Resource Reservation Agents in the Internet

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 architecture where clients can make resource reservations through agents. For each domain in the network, there is an agent responsible for admission control. The architecture provides scalable resource reservations for uni-directional virtual leased lines. Reservations from different sources to the same destination are aggregated as their paths merge toward the destination. When there are sufficient resources in a destination domain, prefix aggregation is performed. Thus, all reservations for a given destination domain are aggregated as their paths merge. Agents are responsible for setting up police points at edges for checking commitments.

1 Introduction

We are designing a resource reservation architecture where clients make admission requests through *agents* [3, 4, 5]. For each routing domain in the network there is an agent responsible for admission control. Each agent knows the topology and static link resources in its domain (figure1). The agent is an end-system that is configured for passively participating in a link-state routing protocol (e.g., OSPF) where each participating router has an identical topological database over the domain. Thus, an agent obtains a copy of the topological database with little signaling overhead, i.e., 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 topological database. 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. Routers need no extensions to allow agents to build a resource map through the management protocol. Agents use the resource map to perform admission control in their domain.

Admission requests contain the bandwidth to be reserved, a source and a destination address. For each request, agents can find where traffic will enter a domain and therefore agents can setup police points in edge routers checking that incoming traffic conforms. Packets using reserved resources are *marked* by applications or edge routers and checked by police points when entering a domain. Similar ideas, known as *differentiated services*, have been proposed in [1, 2].

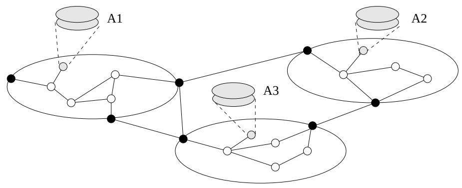

Figure 1: Reservation agents and their domains

In the differentiated services model, packets are marked to obtain either low drop probability or high scheduling 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 service and our agent architecture is to avoid non-scalable signaling state and expensive per packet processing in routers. Therefore resource reservations are handled by agents (end systems), and packet classification in routers involves only a small number of fixed service classes.

Independently of which packet scheduling model is used, there must be an admission control architecture to make sure that the amount of priority traffic is low enough to receive 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 doesn't 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 work 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. Our objective is to find out if such a scheme can scale. The architecture offers a service comparable to unidirectional *virtual leased lines*. In this model, requests may be immediate and open-ended or made in advance by including starting time and duration for the reservations. In [4] we show the implications of supporting a mix of immediate and advance reservations.

2 Reservation Model

An admission request can be 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 starting point is in its domain. If the starting point is *not* in its domain, it finds an agent closer to the starting 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. Also, agents may prefer to deal with only a few adjacent agents. This means that a request is repeated in several steps. In figure 2, it is shown how a request is issued by node Dx for reserving resources from Ax to Dx.

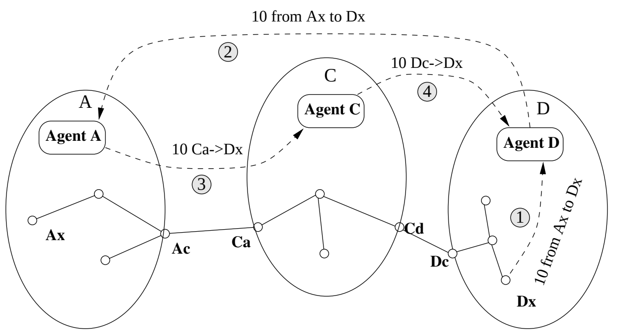

Figure 2: A receiver Dx requesting a reservation to receive data from Ax

If an agent finds that the starting point given in the request is in its domain, admission control is performed on the links from the starting point towards the destination, defined by the routing protocol. This is possible since all routers and the agent in a domain have the same topological database. Thus, an agent can perform admission control on the links from the starting 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).

3 Aggregation

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. Therefore, our architecture supports several ways of aggregating reservation state in agents.

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.

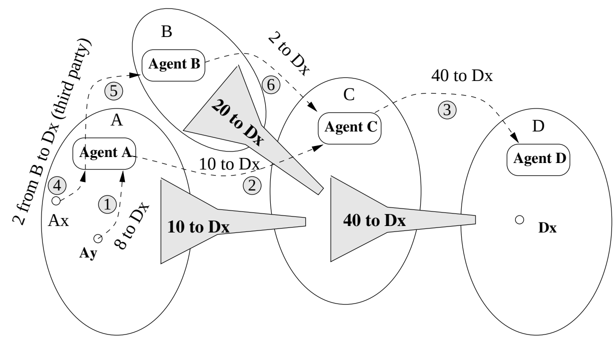

Figure 3: Funnels and aggregate reservations for one destination

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 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. The funnels form a *sink-tree* with the root towards the destination. At each merging point, the reserved bandwidth on the outgoing link will be equal to, or larger than, the sum of the reserved bandwidth on the incoming links.

Requests that specify different destinations can not easily be aggregated into a funnel in the general case. If we allow funnels to split out in different directions, upstream agents must keep information about which destinations are involved and how resources are divided between them. This means that the aggregate cannot be seen as one unit. However, a reservation can cover different destinations if resources for the whole aggregate are reserved to any of the involved destinations. In addition, there should be a scalable way of representing the set of destination addresses. Both of these requirements are often met for well provisioned destination domains in the Internet.

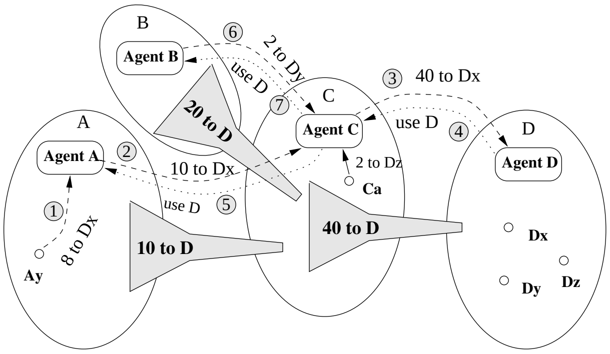

Figure 4: Funnels and prefix aggregation

Therefore, we allow aggregation to be increased by *generalizing* a reservation to be valid for all destinations within the same destination 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, agent D decides when a reservation request for a particular destination node within D can be replaced with a reservation for any node in domain D (or a subnet within 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 that are part of the prefix. An agent generalizes a request by replying with an *address prefix* to be used as destination address for the reservation (figure 4). The aggregation that results from generalizing a reservation is therefore called *prefix aggregation*.

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

To sum up, aggregation is happening in the agents as routers are not dealing with admission control at all. The motivation for aggregation may be less when reservation state is stored in agents with plenty of memory and disc. However, a per-flow reservation model has inherent 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 good scaling properties. Since agents have immediate access to the routing database, they can also handle changing routes and topologies with an outcome as favorable as possible.

4 Policing and classification

The source address in an admission request is used by agents to set up an ingress police point, i.e., in the router closest to the source. Egress policing can be setup by using the destination address, i.e., the routing protocol can find the point where traffic is leaving the domain. By policing at the edges, a provider can check that the rate of priority traffic stays within commitments. At each edge point, there may be several commitments to different destinations. Policing of individual commitments would therefore involve classifying priority packets by their destination addresses before measuring used bandwidth. Although the number of commitments are kept low through aggregation, policing is demanding as it involves per-packet processing compared to reservation setup which involves only per-commitment signaling. At the edge routers, close to the original sources of data, per-commitment policing may be affordable as there are generally few commitments and a low aggregate data rate. If all agents police correctly at the original sources, and we have an architecture that reserves resources along exact routes end-to-end, there is little need for policing in the backbone.

However, a model relying on policing at original sources only, would be volatile as it is based on global trust. Therefore, we suggest adding a model for scalable policing at edges of backbone domains. The method is to check a dynamic subset of the commitments. 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. When violating traffic is found, it is reported to the agent (the agent sets up policing and a trap with the router by using a management protocol). When an agent gets a report from a police point about a violated commitment, the agent should report it back to the upstream agent. The upstream agent has knowledge about the constituent parts that were aggregated into the violated commitment. By performing the same method at each agent, the police point for catching the violating traffic will gradually move back towards the sender (as far as there are cooperating agents), where it should have been regulated in the first place. For managing policing from agents, one architectural requirement is to standardize an interface to allow trusted agents to set up police points in their routers.

5 Interoperation with RSVP

In RSVP [6], reservations are made for individual data flows (i.e., application data streams). In that model, per-flow reservations are maintained in the routers. This results in scaling problems due to reservation state and packet classification overhead in routers. This model inherently relies on per-flow semantics and therefore it is hard to perform aggregation across different administrative domains. This has led some of the RSVP designers and others to advise keeping RSVP out of the backbone.

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 for RSVP can probably be motivated. Already with best-effort multicast, the network trades off bandwidth savings for per-flow state in the routing table. The extra state for RSVP would probably not add too much overhead. One should, however, make sure that multicast and RSVP are used only when resulting multicast trees are sufficiently dense (branched) to justify the control state. Therefore, the agent architecture can be used for unicast reservations, while RSVP is used for multicast reservations.

6 Conclusion

The agent architecture can provide resource reservations in a global network for offering virtual leased lines from any point to any point (i.e., independently of where reservation requests originate). The key ideas of our architecture are *aggregation* as paths merge towards destinations, *prefix aggregation* over destination domains, *advance reservations*, 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 their edge routers.

References

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

[2] K. Nichols, V. Jacobson, and L. Zhang. A two-bit differentiated services architecture for the internet. Internet draft, Bay Networks, LBNL and UCLA, November 1997.

[3] 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.

[4] 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.

[5] O. Schelén and S. Pink. Aggregating resource reservations over multiple routin domains. In *Proceedings of IFIP Sixth International Workshop on Quality of Se rvice (IWQoS'98)*, Napa Valley, California, May 1998.

[6] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala. RSVP: a new resource ReSerVation protocol. *IEEE Network Magazine*, 7(5):8–18, September 1993.