QoS-Adaptation by Software Agents in the Presence of Defective Reservation Mechanisms in the Internet

Hermann de Meer[1], Antonio Puliafito[2], Jan-Peter Richter[1], Orazio Tomarchio[2]

Dept. of Computer Science[1]
University of Hamburg
Vogt-Kölln-Str. 30, 22527 Hamburg - Germany
{demeer,richter}@informatik.uni-hamburg.de

Istituto di Informatica e Telecomunicazioni[2]
Università di Catania
Viale A. Doria 6, 95025 Catania - Italy
{ap,tomarchio}@iit.unict.it

Abstract

Originally, the Internet delivered best-effort service quality with respect to end-to-end delay. Recently, extensions such as RSVP have been proposed to provide guaranteed real-time services as well. Unfortunately, network resources, such as routers, do not yet fully support RSVP reservation protocols so that guarantees cannot truly be given. In this paper, we suggest to follow the paradigm of open programmable networks for a more complete QoS provisioning. Reservation gaps or tunnels are dynamically closed by means of a software-agent approach that is flexibly deployed for an application oriented QoS support. Agents are dynamically located to such tunnels in order to monitor the tunnels, to provide feed-back information in case of QoS violations, and to decide on possible compensating measures to be taken.

Keywords: QoS management, RSVP, Internet, intelligent and mobile agents, open programmable networks.

1 Introduction

Quality of service is an increasingly important issue for distributed multimedia systems. Generally, multimedia applications are very resource demanding so that efficient usage of resource is mandatory, often resulting in extensive resource sharing. To the contrary, hard or soft quality-of-service guarantees, like timely delivery of data, have to be given. Guarantees are usually implemented by some sort of resource reservation schemes, both in time and space. While efficiency and guarantee are equally important properties, they are driven by contradicting forces so that for a real system choices have to be made to what extend one or the other characteristic should be supported. Constraints are implied by the available resources and the need to customize services according to the requirements imposed by specific applications and users' preferences.

Assuming some fault model, the Internet was originally designed to provide a guaranteed reliable communication service. Routing algorithms are applied that are adaptive to node failures and a transport protocol is used that handles occasional losses and corruptions of data packets. Other qualities and guarantees were considered to be of less or of negligible importance. Time, in particular, was assumed to be abundantly available so that a best-effort service quality resulted with respect to end-to-end delay. Recently, extensions to the Internet such as the resource reservation protocol (RSVP) [11] have been proposed to provide guaranteed real-time services as well while simultaneously relaxing the requirements with respect to reliable communication. Unfortunately, network resources, such as routers, do not yet fully support RSVP reservation protocols so that guarantees cannot truly be given. RSVP is based on the concept of a software layer running on the top of IP, thereby using underlying routing protocols and providing admission and policing mechanisms on a per application basis. However, it is unlikely that RSVP will be supported by all routers in the near future [1]. Therefore, RSVP is designed to operate inside non-fully RSVP-based networks. If non-RSVP-capable nodes belong to the path of an established session for which the user wants a certain QoS, RSVP tries to reserve adequate resource on the nodes where this is possible and relies on a best-effort strategy in the remaining cases. Of course, in such cases no global guarantees can be given.

In this paper, we propose an approach how to deal with *tunnels* leading through a so-called *cloud* of non-RSVP-capable network entities and how to manage them with respect to application specific QoS requirements. Note that in our terminology a tunnel refers to those routers which are assigned to an application specific path through a cloud. The goal is to provide further enhanced QoS than currently possible with present Internet technology including reservation mechanisms [11]. We suggest to follow the paradigm of open programmable networks for a more complete QoS provisioning [5]. Such mechanisms can be implemented in a completely distributed fashion and are very well suited to the use of agent technology. The concept of *agent* (mobile, autonomous or intelligent) has recently gained much attention in several research areas, from artificial intelligence to distributed systems, from computer networks to software engineering [2, 3]. Software agents are already arranged for operating in a distributed manner and they permit very flexible communication and co-operation schemes, suggesting themselves as an attractive foundation for open programmable networks.

The remaining of the paper is organized as follows. Section 2 provides the technical prerequisites and introduces

[1] Supported by a grant from *Deutsche Akademische Austauschdienst (DAAD)*

[2] Supported by a grant from *Conferenza Permanente dei Rettori delle Università Italiane (CRUI)*

the tunnel problem in more detail. Section 3 shortly reviews our agent-based QoS reference architecture. Details on our solution approach based on tunnel agents are given in section 4. Finally, the paper is concluded in Section 5.

2 RSVP

Heterogeneous networking will be a reality throughout the next years. Even though new technologies like ATM inherently provide some means of guaranteed QoS provisioning, protocols from the TCP/IP world will continue to be used. This means that QoS provisioning has also to be performed in the Internet world. A very promising approach is based on RSVP [1, 11].

RSVP is a framework for resource reservation and QoS provisioning mechanisms based on an extension of functionality of classical IP routers, including the processing of RSVP messages as well as a more controlled operation of packet forwarding. While the RSVP protocol merely defines the exchange of control messages to build up and maintain a shared knowledge of reservation state, several QoS supporting *integrated services* are defined within that framework that complement the protocol with detailed definitions of router behaviour. Two such services have been defined: controlled-load service [10] and guaranteed service [8]. While the first form of QoS provision may be sufficient for self-adapting applications, more advanced reservations schemes are necessary if QoS guarantees are actually to be given. We apply our agent-based approach to solve a specific problem in the deployment of guaranteed service within the framework of RSVP.

2.1 The tunnel problem

Since the introduction of RSVP requires major changes to the operation of IP routers, not all routers within the internet will be capable of RSVP support in the near or mid future. In fact, the authors of [1] admit that RSVP support may never be ubiquitous, i.e., non-RSVP-capable routers will remain to be a reality. For the sake of brevity, we will refer to RSVP-capable routers and non-RSVP-capable routers as *RSVP routers* and *non-RSVP routers*, respectively, in the sequel. The RSVP protocol that handles the exchange of control messages simply ignores non-RSVP routers. RSVP messages are forwarded through a *cloud* of non-RSVP routers and reservation is merely performed outside the cloud. However, since non-RSVP routers may degrade the perceived QoS in an uncontrolled way, no end-to-end guarantee can be given. It is still believed that the application of RSVP on the remaining part of the path is beneficial, because the performance of the non-RSVP routers *may* be sufficient to fulfil the demanded level of QoS. Besides the ability to communicate control messages of the RSVP protocol through such a cloud of non-RSVP routers, no further action is taken within the RSVP framework to handle such a situation. Instead, the behaviour of the routers within the tunnel is simply ignored and the application is given an indication that the QoS values are to be understood as approximate.

For the sake of a clear terminology, we define the set of routers that comprise the sub-path within a cloud of non-RSVP routers to be a *tunnel*; a single router in the tunnel is referred to as a *tunnel router*. It is our goal to improve the end-to-end quality by monitoring the tunnel and providing feed-back for an enhancement of the reservation scheme. The overall procedure we discuss in the paper is based on the existence of distributed mechanisms that are able to implement cooperative monitoring techniques in order:

- to monitor the behavior of the system in terms of the performance level it provides;
- to use information on the status of the system for establishing a QoS negotiation phase and for proper distribution of application requirements on system resources;
- to interact with the reservation entities within the RSVP routers.

The basic idea of the approach consists in exploiting the interaction among properly defined software entities continuously running inside the system in order to implement QoS monitoring, reservation and adaptation functionalities.

3 Reference Architecture

The use of an agent-based infrastructure for the management of QoS is a good choice for implementing several adaptation strategies by which available resources need to be re-arranged in such a way that the user is still satisfied. Particularly interesting for the implementation of adaptation strategies is the exploitation of agents' cooperation abilities for optimization purposes. Although reasoning and decision are locally performed by each agent, the agents are coordinated by communicating certain events and internal status information to each other. As a result, each agent is influenced by and can influence the behavior of other agents.

According to the specific problem, the basic functionalities that a software agent may require can be summarized as follows:

- to monitor system status by combining and filtering of data to reduce message overhead between managed components and management applications;
- to cooperate with other agents by exchanging data and synchronization messages;
- to migrate from one node to another and execute remotely with a transparent use of the available hw/sw resources.

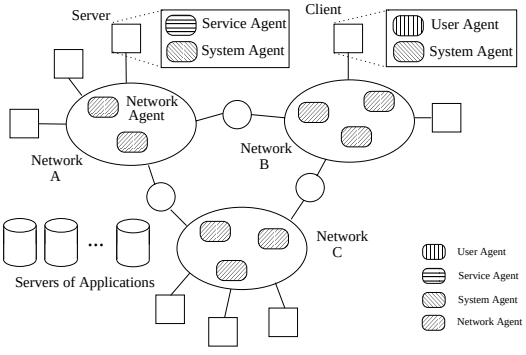

Figure 1: Reference Architecture

To implement agent retrievability and mobility, a computing platform has been developed [7] which, in order to provide a sort of homogeneity among different hardware and software platforms, adopts Java [4] as its underlying framework. A specialized version of such platform has been developed for the purpose of QoS management [6], which comprises specific nodes to collect QoS applications. We identify such nodes as *Servers of Applications* and assume that they store QoS applications to be eventually downloaded and remotely executed on demand.

The reference architecture we refer to is depicted in Fig. 1. Software agents are classified in the architecture according to the specific functionalities they carry out. A user on the client system interacts with the QoS management component by means of an appropriate *User Agent*. This agent is specific for each service required by the user, and enables him to specify and negotiate the desired QoS parameters. The *System Agent* provides all the information about the client's system state; relying on the services provided by the operating system of the host machine, it has the mechanisms to control the QoS on the client. On each server node a *Service Agent* is present which has to monitor whether the QoS parameters (for each service required by the clients) are respected. It keeps track of the users logged in and their occupied resources, in order to optimize the use of available resources. This agent also maintains the knowledge of the server's capabilities.

Control functionalities are implemented by *Network Agents*. Network agents have the following responsibilities: a) reservation of suitable network resources; b) monitoring of system parameters; c) initiation of corrective QoS adaptation activities if QoS violations are detected; d) interaction with the network management system in order to obtain additional resources and/or to optimize their use.

In the next section the reference architecture is specifically applied to the solution of the tunnel problem.

4 Tunnel Agents

In this Section we focus our attention on the behavior of *tunnel agents*. Their primary goal is the monitoring of tunnel properties in order to interact with the RSVP routers for assuring the end-to-end QoS to the user.

4.1 Tunnel setup

When an application on a receiving node makes a request for a reservation, RSVP carries the request through the network by visiting each node that belongs to the routing path between the receiving machine and the sender. On each intermediate node RSVP tries to reserve adequate resources to guarantee the desired QoS specified by the user. If there are enough resources on each router and if the admission test succeeds, the data stream starts flowing on the pre-established path with the negotiated QoS. However, some non-RSVP nodes, referred to as a tunnel, may be located along the route (Fig. 2).

In order to monitor the tunnel behavior and obtain useful performance indices for characterization of the delay experienced in crossing the tunnel, we adopt the agent approach described in the previous section. Two particular network agents, in the following referred to as the *tunnel agents*, are

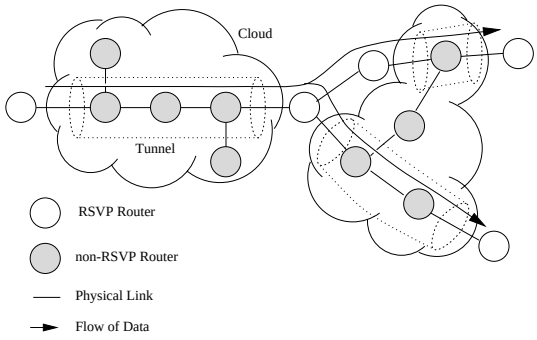

Figure 2: Tunnels

associated to each tunnel. More specifically, tunnel agents are located on the RSVP nodes at the edges of each tunnel, as shown in Fig.3. The RSVP nodes at the edges of a tunnel will be referred to as *edge routers* in the sequel. Note, that an RSVP router may be an edge router for two adjacent tunnels if it is surrounded by non-RSVP routers. In this case two instances of a tunnel agent are located on that router. We assume that each RSVP node is able to execute our agents. Furthermore, all nodes, including non-RSVP nodes, are assumed to support standard management protocols like SNMP [9] to provide some means for monitoring. In an extreme scenario, none of the routers supports RSVP and thus the tunnel spans over the entire path. In this case, the tunnel agents are located on the application hosts and provide the application with useful performance monitoring data to allow for adaptation of traffic volume and encoding.

The code of the agents is maintained in a generic node which plays the role of a Server of QoS applications. Once a tunnel is identified, the user agent located on the receiving application node sends a request to the nearest QoS application server to upload the tunnel agent code, which is then dynamically executed (for further details on these mechanisms see [7]). The main advantage of such approach consists in avoiding to install the code on each node: the location of a tunnel depends on dynamic factors like the existence and location of the session path as well as more static properties like the RSVP capability of the nodes. It may even happen that a node will never be at the edge of a tunnel. A tunnel agent installed on such a node would not be useful and would only consume node resources. The existence of simple mechanisms for code retrieval and downloading is thus a useful feature for our specific problem. Operations performed by tunnel agents will be described in the next two sections.

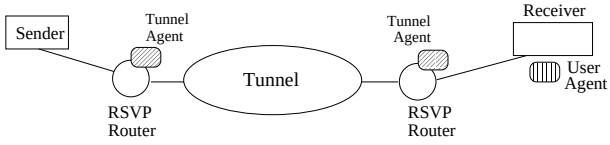

Figure 3: Tunnel Agents

The set-up procedure evolves according to the following

steps:

1. The receiving application becomes aware of the existence of non-RSVP routers along the path. This is part of the standard RSVP functionality. Subsequently, the application triggers its user agent to initiate the tunnel setup procedure.
2. As a second step, the exact location of the tunnel or tunnels must be detected. There are at least three different ways to achieve this goal.
   (a) By using SNMP, the user agent queries the RSVP routers for their respective RSVP neighbours and their direct IP connectivity. If an RSVP router has no direct IP connectivity to one of its RSVP neighbours, it must be an edge router. Following the chain of RSVP neighbours, the user agent eventually learns about the complete sets of RSVP routers and edge routers along the path. Since an SNMP request/reply involves only two UDP packets, it will be the best solution for small networks. However, this method has the disadvantage that it scales quite badly, so it may only be appropriate for short end-to-end paths.
   (b) A *discovery agent* is sent out from the host of the receiving application. This agent migrates from RSVP router to RSVP router and issues local SNMP requests to learn about the RSVP neighbourhood relation and the directly connected routers. Finally the agent returns the collected information back to the application host.
   This solution needs only linear time and is therefore more suitable for longer end-to-end paths.
   (c) A third possible implementation of the tunnel detection step could be an extension of the RSVP code that is installed on the RSVP routers. Since each RSVP router is aware of being an edge router or not, each edge router could send a special message to the application host. However this requires the extension of the functionality provided by the RSVP module on each router, which may not be easily achievable.

   A selection of a tunnel discovery method between the first and the second method can easily be done adaptively based on the number of RSVP hops.
3. An instance of a tunnel agent is uploaded to each edge router.
4. Each tunnel agent that is located on an upstream edge router starts a traceroute procedure to learn about the non-RSVP routers constituting its tunnel. This information is communicated to the corresponding downstream tunnel agents[1].
5. All tunnel agents monitor the tunnel behaviour and eventually interact with the RSVP protocol, as described in the following sections.

[1] A multicast tunnel may have more than one downstream edge.

4.2 Interaction with tunnels

An essential task of tunnel agents is to make decisions how network adaptation can be achieved. These decisions can only be made based on information concerning the structure and state of the tunnel. The topological structure of the tunnel, i.e., the set and connectivity of non-RSVP routers, is determined using a traceroute procedure and SNMP, as described in the last section. SNMP is also used to monitor basic performance measures of the tunnel routers like interface utilization, discard rates, or queue lengths by polling the respective MIB entries.

Additionally, the traffic that goes through the tunnel and the QoS it receives is constantly monitored by the set of agents at the edges of the tunnel using cooperative monitoring techniques. As an example, a pair of tunnel agents is able to measure tunnel-end-to-tunnel-end delay by sending time-stamped probe messages. Note that messages, which are time-stamped at the sender side of the entire path, cannot be used to measure tunnel-end-to-tunnel-end delay without additional message exchange between the tunnel agents.

4.3 Interaction with RSVP

The total end-to-end delay is the sum of all delays encountered along the transmission path. Therefore, a bound on end-to-end delays required by an application is considered as a QoS budget that can be arbitrarily spent by the involved intermediate nodes.

Since the RSVP guaranteed service is based on a fluid flow and leaky-bucket approach, the delay budget is distributed using a reservation of a minimum service rate R. Each RSVP router reserves resources sufficient to support rate R. By this, each local queueing delay is bounded in a way that the sum of all local queueing delays is bounded to a given value. For a detailed discussion see [8]. The rate R is calculated by the receiver in a way that the resulting end-to-end delay meets the QoS demands. However, without agent support this calculation would be done based on the assumption that all routers along the path were RSVP-capable even if the presence of non-RSVP routers was signaled. To the contrary, in our agent-based approach a higher rate $R' = R + \delta$ is used to set aside some additional delay budget that will not be distributed among the RSVP routers. The calculation of R' is done by the user agent in cooperation with the application and the state information provided by the tunnel agents. The additional delay budget which is characterized by the difference between the original rate R and the announced rate R' is subsequently also communicated to the tunnel agents. If more than one tunnel is present along the path between sender and receiver all tunnel agents agree on a distribution of the additional delay budget using a distributed inter-agent negotiation procedure. By monitoring the tunnel behaviour, the agents verify if the QoS budgets currently consumed by the tunnel matches the value that was assigned to that tunnel. In case of a mismatch, the tunnel agent initiates horizontal renegotiation. First, the tunnel agent re-negotiates the partitioning of the set-aside delay budget by re-iteration of the inter-agent negotiation procedure. If there is only one tunnel or the inter-agent re-negotiation was not successful, the QoS parameters in the RSVP routers along the path are re-

negotiated. This is done by communicating the additional need of delay budget to the user agent that re-calculates R' and informs the application to change its RSVP reservation messages accordingly. By re-assignment of budgets it is tried to compensate for the adverse effect of the tunnel routers. Of course, the opposite is also true: if the performance of the tunnel enhances as a result of traffic fluctuation, the horizontal re-negotiation is initiated to relax unnecessary stringent requirements on the RSVP routers. The phase of re-negotiation may fail due to lack of resources. In this case, an indication is sent to the application by the standard RSVP protocol mechanisms. In turn, the application may adapt to the lower level of QoS or may simply terminate.

4.4 Tunnel agent migration

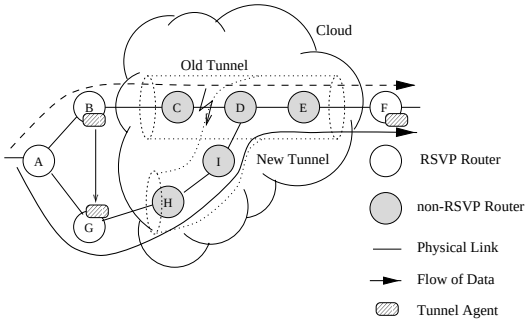

Figure 4: Agent migration after route change

Routing in the Internet is a dynmic process and a route may change during connection time. In this situation the ability of agents to migrate between routers can be exploited. Consider a situation as depicted in Fig.4. A stream of data flows from router A through B, C, D, E to F. When the connection between C and D breaks down, the usual routing algorithms of IP-based networks will find an alternative route, say via G, H and I. Also the mechanisms of the RSVP protocol that strongly rely on IP routing follow this route change and again try to establish reservation state along the new path. The tunnel agent located on B will notice that it is no longer at the edge of an active tunnel and initiates a relocation procedure. This procedure is carried out in cooperation with the agent on the other end of the tunnel or with the user agent on the application host, depending on whether the upstream or the downstream end of a tunnel has moved. By applying algorithms similar to the tunnel detection procedure, the new topology of the tunnel is discovered. In the case that only one edge of the tunnel has moved, the tunnel agent initiates its own migration to the new location of the tunnel edge. If RSVP routers are located along the new path between G and F, the new tunnel actually consists of a sequence of tunnels. In this case new tunnel agents must be started in addition to migration of one tunnel agent.

In the example, tunnel routers D and E both belong to the old route as well as to the new route and therefore comprise part of the new tunnel. By migrating the tunnel agent from router B to router G, much information about the old tunnel can be saved and re-used to manage the new tunnel. Especially topological information on the unmodified parts and to a certain extent also load information on those parts can be kept. Of course, a route change can also cause a tunnel to completely disappear. If the tunnel agents detect that they are now connected by a continuous chain of RSVP routers, they inform the other tunnel agents along the path – if there are any – as well as the user agents on the application hosts and terminate.

5 Conclusions

We have presented an approach based on intelligent software agents for complementing currently still defective reservation schemes such as RSVP in the Internet. More complete QoS provisioning can thus be achieved in a flexible and highly scalable way. We believe that future telecommunication networks will be increasingly open so that services like QoS-management functionalities can be deliberately added. A Java-based agent platform seems to be particularly promising in a heterogeneous environment, which distributed multimedia systems are most likely to face. Further studies are needed to investigate the efficiency of our implementation. In particular, experiences will be necessary to determine agent-control strategies to avoid oscillation effects and to provide limits for a timely response of the agents.

References

[1] R. Braden, L. Zhang, S. Berson, and S. Herzogand S. Jamin. Resource ReSerVation Protocol (RSVP) – ver. 1 Functional Specification. *RFC 2205*, September 1997.

[2] D.M. Chess. Mobile Agents: Are they a Good Idea. *IBM Technical Report RC19887*, October 1994.

[3] M.R. Genesereth and S.P. Ketchpel. Software Agents. *Communications of the ACM*, 37(7):48–53, July 1994.

[4] M.A. Hamilton. Java and the Shift to Net-Centric Computing. *IEEE Computer*, 29(8):31–39, August 1996.

[5] A. Lazar. Programming telecommunication networks. In *Proc. of the 5th Int. Workshop on Quality of Service (IWQoS97)*, pages 3–22, New York, May 1997.

[6] A. Puliafito, O. Tomarchio, and H. de Meer. An agent-based framework for QoS management. In *First World Congress on System Simulation WCSS'97: 4th ANMT on QoS*, Singapore, September 1997.

[7] A. Puliafito, O. Tomarchio, and L. Vita. A Java-based Distributed Network Management Architecture. *Third International Conference on Computer Science and Informatics, CS&I'97*, March 1997.

[8] S. Shenker, C. Patridge, and R. Guerin. Specification of Guaranteed Quality of Service. *RFC 2212*, September 1997.

[9] W. Stallings. *SNMP, SNMPv2 and CMIP. The Practical Guide to Network-Management Standards.* Addison Wesley, 1993.

[10] J. Wroclawski. Specification Controlled-Load Network Element Service. *RFC 2211*, September 1997.

[11] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala. RSVP: A New Resource Reservation Protocol. *IEEE Network*, 7(5):8–18, September 1993.