.

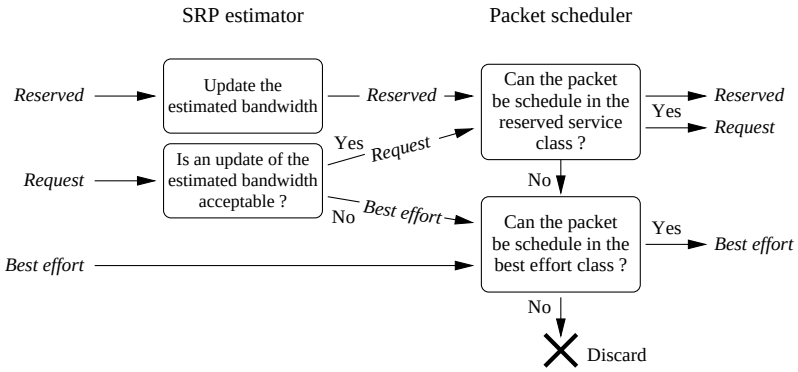

Fig. 3. Packet processing by routers.

Note that the reservation protocol may "tunnel" through routers that don't implement reservations. This allows the use of unmodified equipment in parts of the network which are dimensioned such that congestion is not a problem.

2.2 Packet type encoding

RFC2474 [9] defines the use of an octet in the IPv4 and IPv6 header for Differentiated Services (DS). This field contains the DS Code Point (DSCP), which determines how the respective packet is to be treated by routers (Per-Hop Behaviour, PHB). Routers are allowed to change the content of a packet's DS field (e.g. to select a different PHB).

As illustrated in figure 4, SRP packet types can be expressed by introducing two new PHBs (for *request* and for *reserved*), and by using the pre-defined DSCP value 0 for best-effort. DSCP values for *request* and *reserved* can be allocated locally in each DS domain.

2.3 Feedback protocol

The feedback protocol is used to convey information on the success of reservations and on the network status from the receiver to the sender. Unlike the

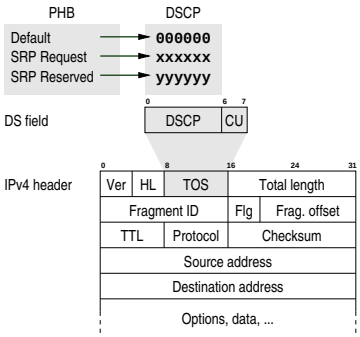

Fig. 4. Packet type encoding using Differentiated Services (IPv4 example).

reservation protocol, the feedback protocol does not need to be interpreted by routers, because they can determine the reservation status from the sender's choice of packet types.

Feedback information is collected by the receiver and it is periodically sent to the sender. The feedback consists of the number of bytes in *request* and *reserved* packets that have reached the receiver, and the local time at the receiver at which the feedback message was generated.

Receivers collect feedback information independently for each sender and senders maintain the reservation state independently for each receiver. Note that, if more than one flow to the same destination exists, attribution of reservations is a local decision at the source.

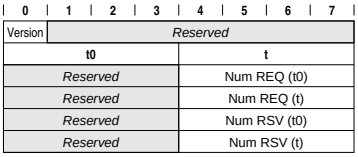

Fig. 5. Feedback message format.

Figure 5 illustrates the content of a feedback message: the time when the message was generated (t), and the number of bytes in *request* and *reserved* packets received at the destination (REQ and RSV). All counters wrap back to zero when they overflow.

In order to improve tolerance to packet loss, also the information sent in the previous feedback message (at time t0) is repeated. Portions of the message are reserved to allow for future extensions.

2.4 Shaping at the sender

The sender decides whether packets are sent as *reserved* or *request* based on its own estimate of the reservation it has requested and on the level of reservation along the path that has been confirmed via the feedback protocol. A source always uses the minimum of these two parameters to determine the appropriate output traffic profile.

Furthermore, the sender needs to filter out small differences between the actual reservation and the feedback in order to avoid reservations from drifting, and it must also ensure that *request* packets do not interfere with congestion-controlled traffic (e.g. TCP) in an unfair way [10].

2.5 Example

Figure 6 provides the overall picture of the reservation and feedback protocols for two end-systems connected through routers $R1$ and $R2$. The initial resource acquisition phase is followed by the generation of request packets after the first feedback message arrives. Dotted arrows correspond to degraded *request* packets, which passed the admission control test at router $R1$ but could not be accepted at router $R2$ because of resource shortage. Degradation of *request*s is taken into account by the feedback protocol. After receiving the feedback information the source sends *reserved* packets at an appropriate rate, which is smaller than the one at which *request* packets were generated.

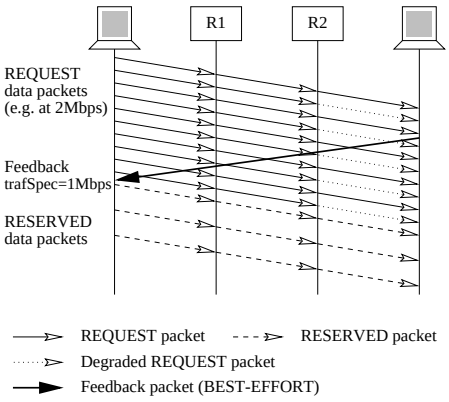

Fig. 6. Reservation and feedback protocol diagram.

2.6 Multicast

In order to support multicast traffic, we have proposed a design that slightly extends the reservation mechanism described in this sections. Refinement of

this design is still the subject of ongoing work. A detailed description of the proposed mechanism can be found in [11].

3 Estimation modules

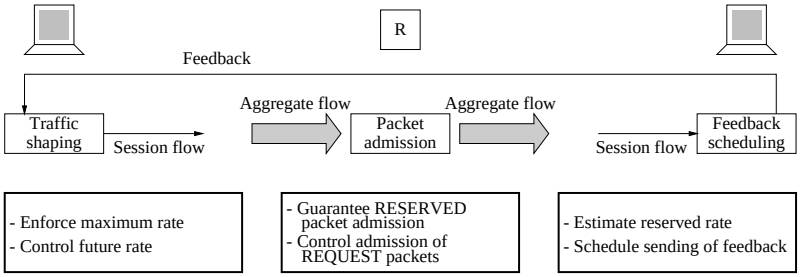

Fig. 7. Use of estimators at senders, routers, and receivers

We call *estimator* the algorithm which attempts to calculate the amount of resources that need to be reserved. The estimation measures the number of *request*s sent by sources and the number of *reserved* packets which actually make use of the reservation.

Estimators are used for several functions.

- Senders use the estimator for an optimistic prediction of the reservation the network will perform for the traffic they emit. This, in conjunction with feedback received from the receiver, is used to decide whether to send *request* or *reserved* packets.
- Routers use the estimator for packet-wise admission control and perhaps also to detect anomalies.
- In receivers, the estimator is fed with the received traffic and it generates an estimate of the reservation at the last router. This is used to schedule the sending of feedback messages to the source.

Figure 7 shows how the estimator algorithm is used in all network elements.

As described in section 2.1, a sender keeps on sending *request*s until successful reservation setup is indicated with a feedback packet, i.e. even until after the desired amount of resources has been reserved in the network. It's the feedback that is returned to the sender, which indicates the right allocation obtained on the path. When the source is feedback-compliant, the routers on the path start releasing a part of the over-estimated reservation already allocated. The feedback that is returned to the sender may also show

an increased number of requests. The sender must not interpret those requests as a direct increase of the reservation. Instead, the sender estimator must correct the feedback information accordingly, which is achieved through the computation of the minimum of the feedback and of the resource amount requested by the source.

Our architecture is independent of the specific algorithm used to implement the estimator. Sections 3.1 and 3.2 describe two different solutions. The definition and evaluation of algorithms for reservation calculation in hosts and routers is still ongoing work. A detailed analysis of the estimation algorithms and additional improvements can be found in [12].

3.1 Basic estimation algorithm

The basic algorithm we present here is suitable for sources and destinations, and could be used as a rough estimator by routers. This estimator counts the number of requests it receives (and accepts) during a certain *observation interval* and uses this as an estimate for the bandwidth that will be used in future intervals of the same duration.

In addition to requests for new reservations, the use of existing reservations needs to be measured too. This way, reservations of sources that stop sending or that decrease their sending rate can automatically be removed. For this purpose the use of reservations can be simply measured by counting the number of *reserved* packets that are received in a certain interval.

To compensate for deviations caused by delay variations, spurious packet loss (e.g. in a best-effort part of the network), etc., reservations can be "held" for more than one observation interval. This can be accomplished by remembering the observed traffic over several intervals and using the maximum of these values (step 3 of the following algorithm). Given a hold time of h observation intervals, the maximum amount of resources which can be allocated Max, res and req (the total number of *reserved* and *request bytes* received in a given observation interval), the reservation R (in bytes) is computed by a router as follows. Given a packet of n bytes:

```
if (packet_type == REQ)
    if (R + req + n < Max) {
        accept;
        req = req + n;   // step 1
    }
    else degrade;

if (packet_type == RES)
    if (res + n < R) {
        accept;
        res = res + n;   // step 2
    }
    else degrade;
```

where initially $R, res, req = 0$. At the end of each observation cycle the following steps are computed:

```
for (i = h; i > 1; i--) R[i] = R[i-1];
R[1] = res + req;
R = max(R[h],R[h-1],...,R[1]); // step 3
res = req = 0;
```

The same algorithm can be run by the destination with the only difference that no admission checks are needed.

Examples of the operation of the basic algorithm are shown in section 4.1.

This easy algorithm presents several problems. First of all, the choice of the right value of the observation interval is critical and difficult. Small values make the estimation dependent on bursts of *reserved* or *request* packets and cause an overestimation of the resources needed. On the other hand, large intervals make the estimator react slowly to changes in the traffic profile. Then, the strictness of traffic acceptance control is fixed, while adaptivity would be highly desirable in order to make the allocation of new resources stricter as the amount of resources reserved gets closer to the maximum. These problems can be solved by devising an adaptive enhanced algorithm like the one described in the following section.

3.2 Enhanced estimation algorithm

Instead of using the same estimator in every network component, we can enhance the previous approach so that senders and receivers still run the simple algorithm described above, while routers implement an improved estimator.

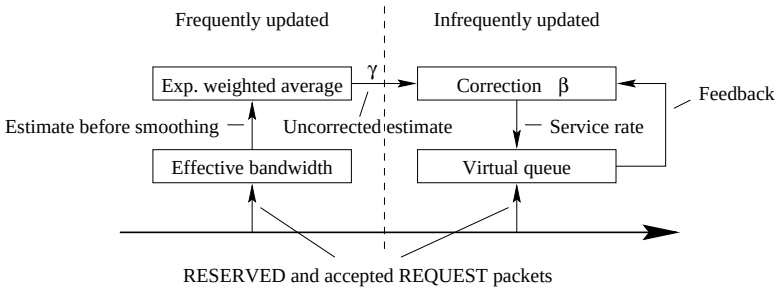

Fig. 8. Schematic design of an adaptive estimator.

We describe an example algorithm in detail in [11]. It consists of the principal components illustrated in figure 8: the effective bandwidth used by *reserved* and accepted *request* packets is measured and then smoothed by calculating an exponentially weighted average (γ). This calculation is performed for every single packet.

The estimate γ is multiplied with a correction factor β in order to correct for systematic errors in the estimation. Packets are added to a virtual queue (i.e. a counter), which is emptied at the estimated rate. If the estimate is too high, the virtual queue shrinks. If the estimate is too low, the virtual queue grows. Based on the size of the virtual queue, β can be adjusted.

4 Simulation

Section 4.1 provides a theoretic description of the behavior of the reservation mechanism in a very simple example, while section 4.2 shows the simulated behavior of the proposed architecture.

4.1 Reservation example

The network we use to illustrate the operation of the reservation mechanism, is shown in figure 9: the sender sends over a delay-less link to the router, which performs the reservation and forwards the traffic over a link with a delay of two time units to the receiver. The receiver periodically returns feedback to the sender.

The sender and the receiver both use the basic estimator algorithm described in section 3.1. The router may – and typically will – use a different algorithm (e.g. the one described in section 3.2).

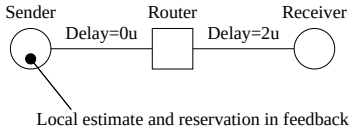

Fig. 9. Example network configuration.

The bandwidth estimate at the source and the reservation that has been acknowledged in a feedback message from the receiver are measured. In figure 10, they are shown with a thin continuous line and a thick dashed line, respectively. The packets emitted by the source are indicated by arrows on the reservation line. A full arrow head corresponds to *request* packets, an empty arrow head corresponds to *reserved* packets. For simplicity, the sender and the receiver use exactly the same observation interval in this example, and the feedback rate is constant.

The source sends one packet per time unit. First, the source can only send requests and the router reserves some resources for each of them. At point (1), the estimator discovers that it has established a reservation for six packets in four time units, but that the source has only sent four packets in this interval. Therefore, it corrects its estimate and proceeds. The first

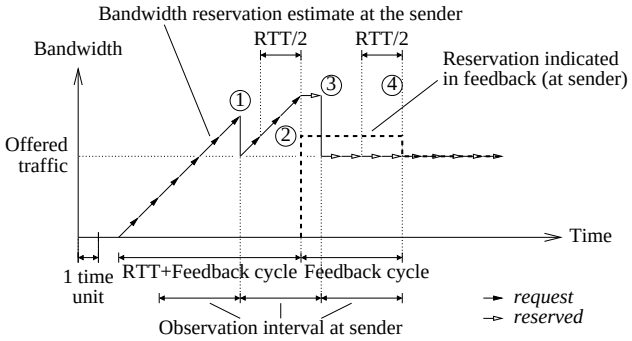

Fig. 10. Basic estimator example.

feedback message reaches the sender at point (2). It indicates a reservation level of five packets in four time units (i.e. the estimate at the receiver at the time when the feedback was sent), so the sender can now send *reserved* packets instead of *request*s. At point (3), the next observation interval ends and the estimate is corrected once more. Finally, the second feedback arrives at point (4), indicating the final rate of four packets in four time units. The reservation does not change after that.

4.2 Simulation results

The network configuration used for the simulation is shown in figure 11.[1] The grey paths mark flows we examine below.

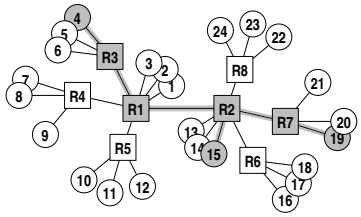

Fig. 11. Configuration of the simulated network.

There are eight routers (labeled **R1**...**R8**) and 24 hosts (labeled **1**...**24**). Each of the hosts **1**...**12** tries occasionally to send to any of the hosts **13**...**24**. Connection parameters are chosen such that the average number of concurrently active sources sending via the **R1**–**R2** link is approximately fifty. Flows

[1] The programs and configuration files used for the simulation are available on `http://lrcwww.epfl.ch/srp/`

have an on-off behaviour, where the on and off times are randomly chosen from the intervals $[5, 15]$ and $[0, 30]$ seconds, respectively. The bandwidth of a flow remains constant while the flow is active and is chosen randomly from the interval $[1, 200]$ packets per second.

All links in the network have a bandwidth of 4000 packets per second and a delay of 15 ms.[2] We allow up to 90% of the link capacity to be allocated to reserved traffic. The link between **R1** and **R2** is a bottleneck, which can only handle about 72% of the offered traffic. The delay objective D of each queue is 10 ms. The queue size per link is limited to 75 packets.

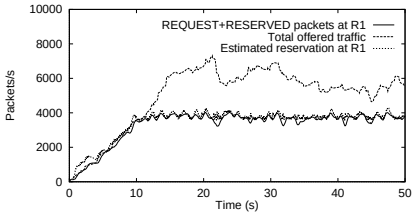

Fig. 12. Estimation and actual traffic at **R1** towards **R2**.

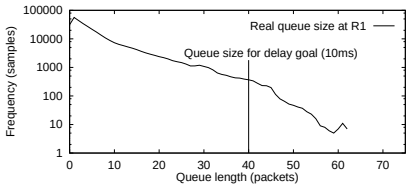

Fig. 13. Queue length at **R1** on the link towards **R2**.

Figure 12 shows the **R1**–**R2** link as seen from **R1**. We show the total offered rate, the estimated reservation ($\gamma\beta$) and the smoothed actual rates of *request* and *reserved* packets. Figure 13 shows the behaviour of the real queue. The system succeeds in limiting queuing delays to approximately the delay goal of 10 ms, which corresponds to a queue size of 40 packets. The queue limit of 75 packets is never reached.

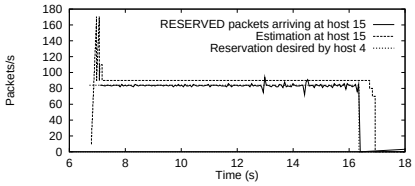

Fig. 14. End-to-end reservation from host **4** to host **15**.

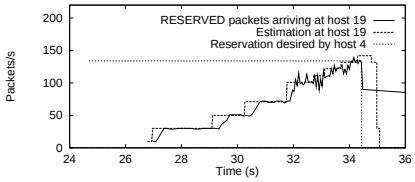

Fig. 15. End-to-end reservation from host **4** to host **19**.

Finally, we examine some end-to-end flows. Figure 14 shows a successful reservation of 84 packets per second from host **4** to **15**. The requested rate, the estimation at the destination, and the (smoothed) rate of *reserved* packets

[2] Small random variations were added to link bandwidth and delay to avoid the entire network from being perfectly synchronized.

are shown. Similarly, figure 15 shows the same data for a less successful reservation host **4** attempts later to **19**, at a time when the offered traffic is almost twice a high as the bandwidth available at the bottleneck.[3]

During the entire simulated interval of 50 seconds, 3'368 *request* packets and 164'723 *reserved* packets were sent from **R1** to **R2**. This is 83% of the bandwidth of that link.

5 Conclusion

We have proposed a new scalable resource reservation architecture for the Internet. Our architecture achieves scalability for a large number of concurrent flows by aggregating flows at each link. This aggregation is made possible by delegating certain traffic control decisions to end systems – an idea borrowed from TCP. Reservations are controlled with estimation algorithms, which predict future resource usage based on previously observed traffic. Furthermore, protocol processing is simplified by attaching the reservation control information directly to data packets.

We did not present a conclusive specification but rather described the general concepts, gave examples for implementations of core elements, including the design of estimator algorithms for sources, destinations and routers, and showed some illustrative simulation results. Further work will focus on completing the specification, on evaluating and improving the algorithms described in this paper, and finally on the implementation of a prototype.

References

1. Diot, Christophe; Huitema, Christian; Turletti, Thierry. *Multimedia Applications should be Adaptive*, `ftp://www.inria.fr/rodeo/diot/nca-hpcs.ps.gz`, HPCS'95 Workshop, August 1995.
2. RFC2205; Braden, Bob (Ed.); Zhang, Lixia; Berson, Steve; Herzog, Shai; Jamin, Sugih. *Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification*, IETF, September 1997.
3. RFC1819; Delgrossi, Luca; Berger, Louis. *ST2+ Protocol Specification*, IETF, August 1995.
4. Ferrari, Domenico; Banerjea, Anindo; Zhang, Hui. *Network Support for Multimedia - A Discussion of the Tenet Approach*, Computer Networks and ISDN Systems, vol. 26, pp. 1267-1280, 1994.
5. The ATM Forum, Technical Committee. *ATM User-Network Interface (UNI) Signalling Specification, Version 4.0*, `ftp://ftp.atmforum.com/pub/approved-specs/af-sig-0061.000.ps`, The ATM Forum, July 1996.
6. The ATM Forum, Technical Committee. *ATM Forum Traffic Management Specification, Version 4.0*, `ftp://ftp.atmforum.com/pub/approved-specs/af-tm-0056.000.ps`, April 1996.

[3] In this simulation, sources did not back off if a reservation progressed too slowly.

7. RFC2475; Blake, Steven; Black, David; Carlson, Mark; Davies, Elwyn; Wang, Zheng; Weiss, Walter. *An Architecture for Differentiated Services*, IETF, December 1998.
8. Bernet, Yoram; Yavatkar, Raj; Ford, Peter; Baker, Fred; Zhang, Lixia; Speer, Michael; Braden, Bob; Davie, Bruce. *Integrated Services Operation Over Diffserv Networks* (work in progress), Internet Draft `draft-ietf-issll-diffserv-rsvp-02.txt`, June, 1999.
9. RFC2474; Nichols, Kathleen; Blake, Steven; Baker, Fred; Black, David. *Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers*, IETF, December 1998.
10. Floyd, Sally; Mahdavi, Jamshid. *TCP-Friendly Unicast Rate-Based Flow Control*, `http://www.psc.edu/networking/papers/tcp_friendly.html`, Technical note, January 1997.
11. Almesberger, Werner; Ferrari, Tiziana; Le Boudec, Jean-Yves. *SRP: a Scalable Resource Reservation Protocol for the Internet*, Proceedings of IWQoS'98, pp. 107-116, IEEE, May 1998.
12. Ferrari, Tiziana. *QoS Support for Integrated Networks*, `http://www.cnaf.infn.it/~ferrari/tesidot.html`, Ph.D. thesis, November 1998.