Quality of Service Agents

Olov Schelén*
Internet Research Institute
HP Laboratories Bristol
HPL-IRI-98-003
September, 1998

agents,
quality of service,
RSVP,
integrated services,
resource reservations

We describe some implications of providing quality of service by using Integrated Services and RSVP. We then present an architecture where clients can make resource reservations in the network through agents. For each domain in the network there is an agent responsible for immediate and advance admission control. Reservations from different sources to the same destination are aggregated as their paths merge toward the destination. In addition, reservations can be prefix aggregated so that a reservation can be used for many end points in a destination domain. Agents are responsible for setting up police points at edge routers for checking granted commitments.

*Department of Computer Science, Lulea University of Technology, Lulea, Sweden

1 Introduction

A substantial amount of work has been done in the IETF on developing router support and protocols for setting up quality of service in the Internet. Originally, the two key groups where the Integrated Services group and the RSVP group. The Integrated services group has defined router support for the transport of audio, video, real-time, and classical data traffic within a single network infrastructure. The router support includes admission control, packet classification and packet scheduling.

The RSVP group has developed a protocol for carrying users admission requests for admission control in RSVP/Intserv capable routers along the path of a data flow. RSVP is multicast capable and receiver oriented, i.e., it is the receivers that make resource reservations for obtaining better quality of service. RSVP together with Integrated services provide fine service granularity to end users by setting up service commitments on a per-flow basis.

Recently it has been identified that there are scaling problems with the RSVP/Intserv model. This is because the amount of state may grow too large in backbone routers where there is a large number of flows. Furthermore, per-packet classification cost depends on the number of reservations in a router. This is because every packet handled by a router must be classified against all reservations to find out which service it should be given.

The current focus in the IETF is on defining a more light-weight service model. For this purpose, the differentiated services group has been formed. The group is focused on scalable router support for service discrimination. The idea is that routers have a small set of pre-configured well-known service classes. Packets contain a field in the IP-header to define which service class they belong to. The service can be compared with the well-known postal service, where there are only a few service classes, e.g., express, budget, etc. Differentiated services offer a more scalable and less fine granular service model than integrated services. For differentiated services to work, there must be police points in the network to ensure that the amount of traffic is low enough in the better service classes. If there would be too much traffic in a service class, the service quality would drop below the expected level. Police points distinguish traffic that is allowed to use reserved resources from the traffic that is not allowed to use it. The work in differentiated services includes defining a field in the IP header to identify which service class packets belong to, some well-known service classes, and policing mechanisms in edge routers.

Differentiated services relies on external support for admission control and accounting. Also, external support is needed to set up appropriate police points. However, defining the admission control architecture is not within the framework of differentiated services. Neither is the mechanisms for setting up police points at appropriate places. Therefore, we have performed work on designing such an admission control architecture. The ar-

chitecture is based on agents, located in end-systems (servers), which are responsible for handling resource reservations. For each domain in the network there is an agent responsible for admission control. The offered service corresponds to unidirectional virtual leased lines. In this architecture aggregation is done to keep down the reservation state in agents. Reservations from different sources to the same destination domain are aggregated as their paths merge toward the destination.

1.1 Outline

This paper starts with an overview of Integrated services (section 2) and RSVP (section 3). In section 4, we explain the scaling properties of RSVP/Intserv and explain within which constraints the model is viable. In section 5, we describe our architecture. In section 6, we explain the relevance of these results to Hewlett Packard. In section 7, we list the deliverables that have been given to HP. This has been in the form of research papers and presentations given at different HP sites.

2 Integrated Services

The Integrated services group in the IETF has defined router support for providing quality-of-service in the Internet. The objective is to provide service for audio, video, real-time, and classical data traffic within a single network infrastructure. The router support includes admission control, packet classification, packet scheduling, and router validation schemes. Each router manages resources on their outgoing links only.

There are three service classes defined in the routers: guaranteed service [2], controlled-load service [7], and the traditional best-effort service. Both guaranteed service and controlled load service are admission controlled. Guaranteed service provides a guaranteed rate (bandwidth) that can be used to also compute a guaranteed delay-bound. Controlled load service is comparable to best-effort service when the network is unloaded. The results have been taken to proposed standard.

Admission requests involve a traffic specification in the form of a token bucket specification. To obtain better service quality end-to-end, admission should be requested at each hop along the path. This can be done through a management protocol, or by a specific reservation setup protocol.

3 RSVP

The Resource ReSerVation Protocol (RSVP) was jointly developed at the Information Sciences Institute of the University of California (ISI) and Xerox Corp.'s Palo Alto Research Center (PARC). Development is now carried on in the RSVP working group in the IETF.

RSVP is designed for supporting integrated services in the Internet. It is used to request QoS for specific data flows (i.e., application data streams). A data flow is defined by the sender and destination addresses and optionally also by the TCP/UDP ports.

RSVP carries admission requests to the admission control modules in each RSVP capable router along the path of a data flow. RSVP does not perform routing. Instead, it relies on the routing-mechanisms used for ordinary best-effort traffic. RSVP was designed to support both unicast and multicast reservations. To work well with multicast, reservations are unidirectional and initiated by individual receivers. This is because receivers in a multicast group may have different quality demands. In cases where receives do not send data themselves, uni-directional reservations use resources efficiently. Bi-directional (or multi-directional) reservations can be set up by making separate reservations in each direction. Receiver initiated reservations are scalable for senders as they have no overhead depending on the number of receivers involved.

To allow receivers to reserve, senders transmit path messages containing traffic specifications. Path messages set up state in all RSVP capable routers along the path. This state contains a traffic specification and the address of the previous RSVP hop. Receivers receive path messages and can use the information in those messages for sending reservation (resv) messages upstream toward senders. The path state in RSVP routers is used to route the reservation requests hop-by-hop back towards the senders.

RSVP relies on *soft state*. This means that reservations must be periodically refreshed. This is done by senders periodically sending path messages and receivers periodically reissuing reservation messages. Reservations that are not refreshed are automatically deleted after a certain period of time. The refresh period is variable, but is in current implementations typically around 30 seconds. The timeout period should be sufficiently large so that reservations stay in place even if some refresh messages are lost. Soft-state means that reservations that are not needed, e.g., after a route change or because a receiver machine crashed, will be automatically removed.

Since RSVP must adapt to route changes, Path/Resv messages are used to update the corresponding path and reservation states. For example, if there is a route change, then a Path message will inform the new nodes of the path change. A corresponding Resv message will be sent which will set the reservation states within the new nodes. The nodes which were in the old path and are no longer in the new path will time-out and

delete their path states using tear messages. Thus, refresh messages which update state information must be sent immediately when route changes are detected.

One reason why RSVP is not widely used is that there is still no support for charging and accounting. This is vital to support any kind of differentiated services. Current work on policy servers for RSVP aims at solving this problem. It is, however, still an open question how well this will work.

4 Scaling problems with RSVP

RSVP and Intserv have some inherent scaling problems. The number of RSVP control messages processed by each router is proportional to the number of QoS flows going through the router. Also, there is reservation state stored for each QoS flow. Thus, the amount of state and the processing overhead for control messages both scale linearly with the number of flows. This puts a heavy load on backbone routers with large interfaces. The primary function of routers is packet forwarding. Managing state information and performing additional lookups degrades router performance.

Unicast routing for best-effort service has good scaling properties as it involves no per-flow state. With RSVP, per-flow state is added. Compared to ordinary routing states, a reservation state is relatively short-lived and thus frequently changed. Due to these problems, one can expect that large backbone routers may have problems in keeping per-flow state. Some form of aggregation will be necessary.

Enforcement of reservations involve scaling problems as well. All incoming packets are classified to determine which QoS the flow should get. In current implementations of RSVP, a packet is classified not only according to fields in its network layer header but also according to transport layer headers, making the classification more expensive. The processing time for determining the QoS is proportional to the number of packets traversing the router. Today, even without reservation state, routers are considered to be bottlenecks in the Internet. From that point of view, Integrated Services implies undesirable overhead.

Although RSVP has scaling problems, it may work well in intranets where there are few large routers and few flows. It may also work well in the global network if the overall usage is controlled, e.g., by pricing. Most important, RSVP may work well for multicast in a global network. This is because multicast routing in itself is stateful, i.e., there is state per flow (multicast group) in the multicast routing table. RSVP may not add much state in the multicast case.

The problems with RSVP scalability are obvious when providing resource reservations

for unicast in the global Internet. Internet telephony is mostly unicast and involves a large number of small flows. With differentiated services it would be possible to provide a service class that is used for the aggregate of IP telephony traffic.

5 Our agent architecture

We are designing a resource reservation architecture where clients make admission requests through *agents* [4, 5, 6]. 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.

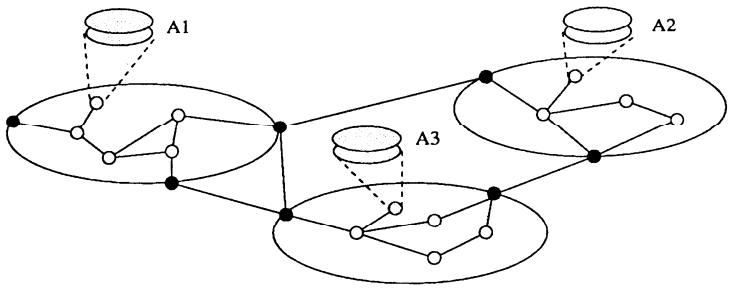

Figure 1: Reservation agents and their domains

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, 3].

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 [5] we show the implications of supporting a mix of immediate and advance reservations.

5.1 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.

If an agent finds that the starting point given in the request is in its domain, admission

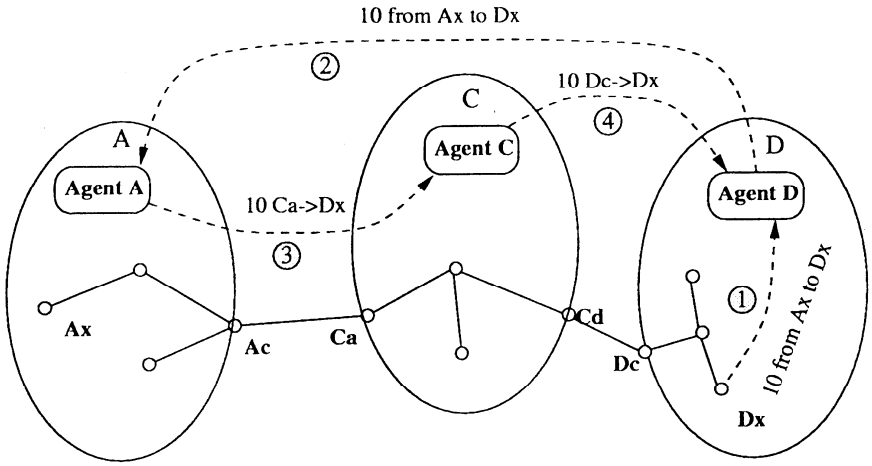

Figure 2: A receiver Dx requesting a reservation to receive data from Ax

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).

5.2 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.

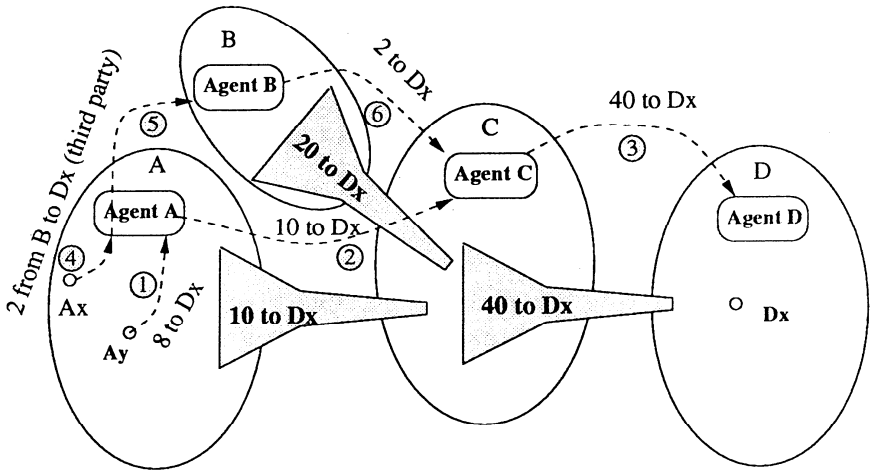

Figure 3: Funnels and aggregate reservations for one destination

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.

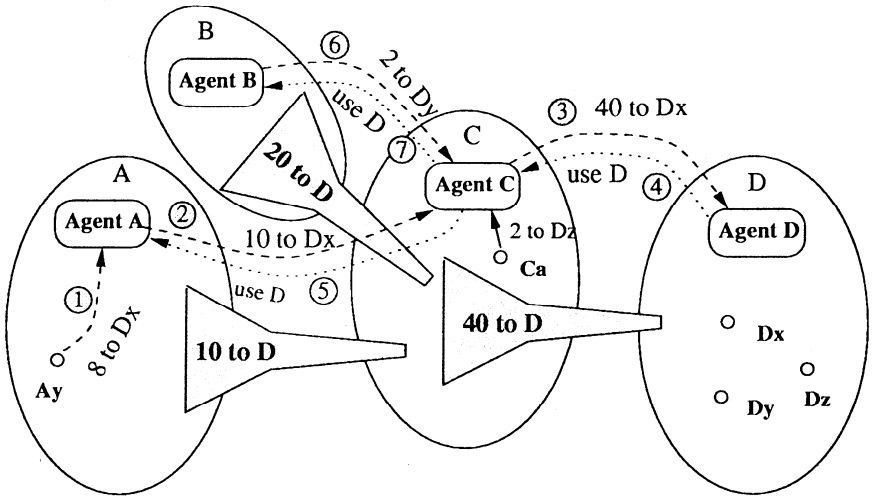

Figure 4: Funnels and prefix aggregation

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 admis-

sion 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.

5.3 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.

5.4 Interoperation with RSVP

In RSVP [8], 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.

5.5 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.

6 Relevance to HP

The project is relevant to HP as reservation agents are deployed in end-systems (servers) with plenty of CPU power and disc space. The architecture provides an opportunity for HP to impact an important field of networking, i.e., providing service quality and accounting in the Internet. There is a large demand for this, both from providers and clients. Providers will have more revenues from their networks and clients will have a

range of services to choose from to support different kinds of applications. Companies that can deliver products concerning reservation agents in domains supporting differentiated services is in a good position for the future.

7 Deliverables

Presentations:

- Oct 1997, HP Cupertino, "Resource Reservations in the Internet".
- Oct 1997, HP Palo Alto, "Resource Reservations in the Internet".
- Oct 1997, HP Roseville, "Resource Reservations in the Internet".
- Nov 1997, HP Bristol, "An Agent-based Architecture for Advance Reservations".
- Dec 1997, HP Grenoble, "An Agent-based Architecture for Advance Reservations".

Papers:

- Olov Schelén, Stephen Pink: Resource Reservation Agents in the Internet. Position paper. To appear in proceedings of 8th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'98), Cambridge, United Kingdom, July 1998.
- Olov Schelén, Stephen Pink: Aggregating Resource Reservations over Multiple Routing Domains. Short paper. Proceedings of IFIP Sixth International Workshop on Quality of Service (IWQoS'98), Napa Valley, California, May 1998.
- Olov Schelén, Stephen Pink: An Agent-based Architecture for Advance Reservations. Proceedings of IEEE 22nd Annual Conference on Computer Networks (LCN'97), Minneapolis, Nov 1997.

Slides for the presentations can be found at "http://www.cdt.luth.se/ olov/slides/". More papers can be found at "http://www.cdt.luth.se/ olov/publications/".