Invited Paper & Talk at
IFIP Fifth International Workshop on Quality of Service (IWQOS '97) – Building QoS into Distributed Systems –
May 21-23, 1997, Columbia University, New York, USA

Quality of Service: Where are We ?

R. Steinmetz[1,2], *L.C. Wolf*[1]

1
Industrial Process and System Communications
Dept. of Electrical Eng. & Information Technology
Darmstadt University of Technology
Merckstr. 25 • D-64283 Darmstadt • Germany

2
GMD IPSI
German National Research Center
for Information Technology
Dolivostr. 15 • D-64293 Darmstadt • Germany

{Ralf.Steinmetz, Lars.Wolf}@kom.th-darmstadt.de

Abstract

Over the last years, Quality of Service (QoS) has evolved as a research field to support new application types in general, very often in networked, computer systems. The most prominent of these applications are distributed multimedia applications which transmit and process audiovisual information streams. Due to their real-time nature, any correct processing and transport of these data must take timing aspects into account. The goal of QoS mechanisms is to ensure that the overall presentation of audiovisual data to the user respects these properties in an end-to-end mode.

In this paper we give an overview about the terms, issues and trends in the provisioning of QoS whereby we concentrate on principles and architectural aspects. We briefly describe the fundamental steps followed by systems offering QoS and review the IntServ work evolving in the Internet as an example for a prominent QoS model. Furthermore, past, actual, and future issues are discussed which we consider to be important for QoS architectures.

Keywords

Quality of Service, Resource Management, Integrated Services

1 INTRODUCTION

Traditional applications such as information processing, number crunching, text processing, etc. operate in a time-sharing environment without any hard time constraints. The system responds to a user interaction as soon as possible but lacks support for real-time data. New applications, especially multimedia applications which make use of continuous-media data such as audio and video, place other demands on distributed computer systems because they require time-dependent data processing. 'Correctness' in such a system is – in addition to the traditional, static meaning – determined by whether deadlines are met or not. Furthermore, the processing demands of digital audio and video data are typically large, i.e., although the available capacity is sufficient, it is not abundant to serve properly a continuous-media data presentation.

Quality of Service (QoS) mechanisms promise to ensure that these requirements are fulfilled. Most work on QoS is directed towards the support of multimedia applications. However, beneath multimedia applications, other applications, e.g., from the areas of distributed simulations and system control, require proper timing considerations as well. Yet, the QoS mechanisms developed for multimedia systems are

This work is supported in part by a grant of Volkswagen-Stiftung, D-30519 Hannover, Germany and partially founded by Deutsche Telekom AG, Technology Centre Darmstadt, D-64295 Darmstadt, Germany.

not appropriate for all of them; e.g., due to different base assumptions and sensitivity. Here, we concentrate on QoS mechanisms for multimedia systems.

QoS is defined as the set of parameters which defines the properties of media streams. The notion of QoS is different for the various system layers, e.g., the description of QoS at the application layer is usually at a higher level than that at the network layer of a communication system. The required QoS depends on various factors such as the used media (video, audio, etc.), the coding format used to encode the data, the application and the type of the application. For instance, the QoS of a video conference is different from that of a video retrieval application, since the dialogue-mode communication of a conference requires a short delay which is not as important for playback applications.

Applications negotiate the desired QoS with the affected components such as the network or with a general QoS management agent which performs the negotiation with the various components on behalf of the application. These negotiations usually take place at the start of the application, i.e., at the setup of the QoS.

Resource management systems that include mechanisms for data streams with guaranteed or statistical QoS have become a key issue in the support of multimedia applications (e.g., [22][11]). Those systems take care of the coordination of media streams, the interfacing between layers of protocol stacks as well as further mechanisms such as process, disk and bandwidth scheduling in order to enforce the appropriate data handling. Most of the involved mechanisms are developed for a completely error-free presentation of continuous-media data at the user interface.

In today's networked environments we still encounter many data paths via networks and communication protocols which are not capable of providing a guaranteed real-time service. In such set-ups it is important to decide which data item must be presented at the user interface and which data items may be discarded. The approaches for this are known as "scaling" and "filtering" of media data streams.

In this paper we give an overview about terms, issues and trends related to the support of QoS, whereby we concentrate on principles and architectural aspects. In the next section, the fundamental steps followed by systems offering QoS are described. Then we briefly review the IntServ work evolving in the Internet as an example for a QoS architecture. In Section 4 we discuss past, actual, and future issues which we consider important for QoS architectures. Finally we summarize the paper in Section 5. Despite the fact that QoS is an end-to-end aspect, we focus within this paper mostly on communication system aspects and their capabilities to support QoS because this is the most active area.

2 NOTION OF QUALITY OF SERVICE

Support for configurable, realizable and maintainable QoS is expected by numerous distributed applications. As such, it must necessarily be provided on an end-to-end basis from the source of data to the final perception of the data by the consuming user. This means that all hardware and software components involved in the overall tasks of the applications must provide appropriate methods and handle the data accordingly – from the local resources at the sender side via the transport system and networks, to the local resources at the receiving side. This applies to end-systems, servers, and networks as well as to system software and applications.

Most of the participating resources are shared among users and various processes. One approach would be to (over-) design them based on peak demands such that never any collisions between demands of different applications can occur. Then it would not be necessary to provide any resource management functionality. Yet, such a scenario would result in huge costs and low resource utilization and, hence, is typically not realizable. So if we have limited resources only, we may use filtering and scaling mechanisms which adapt the generated workload to the available resources by changing the characteristics of the transmitted data stream, e.g., lowering the frame rate of a video stream. Yet, these techniques cannot offer a reliable, constant QoS during the run-time of an application. We believe that typical distributed computer systems still are and will be so for a considerable amount of time in the “window of scarcity” illustrated by Anderson et al. in . Thus, to provide a constant QoS during the run-time of an application, resource reservation and scheduling techniques must be applied.

There are still different perspectives on QoS as observed by service providers and service users, whereby for the former efficiency of resource utilization is the focal point of QoS, while to the latter the completeness of parametrization of the service is most important. These different perspectives have led to different service models, which in the past did not allow for quantitative mappings between them. Hence a goal to be strived for must be an integral view of QoS for both, providers and users of a service. This can be achieved if both components are integrated into one system as for example when the operating system offers services to the applications like process scheduling, but is very difficult when multiple parties have to cooperate as in the case of a network services provider and an application program, which naturally pursue very different objectives.

2.1 QoS provisioning steps and components

In order to provide QoS by using resource reservation and scheduling, several steps must be performed in turn at each system and component participating in the end-to-end application . These steps can be divided into the QoS negotiation phase, consisting of (1) *QoS specification*, (2) *Capacity test and QoS calculation*, and (3) *Reservation*, and the data transmission phase, where the negotiated QoS is enforced by appropriate resource scheduling.

Overall, several resource management components interact to provide QoS assurance: Applications, QoS translators, admission control, resource scheduler. Additionally, further components are needed, for example, a resource monitor which measures the availability of resources and monitors whether the promised QoS is actually provided. Besides these local mechanisms at the end-systems and routers, resource reservation protocols are needed. They exchange and negotiate QoS specifications (accumulated in *FlowSpec*s) between the participating systems. These protocols perform no reservation of required resources themselves, but are only the vehicles to transfer information about resource requirements and to negotiate QoS values – the reservation itself is left to local resource management modules.

2.2 QoS classes and layers

Several classes of QoS are typically distinguished, the extreme on one side is a hard “deterministically guaranteed QoS” where the reservation is based on peak requirements and worst-case assumptions, the other is the "best-effort" approach where no reservation is made at all (and should therefore, at least with respect to QoS, better be called "no-effort"). Between these exist various forms of weaker QoS (statistical, predictive) based on average case and predicted assumptions. The hard QoS guarantee, which has been the focus of the early work on QoS support, requires more resources and is more costly, than the 'weaker' approaches. Yet, in the latter cases, needed resources may not be available leading to worse quality.

The notion of QoS is very different at the various system layers, e.g., in [17], four layers of QoS are distinguished: User QoS, Application QoS, System QoS and Network QoS. The user QoS parameters describe requirements for the perception of multimedia data at the user interface. The application QoS parameters describe requirements for the application services possibly specified in terms of media quality (like end-to-end delay) and media relations (like inter/intra-stream synchronization). The system QoS parameters describe requirements on the processing and the communication services resulting from the application QoS in quantitative (e.g., bits/s or processing time) and qualitative (e.g., multicast, inter-stream synchronization, error recovery or ordered delivery of data) terms. The network QoS parameters describe requirements on network services (e.g., network load or performance). Since each of these layers needs the QoS specified in its own terms, a mapping between them is necessary. While this mapping is an important issue for all networked multimedia applications, no overall solution has been found yet, but only partial approaches for simple conversions, e.g., between transport and network layer, have been devised.

An interesting approach to coordinate all the different negotiations taking place between peers and different layers is the QoS broker architecture [12], in which the broker is a central point for all negotiations taking place, thereby isolating the negotiation partners, which releases them of the burden of knowing all the details for communication between them.

2.3 QoS specification

In general, three QoS parameters are of main interest with respect to the transport of continuous-media data: *throughput*, *delay* and *reliability.*

Throughput, as the most prominent QoS parameter, specifies how much data (maximum or average) is to be transferred within the networked system. In general, it is not sufficient to specify the rate only in terms of bits per second but should describe the packetization, e.g., by specifying the maximum and the average packet size and the packet rate. The reason is that the QoS scheme shall be applicable to various networks as well as to general-purpose end-systems and the costs of operations such as buffer management, timer management, etc., which play an important role for protocol processing, are related to the number of packets processed (and are mostly independent of the packet size). *Delay* as the second parameter specifies the maximum delay observed by a data unit on an end-to-end transmission. *Reliability* pertains to the loss and corruption of data, for that, loss probability and the method for dealing with erroneous data should be specified. *Jitter,* the delay variance, is the fourth QoS parameter typically considered. It is the result from varying delays during processing and transmitting the data. It can be smoothed by buffering at the receiver side which, however, increases the end-to-end delay. All QoS parameters are closely related: The smaller the overall bandwidth of a resource is compared to its load, the more messages will be accumulated in front of it and the larger the buffers need to be to avoid loss. The larger the buffers become, the more likely messages need to wait to be serviced, that is, the larger the delay will get.

The parameters used within the workload description specify *what amount* of data the source intends to transmit. This must be viewed in the context of a *workload model* which specifies *how* the source generates data and feeds it into the system.

2.4 Adaptive mechanisms

Adaptive methods, using scaling and filtering, may be seen as an alternative to reservation-based QoS provisioning, e.g., if reservations are not supported by the used networks. These techniques adapt the generated workload to the available resources by changing the characteristics of the transmitted data stream, e.g., lowering the frame rate of a video stream, and, thus, allow a smooth decrease in quality.

To perform scaling, a feedback control loop is introduced – the load state of network and local end-system resources are monitored and if significant changes occur, appropriate actions must be taken to reduce the load. This reduction can be achieved in various ways: by explicit communication between receiver and sender (the receiver informs the sender to slow down), completely in the network on a hop-by-hop basis, or by feedback from congested network nodes to the sender. Multiple systems have been developed for such scaling especially in the unicast case.

Implementing scaling mechanisms within each single application forces programmers to construct their own mechanisms. Further, leads to interworking problems between applications in case of several streams being scaled simultaneously and raises questions about fairness and balancing between streams. These problems can be solved by 'middleware' approaches (e.g., [9]) where media scaling methods are integrated into a general system support for multimedia.

Distributed applications involving many participating systems are often of heterogeneous nature and, hence, cannot be supported by the described scaling mechanisms. This can happen, for instance, if several multicast receivers require different amounts or different encoding of data from the sender, e.g. due to varying network capabilities or due to different processing capacities of the receivers. Such a scenario can be supported in multimedia applications by using filtering mechanisms (e.g., [13]). Then, an intermediate node within the network changes the amount of transmitted information – by re-coding or, the simpler case, by removing parts of the data and forwarding only a subset of the full information.

The removal of data requires that the full stream has been encoded into a hierarchy of information layers, as is for instance possible with MPEG-2. Then the application can split the data into streams and sends them independently, as has been described, e.g., in [8] using one ST-2 stream and in [4] using one IP multicast group for each layer. A refinement is the use of error detection within the receiver-driven layered multicast approach ([10]) where receivers start out to receive the base layer and add further enhancement layers until they have either subscribed to all layers or until experienced packet loss. Such approaches can result in synchronization problems because the independent transmission of the layers can lead to differing delays among the layers (e.g., due to the use of different routes or priorities). This can be avoided if the application transmits one stream, but describes the structure of the stream (e.g., as part of the FlowSpec), allowing receivers to specify filters to be installed in the network which strip off information not to be transmitted to that receiver [21].

3 QUALITY OF SERVICE ARCHITECTURES

QoS is inherently an end-to-end requirement since for the user, the whole concept of QoS is only attractive if the presentation at the user interface satisfies his/her needs. This means that an overall approach to QoS provisioning is necessary. Unfortunately, most work on QoS support concentrates on certain aspects only, e.g., how to offer QoS in the communication system (which is undoubtful among the most important) such as the work on Tenet (at UC Berkeley and ICSI) [2] or on ATM. Only few QoS architectures using an end-to-end view have been developed which also take end-system components (CPU, memory, disk, …) into account, e.g., HeiTS/HeiRAT (at IBM Heidelberg) [19][20] and QoS-A (at Lancaster University) [5] being among the first, and these most often also lack one or another component. In the following, we briefly describe the approach followed in the Internet community due to its foreseen impact on future use of the Internet. An overview about QoS architectures can also be found in [6].

The goal of the Integrated Services (IntServ [14]) activities, in relation with the work on the RSVP protocol, is to provide a general solution for QoS guarantees in the future internet. The Resource ReSerVation Protocol (RSVP) [24] is a reservation protocol in the Internet suite used to transport FlowSpecs that adhere to Intserv rules between resource managers to perform reservations for *flows*. The primary background of RSVP can be seen within conferencing applications, yet its target is to solve the requirements of other applications as well.

RSVP adds reservation to the existing resp. up-coming Internet protocols (IPv4 and IPv6), relying on those protocols for the interchange of data. RSVP is directed towards the support of large receiver groups, therefore, it is receiver oriented. It allows for the definition of filter specifications for reservations made for a flow, and it provides means to merge reservations.

Instead of hard state controlled mainly by connection-setup and -release, RSVP keeps *soft state*. This state, created similarly to hard state by the exchange of reservation messages, must be refreshed by reservation updates periodically, otherwise, if no such update is received, the reservation times out and is removed.

Two types of descriptions are used for the QoS specification: the traffic specification (TSpec) describes the behavior of a flow, and the service request specification (RSpec) describes the service requested under the condition that the flow adheres to the restrictions of the TSpec.

On this basis, various services are defined. *Guaranteed* QoS requests that the maximum end-to-end delay of a packet is strictly limited to the given value in the RSpec, under the condition that the flow sticks to a certain traffic pattern. This specific service is useful for applications with hard real-time restrictions such as audio transmissions. *Controlled-load* service requests that all network elements behave under any circumstances for a reserved flow that describes its traffic characteristics as they would for a best-effort flow in a situation of light load and without congestion. This service is useful for multimedia applications such as some video conferencing systems which work well on lightly loaded networks without reservations although they fail under heavy load. *Committed rate* QoS requests that all network elements between the source and the targets in a flow maintain the data rate; the overall delay may be quite large, and losses are more probable than with the services mentioned above.

Communication system QoS support (at least for deterministic guarantees) using reservations can only be successful if the data transmission uses exactly the same route on which the reservations have been made (at least as long as no failure occurs). Thus, opportunistic re-routing – route changes due to slightly better metrics but which are not inevitable necessary – must be inhibited, i.e., 'route pinning' must be done, affecting the operation of IP. Route pinning has been discussed several times within the RSVP working group and is currently not provided. Therefore, a hard QoS guarantee cannot be given by RSVP.

4 PAST, ACTUAL AND FUTURE ISSUES

Many architectures and pieces for QoS support have been proposed during the last years. All of these consider only some aspects of the overall end-to-end problem. While such an approach has the advantage of functional decomposition and reduced complexity, it has the drawback of necessary transitions with the potential risk of information loss. Therefore, an integrated approach or at least smooth transitions among the various components would be helpful.

What we believe to be missing in general are large scale end-to-end experiences. Furthermore, several issues must be attacked to provide a complete solution for QoS provisioning. Among these are the (architectural) topics of QoS routing, pricing, security, support for heterogeneous systems, scalability, and reservations in advance which will be discussed briefly in the following.

4.1 Local components

Some effort within the research community has been devoted towards QoS provisioning in servers and end-systems, e.g., CPU and disk scheduling mechanisms. However, most applications do not use such mechanisms resp. are not executed on systems providing such means. So, are these mechanisms still needed for end-systems or servers ? The answer is yes, of course, for servers as well as for other shared resources and systems. For most end-systems which serve only one user and which are used for relative uncritical applications such as video playback, we expect that CPU scheduling and memory management will not find wide-spread distribution. Yet, for high-end usage, e.g., processing of large video streams, and high-quality applications, e.g., video recording, there will be a need for such techniques.

4.2 Routing

QoS driven routing algorithms are needed for the efficient establishment of reservations. These algorithms suggest one or multiple suitable paths towards a given target considering a given set of QoS requirements. An attempt is then made to make a reservation on such a path. Without appropriate routing mechanisms which take QoS requirements into account, the setup of reservations becomes a mere trial-and-error approach.

A QoS driven routing algorithm has to consider the currently available capacity of a resource to avoid an immediate rejection of the reservation attempt and the QoS requirements of the reservation to find a route best-suited for this QoS. It should also consider the resource load after the routing decision to avoid using up the majority of resources on this route.

Some of the problems to be solved with QoS routing are: how much state information should be exchanged among the routers; how often should this state information be updated; must there be a distinction between exterior and interior systems and if yes, how can it be made; is it possible to hide internal details of an autonomous system; can the complexity of path computation be managed ?

QoS routing is currently still in its infancy. At least in the Internet, its necessity, the principle ability to perform QoS routing, and the proposed approaches are currently under highly controversial discussions. The ATM camp, on the other hand, designed PNNI which provides at least some QoS routing support.

4.3 Pricing

An important issue for the future success of distributed multimedia applications, and therefore of QoS methods themselves are the costs for any data transmission (i.e., with or without QoS). QoS methods have to take cost into account as an additional (possible negotiable) parameter. However, as discussed in [16], most research has focused on specific issues; architectural issues have most often been neglected. The issues to be attacked are among many others: who pays for a service, and how is this indicated, especially if the receiver benefits from the transmitted data; can the user specify a limit on its expenditures; how can fairness be provided such that each receiver within a multicast session pays its share, how can payment cross a firewall, how can a department or group be charged instead of the overall company ?

In addition to these aspects which apply to transmissions without QoS, further questions have to be answered in QoS provisioned systems, e.g.: how can resource consumption be "weighted" (e.g., delay vs. loss); what QoS do users accept for a specific price and which pricing schemes do they understand; how can fairness be provided such that all users – benefiting from a reservation made for a multicast transmission – share the costs in a fair manner ?

Recently, some investigations on such questions have started which will hopefully offer answers soon.

4.4 Security

Distributed multimedia applications require secure communication. Otherwise, integrated services networks will not find widespread usage by companies, e.g. video-conferences. This requires, in addition to suitable algorithms for encryption, also firewalls which provide appropriate mechanisms to support such data flows.

Firewalls must be enhanced to allow audiovisual data flows to pass through, potentially after a check of their contents. Firewalls must also provide means for the reservation of resources. In the case that the flow's packets are not simply forwarded but checked by some application code on the firewall, the resources to be reserved are not only network resources but local components as well. Hence, mechanisms such as CPU scheduling and memory management must be considered.

For some application scenarios, privacy should be provided, however, encryption of audiovisual data is too expensive or not necessary. For instance, a personal conversation is most often not strictly confidential, we just want privacy such that no one can eavesdrop without significant effort. Thus, protocols should inhibit that someone can attach to such a stream without permission. Unfortunately, with IP multicast such a misuse is possible.

4.5 Synchronization

Much work has been spent on synchronization protocols and mechanisms for audiovisual data (e.g., [3]) and synchronization at the user interface is undoubtful an important issue. Yet, is it for most application scenarios (except, e.g., multi-user games and other competition-type applications) really such a difficult issue or can it be handled by well-known methods: Synchronized clocks (e.g., using NTP), interleaved streams, and buffer space (as long as it does not increase the delay much) ?

4.6 Heterogeneity

An important issue to be tackled by future systems is the support of heterogeneous systems considering heterogeneity with respect to different system technologies, e.g., ATM vs. Internet protocols, and with respect to varying system capabilities, e.g., broadband vs. narrowband networks and according clients.

The integration of the various network infrastructures into a global, ubiquitous network capable of providing suitable support for multimedia communications must address, e.g., current Internet technology, ATM, and mobile systems. Efforts for the integration of ATM *into* the Internet world started recently (using ATM as a subnetwork with RSVP on top of it). If there will be native ATM applications, e.g., video-on-demand, then there is also the need for a 'side-by-side' integration of ATM and Internet protocols, however, no advanced work on that is existing by now.

Filtering mechanisms, which reduce the amount of transmitted and processed data, can support networks and end-systems with heterogeneous capacities. The basic mechanisms for this have been designed. Future work is necessary, for instance, to evaluate their performance characteristics and to study the possibility to use them with switched networks. Another interesting question is the ability to deploy such filters in active networks [18].

4.7 Scalability

An important condition to be fulfilled by the methods designed for QoS provisioning for shared and distributed components is that they must be scalable – they must remain usable even if they are applied on a very large scale. With respect to multimedia applications, e.g., multicast video conferences, scalability has at least two aspects:

(1) scalability with respect to the number of participants in one application,
(2) scalability with respect to the number of concurrent applications.

The first requirement states that it must be possible to transmit a flow (distributed via multicast) to a potentially very large number of participants. This is, for instance, the case in transmissions from IETF meetings or prominent lectures. To fulfill this requirement, mechanisms for resource sharing among participants and for the aggregation of reservations must be provided, furthermore, there should be no central component which has to process requests from new participants joining resp. old participants leaving such a conference. Based on their receiver-oriented reservation and flow joining concepts, RSVP and IP multicast support this requirement. Using stream group concepts and receiver-initiated stream joining, ST-2+ [15] may support this as well, yet, we are not aware of any large scale experiences validating this.

The second requirement demands that it should be possible to support many independent applications, and hence flows, – for example, thousands of video-conferences (probably with very few participants) and video-retrieval sessions. Therefore, the processing and storage effort per flow must be very small. The abilities of current protocols and architectures in this respect are not yet clear. We believe that it can be only shown by experiments because processing and storage overhead depend not only on architecture but on particular implementation as well. The soft state approach used by RSVP requires periodic message exchange per flow per receiver (potentially reduced by aggregation among receivers) to refresh the reservation and, hence, avoiding the loss of state. This needs transmission bandwidth and, perhaps more important, processing in the routers. Additionally, in order to be able to remove expired soft state, a timer must be kept per each receiver within each flow. While some of these can be aggregated (and efficient timer mechanisms such as timing wheels are known for quite a while), it can be expected that all these timers lead to some non-negligible overhead. Hard state approaches, on the other hand, avoid these problems since they neither require the permanent exchange of refresh messages nor timers per receiver (but per neighbor). However, they must keep the state all of the time and cannot, as soft state approaches may do, throw it away in case of lack of storage capacity.

Which type of scalability is more important depends, on the predominant usage scenario. Currently, it seems to the authors that more attention has been given to the first issue: the scalability of one application. In future, small-sized applications will probably be more important, hence, more consideration should be given to the second issue: the scalability of concurrent applications. In general, more real-world experiments are necessary.

4.8 Reservation in advance

For several application scenarios, the model of immediate reservations, the only one offered by current reservation systems, is not fully appropriate. With this approach, resources are reserved when the application starts. Yet, for events where it is difficult to find a date suitable for all participants, e.g., for video-conferences, the reservation must be set in advance if there is a noticeable blocking probability. Hence, mechanisms for Resource Reservation in Advance (ReRA) [23] are needed. By now, only preliminary results are available on this topic but neither complete end-to-end investigations nor large-scale experiences have been reported. There are several difficult issues which must be resolved before ReRA can find widespread support. It might even be that the required overhead is too large to pay off (it might be more cost effective to reduce the blocking probability by over-provisioning resources). Despite a more complex resource management, some of the problems occurring in ReRA systems are state maintenance and failure handling.

All systems performing an advance reservation must keep the associated state information for a potentially long time. This must be stored in non-volatile memory to survive system failures and regular shutdowns, e.g., due to system maintenance. Alternatively, similar to the approach followed by RSVP, the reservation may be refreshed from time-to-time. As closer the actual usage time comes as more often are refresh messages send [7]. The drawback of this approach is that reservations which have been set might be lost during a system outage and cannot be reset because others have occupied the resources in the meantime.

The treatment of failures occurring between the reservation setup and its use must differ from the steps taken to resolve errors of running applications. The reason is that the application which had lost its reservation is not running. So, which entity is to be notified and by which means? Further, potentially the failure situation can be cleared already before the resources are needed. Should the application be informed in such a case ?

5 SUMMARY

The provision of QoS in distributed systems has been an active resource area over the last years. Early work concentrated on deterministic guarantees exploiting pessimistic approaches by use of worst-case assumptions. Later, more optimistic models providing statistical and predictive services have been offered.

The need for reservations was highly controversial a couple of years ago. Now, the concept of reservation-based QoS has found wide-spread acceptance, nevertheless there are still *'reservations about reservations'* whose advocates consider reservations as too complex and propose adaptive mechanisms as overall solution. Neither reservation-based nor adaption-based QoS support would be necessary if the available system resources would become abundant. Yet, we believe that resource demand grows at least at the same pace as available resources, hence, reservation will be necessary for quality demanding applications and users in the future.

Currently, system components are available, in the labs and partially deployed to 'regular' users, which are able to provide QoS support. An integrated end-to-end approach, from a data source (such as a disk in a video-server), via local and network resources (like CPU and transmission links), towards sinks at receivers (e.g., monitor) has not been settled yet.

Various parts for a complete QoS infrastructure must be developed in the future, for instance, QoS routing, pricing, and security. Perhaps most important will be the verification of the suitability of the proposed mechanisms for the large-scale use: for (few) large multicast sessions with many receivers as well as for many small, concurrent sessions.

REFERENCES

[1] D.P.Anderson, S.Tzou, R.Wahbe, R.Govindan: "Support for Continuous Media in the DASH System", Proc. of 10th ICDCS, Paris, France, 1990.

[2] A.Banerjea, D.Ferrrari, B.A.Mah, M.Moran, D.C.Verma, H.Zhang: *The Tenet Real-time Protocol Suite: Design, Implementation, an Experiences*, IEEE/ACM Transactions on Networking, February 1996.

[3] G. Blakowski, R. Steinmetz: "A Multimedia Synchronization Survey: Reference Model, Specification, and Case Studies", IEEE Journal on Selected Areas in Communications, February 1996.

[4] J.C. Bolot, T. Turletti, S.Schröder, I. Wakeman: "Scalable Feedback Control for Multicast Video Distribution in the Internet", Proc. of SIGCOMM 1994.

[5] A. Campbell, G. Coulson, D. Hutchinson: "A Quality of Service Architecture", ACM Computer Communication Review, April 1994.

[6] A.Campbell, C.Aurrecoechea, L.Hauw: "A Review of Quality of Service Architectures", to appear in ACM Multimedia Systems Journal, 1997.

[7] M. Degermark, T. Köhler, S. Pink, O. Schelén: "Advance Reservation for Predicted Service", Proc. of 5th NOSSDAV, Durham, NH, USA, 1995.

[8] L.Delgrossi, C.Halstrick, D.Hehmann, R.Herrtwich, O.Krone, J.Sandvoss, C.Vogt: "Media Scaling in a Multimedia Communication System", ACM Multimedia Systems Journal, 1994.

[9] T. Kaeppner, L. Wolf: "Media Scaling in Distributed Multimedia Object Services", Proc. of 2nd IWACA, Heidelberg, Germany, 1994.

[10] S.McCanne, V.Jacobson, M.Vetterli: "Receiver-driven Layered Multicast", Proc. of SIGCOMM 1996.

[11] K. Nahrstedt, R. Steinmetz: "Resource Management in Networked Multimedia Systems", IEEE Computer, April 1995.

[12] K.Nahrstedt, J.Smith: "The QoS Broker", IEEE Multimedia, Spring 1995.

[13] J. Pasquale: "Filter Propagation in Dissemination Trees: Trading off Bandwidth for Processing in Continuous Media Networks", Proc. of 4th NOSSDAV, Lancaster, UK, 1993.

[14] R. Braden, D. Clark, S. Shenker, "Integrated Services in the Internet Architecture: an Overview", RFC 1633, June 1994.

[15] L. Delgrossi, L. Berger, "Internet Stream Protocol Version 2 (ST2) Protocol Specification - Version ST2+", August 1995.

[16] S. Shenker, D. Clark, D. Estrin, S. Herzog: "Pricing in Computer Networks: Reshaping the Research Agenda", ACM Computer Communication Review, April 1996.

[17] R. Steinmetz, K. Nahrstedt: "Multimedia: Computing, Communications and Applications", Prentice-Hall, 1995.

[18] D.L.Tennenhouse, D.J. Wetherall: "Towards an Active Network Architecture", ACM Computer Communications Review, April 1996.

[19] C. Vogt, R.G. Herrtwich, R. Nagarajan: "HeiRAT: The Heidelberg Resource Administration Technique – Design Philosophy and Goals," Proc. of KiVS, Munich, Germany, Springer-Verlag, 1993.

[20] C. Vogt, L.C. Wolf, R.G. Herrtwich, H. Wittig: "HeiRAT – Quality-of-Service Management for Distributed Multimedia Systems", to appear in ACM Multimedia Systems Journal – Special Issue on QoS Systems, 1997.

[21] L.C. Wolf, R.G. Herrtwich, L. Delgrossi: "Filtering Multimedia Data in Reservation-Based Internetworks", Proc. of KiVS, Chemnitz, Germany, Springer-Verlag, 1995.

[22] L.C. Wolf: "Resource Management for Distributed Multimedia Systems", Kluwer, Boston, USA, 1996.

[23] L.C. Wolf, R. Steinmetz: "Concepts for Resource Reservation in Advance", Multimedia Tools and Applications Journal, May 1997.

[24] L. Zhang, S. Deering, D. Estrin, S. Shenker, D. Zappala: "RSVP: A New Resource ReSerVation Protocol," IEEE Network, September 1993.