# Aggregating Resource Reservations over Multiple Routing Domains

Olov Schelén and Stephen Pink
Computer Science and Electrical Engineering
Luleå University of Technology, SE – 971 87 Luleå, Sweden
{olov, steve}@cdt.luth.se

## Abstract

*We present an agent-based architecture for resource reservations. For each domain in the network there is an agent responsible for admission control. The architecture provides scalable per-link resource reservations in agents and low per-packet overhead in routers. The key ideas are the following. First, reservations from different sources to the same destination are aggregated as their paths merge toward the destination. Second, an agent in charge of resources at the final destination can generalize reservations for specific end points so that they are valid for any end point in the destination domain, thereby allowing more aggregation. Third, agents can do bulk reservations in advance with neighboring agents, thereby allowing aggregation over time. Fourth, agents are responsible for setting up police points at edge routers for checking commitments. Agents can minimize per-packet policing overhead in routers by varying the granularity of policing over time.*

## 1 Introduction

We are designing an architecture for resource reservations where clients make admission requests through agents [SP97b] [SP97a]. For each routing domain in the network there is an agent which knows the topology and static link resources in the domain (figure 1). The agent is an end-system that is configured for passively participating in a link state routing protocol (e.g., OSPF) where all routers have a complete topology map of the domain. Thus, an agent obtains the same topology map as the routers with little signaling overhead. Link state advertisements are sent from the nearest router to the agent. Agents retrieve link properties, such as static bandwidths, by querying routers seen in the topology map. For this, we use a network management protocol (e.g., SNMP). Queries are done at startup and when topology changes are detected by the routing protocol. Generally, routers need no extensions to allow agents to build a resource map through the management protocol.

Agents do parameter-based admission control over the static resources in their domain. There is no signaling of traffic dynamics between routers and agents. Admission requests contain the bandwidth to be reserved, a source and a destination address. This simple model allows us to focus on scaling properties for the reservation architecture rather than on complicated admission control. A more sophisticated model could involve token bucket specifications, delay specifications, bounds on packet sizes, etc.

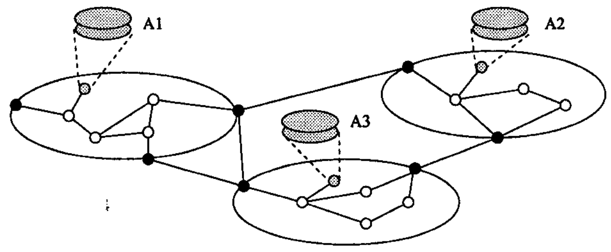

Figure 1: Reservation agents and their domains

Packets using reserved resources are *marked* by applications or edge routers. Core routers handle marked packets by classifying them into a small number of predefined service types. Similar ideas, known as *differentiated services*, have been proposed in [CF97] and [NJZ97]. In the differentiated services model, packets are marked to obtain either lower drop probability or higher strict priority. In this paper, we abstract from these details and use the term *priority packets* to denote marked packets, and the term *priority traffic* for all priority packets collectively. A key point of differentiated services and our agent architecture is to avoid non-scalable signaling state and expensive per packet processing in routers.

Independently of which packet scheduling model is used in routers, there must be an admission control architecture to make sure that the amount of priority traffic is low enough to receive a good quality of service. One issue is whether the admission control architecture should reserve resources over the links that are actually going to be used, or just limit the overall rights to send priority traffic. A scheme that does not consider exactly where traffic will go cannot give reliable commitments without very low utilization. Thus, it is likely that such a scheme would need to gamble and accept occasional service loss.

In this paper we focus on agent-based admission control where the objective is to administer resources exactly, i.e., admission control in an agent maintains information about reserved resources on each link in its domain. Agents do admission control without involving the routers. Our objective is to find out if such a scheme is scalable. The paper is organized as follows. Section 2 describes how reservations are aggregated and how agents can find out where traffic will enter a domain and which path it will take through the domain. Section 3 describes how commitments are enforced by agents setting up police points in edge routers checking that incoming traffic conforms. Section 4 provides a comparison with RSVP, and finally, section 5 discusses how the architecture can handle changing topologies.

In our architecture, reservation requests may be immediate and open-ended or made in advance by including starting time and duration for the reservations. In earlier work [SP97b], we have shown the benefits and costs of supporting a mix of immediate and advance reservations in terms of link utilization, rejection probabilities, preemption probabilities etc.

## 2 Reservation Model

An admission request is directed to any agent, e.g., an agent that manages a user's account. Each agent sets up reservations between any two points in the network by invoking other agents. Thus there is *no difference* between reservations for *sending* or *receiving* data, reservations for *nomadic computing*, or reservations paid by a *third-party*.

Each reservation request contains a bandwidth to be reserved, a source and a destination address. The source address determines the point where the reservation starts (i.e, the point where traffic using the reservation will enter) and the destination address determines the point where it ends. An agent receiving a request first considers whether the start point is in its domain. If the start point is *not* in its domain, it finds an agent closer to the start point and repeats the request with that agent. Technically speaking, it is possible to directly identify the agent that is responsible for the source domain, but in practice clients may prefer using a particular agent and agents may prefer repeating the request in several steps according to established accounting relations. In figure 2, it is shown how a request is issued by node Dx for reserving resources from Ax to Dx.

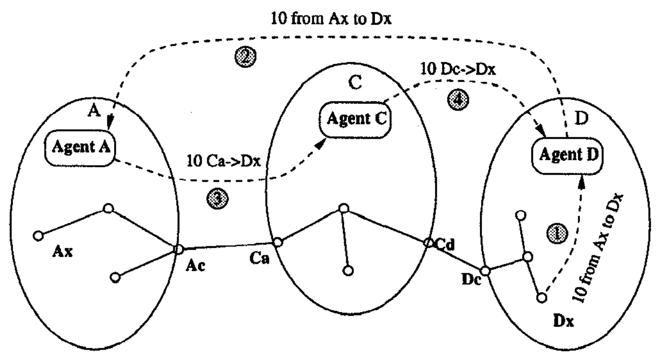

Figure 2: A receiver Dx requesting a reservation to receive data from Ax

If an agent finds that the start point given in the request is in its domain, admission control is performed on the links from the start point towards the destination, defined by the routing protocol. This is possible since all routers and the agent have the same topology map of the domain. Thus, an agent can perform admission control on the links from the start point to an edge router in its domain.

An agent cannot perform admission control beyond its domain. Therefore, if the destination is outside of the agent's domain it must request a reservation with the neighboring agent, giving the edge point where traffic will cross the borders as the source address of the reservation (figure 2). Without distinguishing between reservations to different destinations it would be impossible for agents to grant reservations to other domains. This is because there is no way to make sure that there are sufficient reservations along any path in other domains further downstream (unless we accept very low utilization for priority traffic).

If reservations were always done "on demand" there would be signaling between all neighboring agents along the path for each end-to-end admission request. To avoid this, agents make bulk reservations in advance so that admission requests become much fewer and more aggregated than if they were made only on demand. If an agent already has a sufficiently large reservation from its edge to a distant destination, it can grant further requests for that destination by just doing admission control over its local links and check that it has a sufficiently large reservation beyond its edge.

To grant requests spanning many routing domains, there must be a commitment from all agents along the path involved in the request. When an agent is making a reservation with a neighboring agent, the source address included in the request determines the point (router interface) where the traffic will cross the borders, i.e., enter the domain of the neighboring agent. The neighboring agent does not care where traffic came from originally. Thus, priority packets can use reserved resources as long as the destination address matches the reservation. This allows agents to aggregate traffic with different sources into one single reservation made with a neighboring agent.

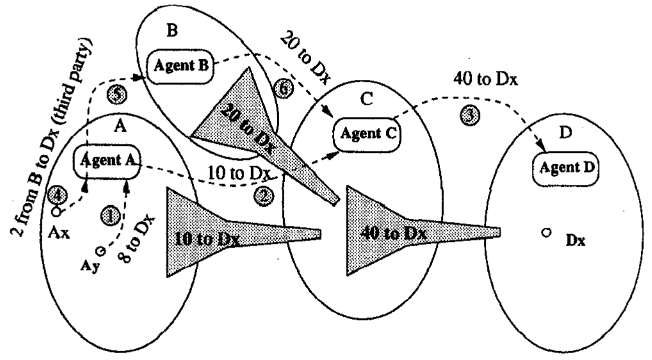

Figure 3: Funnels and aggregate reservations for one destination

An end-to-end resource reservation can be seen as a number of consecutive *funnels* (figure 3). Priority packets poured into any of these consecutive funnels use reserved resources all the way to the destination. Agents may aggregate several requests with different source addresses into the same funnel, as long as they all specify the same destination. Agent C (in fig. 3) may set up a funnel starting at its edge by pre-reserving resources with neighboring agent D to a distant destination and then aggregate requests for resources between domains C and D into the established funnel. This advance bulk reservation implies that downstream agents only keep state for the aggregate. The agent that decides to aggregate into a bulk reservation keeps information about its individual commitments constituting the aggregate. Aggregation by merging reservations towards each destination can be done with full control over the resources.

Requests that specify different destinations cannot easily be aggregated into a funnel in the general case. If we allow funnels to split out in different directions further downstream, upstream agents must keep information about which destinations are involved and how resources are divided between them. This means that the aggregate no longer can be seen as one unit. However, if a set of destinations can be identified by one common identifier, and there are sufficient resources available in each branch to service the whole aggregate, then split outs can be scalable and exact.

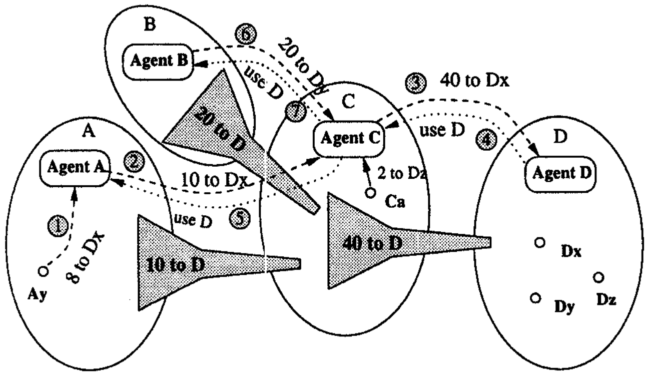

Figure 4: Funnels and generalized destinations

With this argument, we allow aggregation to be increased by *generalizing* a reservation to be valid for all destinations within the same destination domain. This means that a funnel provides resources to all endpoints within the destination domain, i.e., resources in the destination domain must be sufficient enough to ensure that priority packets get expected service independently of how they are routed within that domain. It is only the agent responsible for the destination domain that can judge if resources are sufficient for allowing generalization. Consider the example in figure 4. Agent D can judge if the resources in D'are sufficient to let incoming funnels be valid for the whole domain. Thus, it must be agent D that decides when a reservation request for a particular destination node within D can be replaced with a reservation for any node in domain D.

In IP networks an *address prefix* is often used to represent a domain (individual nodes within the domain are distinguished by the remaining suffix). An address prefix is represented by an IP address and an associated mask telling how many bits are part of that prefix. An agent generalizes a request by replying with an *address prefix* to be used as destination address for the reservation instead of the original address included in the request (figure 4). The aggregation that results from generalizing a reservation is also called *prefix aggregation.*

Clients or agents that have been granted a generalized reservation can use it for any destination matching the prefix of the reservation. Agents providing generalized reservations must handle the committed bandwidth of priority traffic to any destination in its domain. A conservative agent would therefore generalize reservations up to an aggregated value equal to the narrowest link. In practice we believe that this can be quite useful for well provisioned local area networks.

To sum up this section we stress that aggregation is happening in the agents as routers are not dealing with admission control at all. The motives for aggregation may be smaller when reservation state is stored in agents with plenty of memory and disc. However, a per-flow reservation model would have scaling problems both in terms of state and processing cost in agents. Therefore it is important to show that agents can provide exact resource reservations with nice scaling properties.

## 3 Policing and classification

Agents establish police points in edge routers of their domains to protect against sources and neighboring agents sending more priority traffic than committed. We call such traffic *excessive.* An ingress police point is established by looking at the source address in a reservation. This address identifies the edge router where traffic using the reservation will enter the domain. An agent may also establish an egress police point. Given the ingress point and the destination address, the routing protocol can find the point where traffic is leaving the domain.

In our architecture the number of commitments is kept low through aggregation. Still, the number may be quite large as we distinguish between reservations for different destinations (or prefixes). Policing is demanding as it involves per-packet processing, i.e., each priority packet must first be classified against all commitments before measuring bandwidth. Therefore, we have support for reducing the classification overhead. To obtain this reduction without loosing control, we accept more signaling between policing routers and their agent, as well as some delay until a misbehaving source is found.

The method is to check only aggregate bandwidth of priority traffic. When excessive priority traffic is found, this is reported to the agent (the agent sets up a trap with the policing router by using a management protocol). The agent can go through its commitments associated with the actual edge point and set up close policing by matching different destinations one-by-one, or randomly, in the router. The key idea is that the agent has knowledge about any policing that could be done, but saves per-packet policing overhead in the routers by not checking all commitments at once. The strategy depends on whether processing power or link bandwidth is the bottleneck in the router.

A police point judges whether each priority packet is excessive or not. Agents may set up actions in police points to degrade service for excessive packets, e.g., they can be either re-marked or dropped. The appropriate action depends on the policy associated with the service commitment that was given. In addition, agents should report excessive traffic back to upstream agents. There are two reasons for this. First, the agent detecting excessive traffic cannot police individual flows within in a commitment. Upstream agents have the details of the aggregate and can make sure that only violating parts of the aggregate will have service degradation. Second, it is a waste of network resources to serve excessive priority traffic and then degrade it, or even drop it, further downstream. As police points find out where excessive traffic is coming from, the report proceeds between agents towards the agent responsible for the source. There, a final police point should be established to regulate the excessive traffic.

## 4 Comparison with RSVP

In RSVP [ZDE+93], QoS setup is managed by end points and routers. Reservation state occurs for each flow in each router along the path. A flow is defined by a pair of sender and destination addresses (and ports), resulting in large numbers of fine granular reservations to manage by routers. Also per-packet processing is expensive as every packet must be classified against the total number of reservations. In RSVP, it is difficult to introduce aggregation as the model relies on per-flow semantics. To add aggregation, some way of identifying aggregates would be necessary. Transparent aggregation along a path would require labeling aggregates at one RSVP router, making sure that downstream routers know about the labels, and finally splitting up the aggregate at some hop. Overhead in intermediate routers is saved, but routers constituting aggregates would get even more per-packet overhead than before. Aggregation with RSVP can be difficult to support over several domains owned by different providers.

Having per-flow reservation state in the routers, as in RSVP, is not advisable for unicast reservations. Even if aggregation is supported over some parts of the path, there would generally be an explosion in state and packet processing overhead compared to ordinary best-effort routing. For multicast, the overhead associated with RSVP can probably be justified. Already by using best-effort multicast, a user has decided to trade off bandwidth savings for per-flow state in the routing table. The extra state for RSVP should only add some overhead. One should, however, make sure that multicast and RSVP are used only when resulting multicast trees are sufficiently dense (branched) to motivate the control state.

Our agent architecture is not currently tailored for multicast reservations as more research is required to develop a solution that provides sufficient benefits over RSVP. Therefore, the agent architecture can be used for unicast reservations, while RSVP can be used for multicast reservations.

## 5 Changing topologies and resources

In the Internet, agents must handle occasional topology changes, as well as pathological changes such as route-flaps [Pax96]. Agents listen to the routing protocol and detect routing changes on average as fast as any router in the domain, depending on when the routing protocol converges. When changes occur, agents can schedule resources along new paths. If it is found that there are insufficient static resources along the new path, agents can perform temporary actions by setting up police points to re-mark or drop traffic from some commitments so that other commitments can be met without quality loss. Once an agent has decided to degrade a commitment, the method for handling the resulting excessive traffic is the same as described in section 3. However, in this case traffic is considered excessive because of network failure (not client failure) and therefore clients could be given some kind of refund.

## 6 Conclusion

The agent architecture can provide resource reservations in a global network for sending data from any point to any other point, independently of where the reservation request originated. The key ideas of our architecture are *aggregation* (merging towards destinations), *generalization* of destination domains, *advance reservations* (bulk), and minimal *flexible policing*.

The architecture separates functionality between packet forwarding in routers and QoS negotiation in agents. For enforcement, agents must set up police points in some routers. Thus, one architectural requirement is to standardize an interface to allow trusted agents to setup police points.

## References

[CAH96] A. Campbell, C. Aurrecoechea, and L. Hauw. A review of qos architectures. In *Proceedings of 4th IFIP International Workshop on Quality of Service (IWQoS'96)*, Paris, France, March 1996.

[CF97] David D. Clark and Wenjia Fang. Explicit allocation of best effort packet delivery service. Technical report, MIT Lab for Computer Science, Boston, Massachusetts, November 1997.

[NJZ97] Kathie Nichols, Van Jacobson, and Lixia Zhang. A two-bit differentiated services architecture for the internet. Internet draft, Bay Networks, LBNL and UCLA, November 1997.

[Pax96] Vern Paxon. End-to-end routing behavior in the internet. In *Proceedings of SIGCOMM*, pages 25–38, Palo Alto, California, August 1996. ACM.

[SP97a] O. Schelén and S. Pink. An agent-based architecture for advance reservations. In *IEEE 22nd Annual Conference on Computer Networks (LCN'97)*, Minneapolis, Minnesota, November 1997.

[SP97b] O. Schelén and S. Pink. Sharing resources through advance reservation agents. In *Proceedings of IFIP Fifth International Workshop on Quality of Service (IWQoS'97)*, New York, May 1997.

[ZDE+93] Lixia Zhang, Stephen Deering, Deborah Estrin, Scott Shenker, and Daniel Zappala. RSVP: a new resource ReSerVation protocol. *IEEE Network Magazine*, 7(5):8–18, September 1993.