Chapter 2

End-to-End Quality of Service

This section introduces the notion of Quality of Service as it is currently provided by the Internet. As of today, the Internet offers a point-to-point delivery service, which is based on the "best effort" delivery model. In this model, data are delivered to their destinations as soon as possible, but with no bandwidth or latency guarantees. Using protocols, such as TCP, the highest guarantee the network provides is reliable data delivery. This is adequate for traditional applications such as FTP and Telnet, but inadequate for applications requiring *timeliness* guarantees. For example, distributed multimedia applications need to communicate in real-time and are sensitive to the quality of service they receive from the network. For these applications to perform adequately and be widely used, Quality of Service (QoS) must be quantified and managed.
Up to now, QoS has been specified in terms of raw bandwidth and latency for network resources, or CPU and memory utilization for system resources, and the network infrastructures have been deployed to support real-time QoS and controlled end-to-end delays. However, at present the notion of QoS extends from the communication level up to the application level, in order to map QoS application requirements into low-level QoS parameters.
One distinguishing feature of QoS is that the application-level QoS parameters are subject to negotiation between applications and both the system and network components, in order to determine whether the application requiring QoS can be provided by the entire system. Several resource managers (brokers) can be deployed to provide support for the dynamic management of QoS.
In this chapter the traditional concept of Quality of Service at the network level is introduced, a taxonomy of the applications and their QoS requirements is provided, and a more recent notion of end-to-end QoS is presented.

2.1 The Quality of Service

Standard Internet Protocol (IP)-based networks provide "best effort" data delivery by default. Best-effort IP confines the responsibility of meeting application specific requirements at the application level in the end-hosts; thus, the network typically can remain relatively simple [33]. This approach allows one to construct scalable applications, as evidenced by the ability of the Internet to support its phenomenal growth. As more hosts are connected, network service demands eventually exceed capacity, but service is not denied. Instead it degrades gracefully. Although the resulting variability in delivery delays (jitter) and packet loss do not adversely affect typical Internet applications (e.g., email, file transfer and Web applications) other applications cannot adapt to inconsistent service levels. Delivery delays cause problems for applications with real-time requirements, such as those that deliver multimedia, the most demanding of which are two-way applications such as telephony.
Increasing bandwidth is a necessary first step for accommodating these real-time applications, but it is still not sufficient to avoid jitter during traffic bursts. Even on a relatively unloaded IP network, delivery delays can vary enough to continue to

adversely affect real-time applications. To provide adequate service -- some level of quantitative or qualitative determinism -- IP services must be supplemented. This requires extending the network software so as to distinguish traffic with strict timing requirements from traffic that can tolerate delay, jitter and loss of data. That is what Quality of Service (QoS) protocols are designed to do. QoS does not create bandwidth, but manages it so it is used more effectively to meet the wide range of application requirements. The goal of QoS is to provide some level of predictability and control beyond the current IP "best-effort" service.
There is no common or formal definition of QoS. However, there are a number of definitions at the communication level where the notion originated to describe certain technical characteristics of data transmission.

2.1.1 The traditional Notion of Quality of Service

Traditional QoS (ISO standard) was provided by the network layer of the communication system. For example, the ISO standard defines QoS as a concept for specifying how "good" the offered networking services are. So, QoS can be characterized by a number of specific parameters. The OSI Reference Model has a number of QoS parameters describing the speed and reliability of transmission, such as throughput, transit delay, error rate and connection establishment failure probability. From the point of view of the QoS, the ability of a network to deliver the service needed by a specific network applications from end-to end with some level of control over delay, loss, jitter, and/or bandwidth can be categorized into the following three levels of service:

- Best Effort Service -- basic connectivity with no guarantees. The Internet is an example of best effort level of service.
- Differentiated Service -- expedited handling for specific classes of traffic.
- Guaranteed Service -- a reservation of network resources to ensure that specific traffic gets a specific level of service it requires.

2.1.2 Network QoS Issues

Since today's Internet interconnects multiple administrative domains (autonomous systems (AS)) based on IP technology, it is the concatenation of domain-to-domain data forwarding that provides end-to-end QoS delivery (Fig. 3.2).

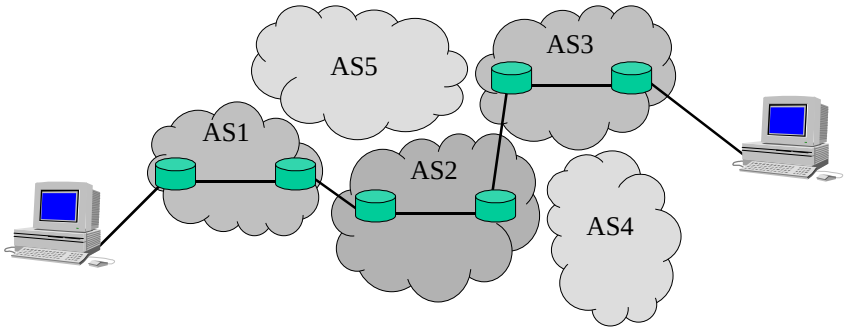

Figure 2.1: End-to-end QoS delivery.

The management of QoS at the network level involves 5 different issues:

- QoS Specification (RSpec)
  In general, each flow should be able to express its delay, jitter, loss and bandwidth constrains.

- Flow Specification (TSpec)
  - Each flow should indicate to the network the load it will impose on it (peak rate, average rate).
  - The network guarantees satisfaction of the QoS specs if the flow does not violate its flow specs.
- Admission control.
  Given the QoS and flow specs, the network should perform admission control.
  A flow is admitted only if:
  - its constraints can be satisfied
  - none of the constraints of existing flows are violated
- Signaling
  The information regarding the admission of a new flow is propagated in the network to set up appropriate state
- Policing
  The network has to ensure that a malicious flow does not violate its flow specifications thereby hurting other flows
- Router Support
  - Edge routers would typically be responsible for admission control, policing, and signaling
  - Core routers will need to implement QoS-sensitive packet classification, scheduling and dropping policies

2.1.3 Approaches to Network QoS Support

Mainly, two are the approaches to provide QoS at the network level, IP based and ATM based, as depicted in Fig. 2.2. The former approach maintains the wide spread packet switching IP technology, and provides differentiated delivery services for individual flows or aggregates by adding "smarts" to the Net and improving on best effort service. The latter approach, instead, substitutes IP technology by providing connection-oriented services.
ATM provides both Data Link and Network OSI levels and is being developed to incorporate the QoS. ATM has the ability to handle a number of traffic types by statistically multiplexing together blocks of data called cells. ATM has five basic traffic classes [6], constant bit rate (CBR), real-time variable bit rate (rt-VBR), non-real-time variable bit rate (nrt-VBR), unspecified bit rate (UBR) and available bit rate (ABR).

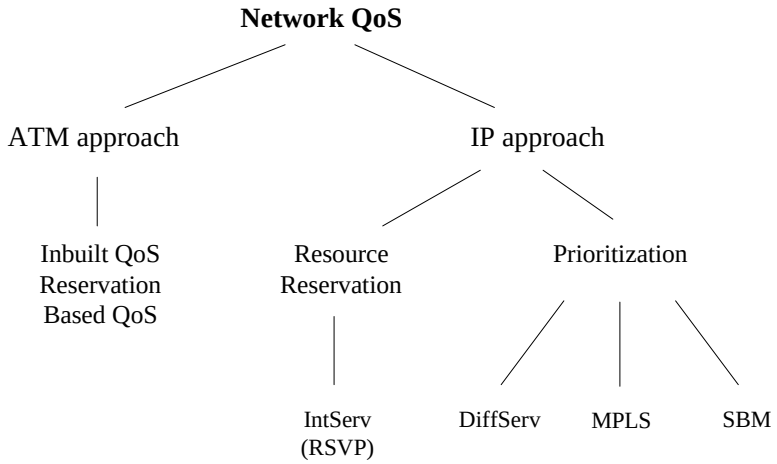

Figure 2.2: Network QoS Taxonomy.

These traffic classes are defined by parameters such as sustainable bit rate (SCR), peak cell rate (PCR), cell loss ratio (CLR) and cell delay variation (CDV). Before a connection can be established the network and host have to agree on the parameters to define the traffic class of the flow. This agreement is called a contract. Connection admission control and traffic policing are used to control the amount of traffic entering the network and so prevent excessive congestion. The admission control policy has to measure the available bandwidth within the network and calculate the effect any new connection will have on this available bandwidth. If a new connection is going to have a detrimental effect on existing connections then the new connection should be rejected. Traffic policing is responsible for ensuring that connections abide by their contracted bandwidth requirements. The traffic policing mechanism is only necessary at the switch connected to the source end node.
Although there is variety of choices, there are two major approaches for supporting QoS into IP based network:

- *Resource reservation* (integrated services): network resources are apportioned according to an application's QoS request, and subject to bandwidth management policy.
- *Prioritization* (differentiated services): network traffic is classified and apportioned network resources according to bandwidth management policy criteria. To enable QoS, network elements give preferential treatment to classifications identified as having more demanding requirements.

These two types of QoS managements can be applied to individual application "flows" or to flow aggregates, hence there are two other ways to characterize types of QoS:

- *Fine Grained* or *Per Flow:* it provides "per flow" QoS guarantees and is represented by the integrated services (IntServ) framework. A "flow" is defined as an individual, uni-directional, data stream between two applications (sender and receiver), uniquely identified by a 5-tuple (transport protocol, source address, source port number, destination address, and destination port number). Each router on the path of a flow participates in admission control. The router keeps track of all admitted flows. For each flow, it allocates bandwidth, buffer space, and a priority. A new flow is not admitted unless the router can satisfy its bandwidth and storage needs as well as its delay constraint. For the signalling IntServ adopts the Reservation Protocol (RSVP) [7]. The problem with integrated services is that per-flow state is required in routers, thus this approach has been criticized for lack of scalability.
- *Coarse Grained* or *Per Aggregate:* An aggregate is simply two or more flows. Typically the flows will have something in common (e.g. any one or more of the 5-tuple parameters, a label or a priority number, or perhaps some authentication information). Flows are classified into a small number of classes and a different behavior for each class is defined.

Applications, network topology and policy dictate which type of QoS is most appropriate for individual flows or aggregates. To accommodate the need for these different types of QoS, there are a number of different QoS protocols and algorithms:

- *ReSerVation Protocol (RSVP):* Provides the signaling to enable network resource reservation (otherwise known as Integrated Services). Receiver initiates network resource reservation by sending an RESV message. The amount of reservation guarantees satisfaction of timing, bandwidth, and buffer size constraints. Reservation establishes soft state along the path to the sender and sets aside the required resources. Soft state is refreshed periodically by the receiver or else it times out canceling the reservation. Although typically used on a per-flow basis,

RSVP is also used to reserve resources for aggregates (as we describe in our examination of QoS architectures).

- *Differentiated Services (DiffServ)*[8]: Provides a coarse and simple way to categorize and prioritize network traffic (flow) aggregates. Differentiated services classify all flows into a small number of classes and define a different "per-hop-behavior" for each class. 6 bits out of the Type-of-Service byte in the IP header are used to define per-hop behaviors. Clients pay the network provider for a certain profile of traffic. Edge routers mark client packets. The 6-bit per-hop behavior code in the IP header tells each core router what to do with the packet.
- *Multi Protocol Labeling Switching (MPLS)* [9]: Provides bandwidth management for aggregates via network routing control according to labels in (encapsulating) packet headers.
- ***Subnet Bandwidth Management (SBM)*** [10]: Enables categorization and prioritization at Layer 2 (the data-link layer in the OSI model) on shared and switched IEEE 802 networks.

2.2 Applications QoS

Different applications running on the same distributed system may have different subsets of relevant QoS parameters, with different values required, and some parameters may be not mutually independent, and may be time depending. For instance, there are 5 types of QoS parameters for the distributed multimedia applications (such as video-on-demand services, teleconferencing [34], computer supported cooperative work [35, 36] and tele robotic applications [37]):

- performance-oriented, e.g. end-to-end delay, bit rate;
- format-oriented, e.g. video resolution, frame rate, storage format, compression scheme;
- synchronization-oriented, e.g. the skew between the beginning of audio and video sequences;
- cost-oriented, e.g. connection and data transmission charges, copyright fees;
- user-oriented, e.g. subjective image and sound quality.

We are particularly interested in the QoS application requirements that involve the network.

2.2.1 Applications Taxonomies

The taxonomy of the networked applications (see Fig. 2.3) revolves around the timely delivery of packets, but addresses other qualitative issues as well. It is based on the simple classification of applications into two principal categories.

First, there are applications that are sensitive to the delay incurred by their data flows as they traverse the network; these applications are *real-time* applications. If a packet arrives late to a real-time application, it is no longer useful.

Second, there are applications that always wait for their data to arrive; they are called *elastic* applications.

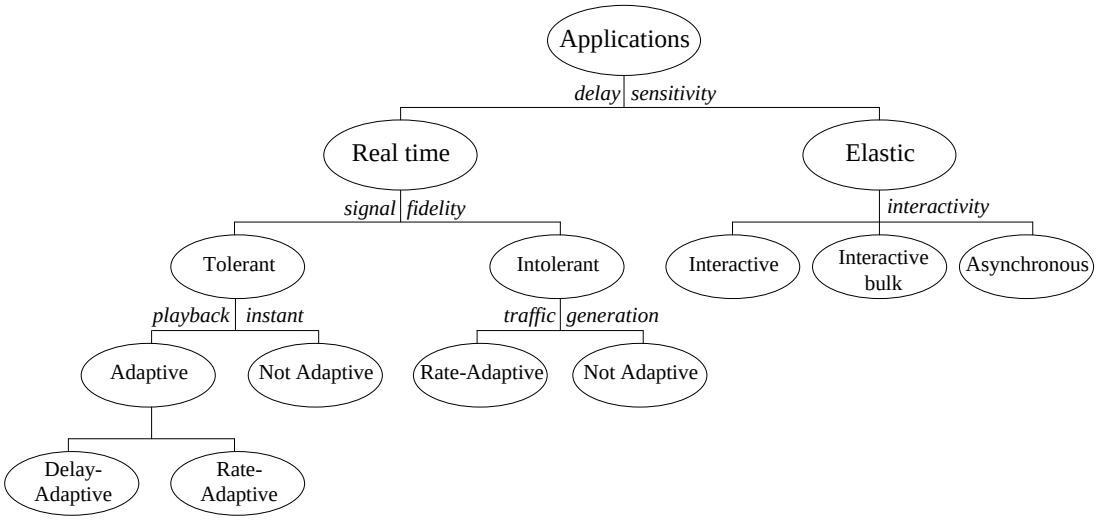

Figure 2.3: Applications Taxonomy.

Elastic Applications: traditional applications such as remote terminal (e.g., Telnet), file transfer (e.g., FTP), name service (e.g., DNS), and electronic mail (e.g., SMTP) are rather elastic in nature, in that they tolerate packet delays and packet losses rather gracefully, and so they are rather well served by the current Internet's best effort service. Also Web browsing may be thought of as an elastic interactive application. A rough set of categorizations for elastic applications might include:

- Interactive burst (Telnet, X, NFS)
- Interactive bulk transfer (FTP)
- Asynchronous bulk transfer (electronic mail, FAX)

The categories are listed here in approximately decreasing order of delay sensitivity. Note that it is the average delay of data that impacts the performance of elastic applications

Real Time Applications: At the other extreme of delay sensitivity are applications with real time requirements. These application are considered *playback* applications.

Data originating at a source is encoded, packetized, sent across the network, decoded by the receiver, and attempted to be replayed at the destination as a replica of what was encoded initially. However packets are likely to experience a range of delays across the network, and earlier packets may take longer to transit the network than later packets. Consequently a real-time application typically buffers the arriving packets at the destination. The buffer allows the application not only to smooth out the delay variation, or *jitte*r, but also to re-order packets if necessary. In any event, data is reconstituted at the destination by establishing a *playback poin*t, the point in time after which a packet is delivered. That point is usually a fixed offset from the original departure time. In practical terms a packet is buffered according to this departure time, then a buffering delay is added to smooth the play out of the data stream. Depending on the application, the buffering delay may be quite short on the order of milliseconds, as in a conversational audio application, or seconds and potentially minutes, as in a streaming video-on-demand application. Packets arriving after the playback point are discarded. These are considerations that impact the design of buffering at the end-systems.

In choosing a playback point, an application approximates the amount of delay it is able to tolerate. This approximation may be based on a delay bound promised by a particular service class, on actual measurements, or on predictions about future packet delays. The delay bound need not be fixed for the lifetime of the data flow.

The performance of a playback application is measured by two parameters: latency and fidelity. Some playback applications are particularly sensitive to latency, such as a distributed music performance that relies on interactions between the end systems, whereas other applications are less so, such as an Internet lecture that behaves more like a unidirectional streaming application. Likewise, applications exist that are quite demanding of fidelity of the real-time signal, whereas others are more willing to forego exactness. This latter dimension of performance, leads to two further classifications of the real-time playback applications into those that are tolerant and intolerant of signal fidelity. For those that are tolerant a *predictive service* class is proposed. For intolerant applications a service model called *guaranteed service* is proposed.
The calibration of the offset delay is quite important, as it determines the latency of the application. Furthermore, a realistic offset is critical to the fidelity of the application. If the offset delay is incorrect, more packets arrive late and are dropped, making the data stream less complete. Alternatively late packets may cause the application to adjust its playback point, which introduces distortion in the signal. Applications of this sort are considered *adaptive* playback applications. A final reaction to late packets might be for the application to alter its traffic generation scheme, e.g., switch to a less demanding audio or video encoding. These applications are known as *rate-adaptive* playback applications. Although this technique may reduce delay, it necessarily compromises data resolution. All of the previous examples show that late packets decrease playback fidelity.
While a tolerant application has the option of selecting one of these methods for handling late packets (the appropriateness of one over the other depends on the application itself), an intolerant application must use a fixed offset delay to avoid loss of fidelity. The intolerant application must choose a “perfectly reliable upper bound on delay”, the maximum delay of any packet, whereas the tolerant application can resort to using a “fairly reliable upper bound”, which is conservatively predictive [38]. The motivation for using predictive service over guaranteed service is that it offers better network utilization and presumably lower cost.

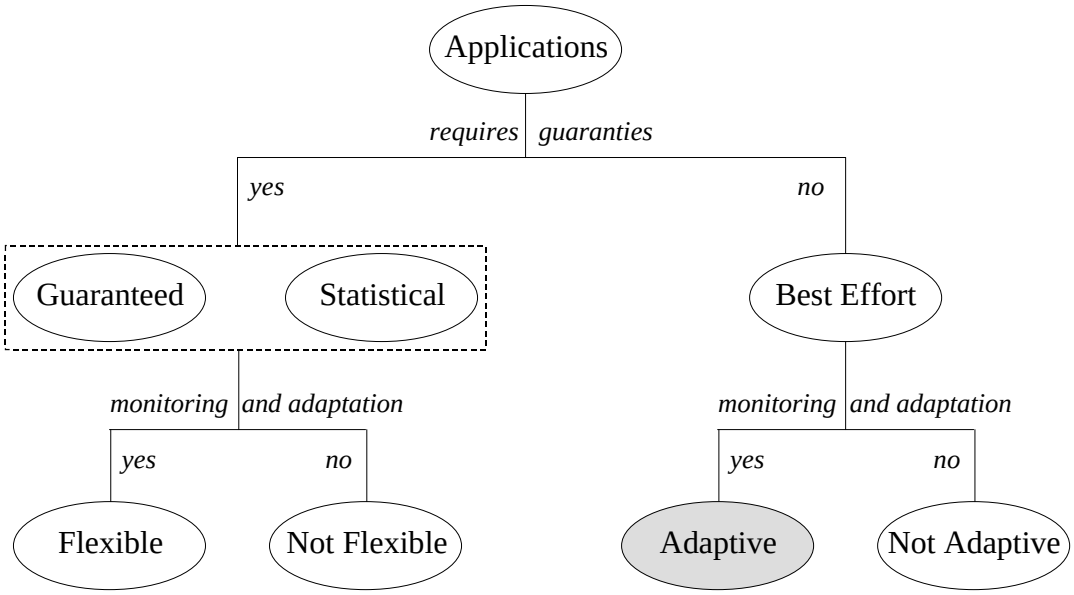

Figure 2.4: Guaranties required by the Applications

A simpler taxonomy involves real time constraints of the applications. Real-time constraints are classified as *hard*, *firm* and *soft*, depending on the consequences of the constraint being violated. A task with *hard* real time constraint has disastrous

consequences if its constraint is violated (for instance, a missile control). A task with *firm* real time constraint has no value to the system if its constraint is violated. A task with *soft* real time constraint has decreasing, but usually nonnegative, value to the system if its constraint is violated.
Another taxonomy (see Fig. 2.4) classifies the networked applications depending on the guaranties they require from the underlying network; there are three classes of applications:
a) *Guaranteed* – all deadlines are guaranteed to be met all the time. This application gets highest priority and will need resources to be reserved considering the worst case situation.
b) *Statistical* – deadlines are guaranteed to be met with a certain probability (e.g. a service that guarantees that 90% of the deadlines will be met over an interval). The statistical behavior of this class needs to be monitored and maintained.
c) Best Effort – no guarantees are given for meeting deadlines. Deadlines are met on a best-effort basis with the resources leftover from those reserved for guaranteed services. These applications can be pre-empted by higher priority classes, and no resources are reserved.
Most of the applications that require guaranties from the network are *flexible*, in the sense that they can tolerate a QoS range of input quality and resource availability beyond a certain minimum level, and can improve their performance, provided that a larger share of resources is available. Flexible applications monitor the distributed resource and dynamically adapt themselves by modifying their QoS requirements and by negotiating a new QoS level with the environment. If resources above the minimum requirements are shared among all applications, statistical multiplexing gain can be improved. In addition, for the flexible applications that involve interactive activities that cannot be predicted beforehand, it may be hard or impossible to specify a maximum demand for QoS.
Finally, it is worth pointing out that “an application is best-effort based” means it does not require guaranties, but it may adopt several software architectures, mechanisms and policies in order to provide the users with the suitable QoS. For instance, the QoS of a web service, as perceived by the users, may be increased by using caching at the browser side, pre-fetching at the proxy, and replication at the server side (several web server replicas) and at the browser side (mechanism for binding the best replica). Although these three different approaches do not require guaranties from the network, they can provide the user with timely responses, thus increasing the user's satisfaction.
In particular, *adaptive* best-effort applications monitor the performances of the involved resources and dynamically adapt themselves by modifying their behavior in order to meet the QoS requirements of the user, but unlike flexible applications they do not reserve network resources. The negotiation phase of flexible applications becomes the adaptation phase of best effort applications. For instance, the web browser involved in the previously cited web service may periodically monitor the performances of each web replica server, in order to retrieve a web resource to the best replica.

2.3 The Notion of End-To-End QoS

The original QoS parameters apply mostly to lower network protocol layers, and are not meant to be directly observable or verifiable by the application. Consequently, the resulting QoS coverage of OSI as a whole is incomplete and even inconsistent. This situation, while acceptable when communication networks were used mostly for non-time-dependent data, is no longer satisfactory with respect to the new requirements

stemming from distributed multimedia systems. As time-dependent data are becoming prevalent in multimedia applications, the entire distributed system must participate in providing the guaranteed performance levels.

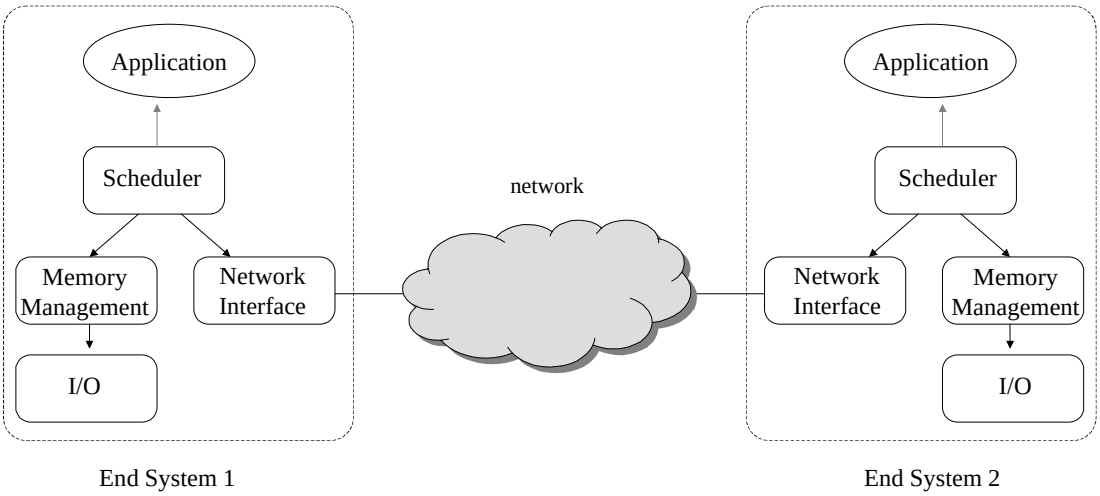

Figure 2.5: End-to-end QoS entities

The QoS notion must be extended because many other services contribute to the end-to-end service quality. This in turn means that both the network and the end-system have to contribute towards achieving this end-to-end QoS. Fig. 2.5 above illustrates a basic diagram of the entities involved in providing this end-to-end QoS. Beyond its intuitive meaning as system characteristics that influence the perceived quality of an application, there is little consensus on the precise meaning, let alone the formal definition of QoS. For instance, the ITU/ISO Reference Model for Open Distributed Processing [39] refers to QoS as ``A set of quality requirements on the collective behavior of one or more objects". According to ISO, ``QoS characteristics are intended to be used to model the actual behavior of systems. It is defined independently of the means by which it is represented or controlled". This type of definition is too general, since it tends to include all system parameters without distinction. In [40] the following working definition is proposed: “By quality of service we mean the set of those quantitative and qualitative characteristics of a distributed multimedia system, which are necessary in order to achieve the required functionality of an application. This includes the presentation of multimedia data to the user, and in general the user's satisfaction with the application".

2.3.1 End-To-End QoS Entities

To discuss further QoS and resource management, we adopt a layered model of the end-to-end networked applications with respect to QoS (see Fig. 2.6). That model consists of three layers: application, system (including communication services and operating system services), and network. Above the application may or may not reside a human user. This implies the introduction of QoS in the application (application QoS), in the end system (system QoS) and in the network (network QoS). In the case of having a human user, the QoS model may also have a user QoS specification.

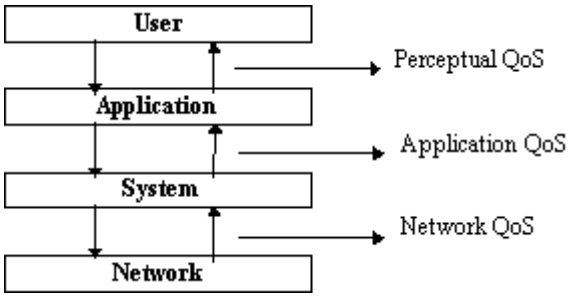

Figure 2.6: QoS Levels

The system layer includes (see Fig. 2.7) several modules [41], such as task scheduler, memory management and I/O subsystem. Therefore, the QoS system parameters mirror the requirements on CPU scheduling (e.g., task start time, priority, duration and deadline), on memory management (buffer allocation) and on I/O management (for example, the maximal frame rate for a video device). In this view, the QoS requirement originates with an application process which conveys it in terms of QoS parameters to other system components. This is generally followed by a negotiation process whereby the components of the system determine if collectively they are capable of satisfying the requested QoS level. The QoS of a given system is expressed as a set of (parameter-value) pairs, sometimes called a tuple; each parameter can be considered as a typed variable whose values can range over a given set. Note that different applications running on the same distributed system may have different subsets of relevant QoS parameters, with different values required, and some parameters may not be mutually independent.

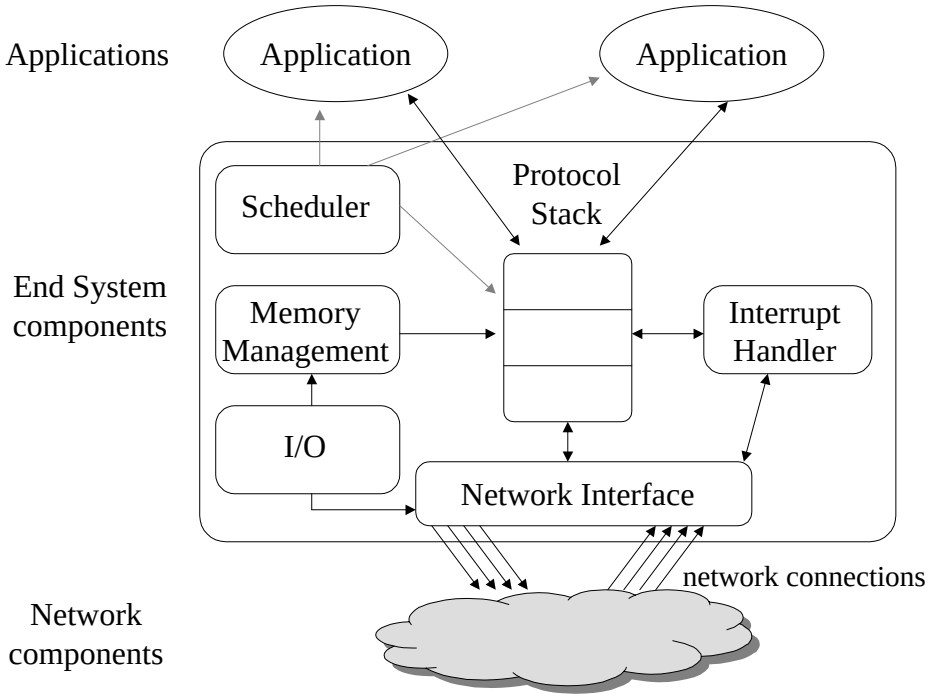

Figure 2.7: QoS End System Components

Resource management is a fundamental task performed by the operating systems. Most current mainstream operating systems implement resource management policies which are oriented towards overall system throughput and fairness. They perform reasonably well for a mix of interactive, batch processing, or server applications. Real-time

operating systems are designed to give hard guarantees on resource allocations to allow applications to perform time-critical tasks. This is achieved by allocating fixed shares of resources to processes for their entire lifetime, or at least for a relatively long duration. Multimedia operating systems (e.g., [42], [43], [44], [45], [46]) give soft real-time guarantees for resource allocations to processes which may be renegotiated dynamically at runtime, encouraging applications to adapt to the changing overall system load. This is usually achieved through explicit resource allocation and revocation. This process is commonly referred to as *QoS-Management*. In conclusion, one distinguishing feature of QoS is that these QoS parameters are subject to negotiation between applications and both system and network components, in order to determine if these QoS parameters may be provided by the entire system.

References

[6] ATM Forum Specification, [Online] http://www.atmforum.org/

[7] L. Zhang, S. Deering, S. Shenker, Zappala, ``RSVP: A New Resource Reservation Protocol", IEEE Network, Vol. 7, pp. 8-18 (1993).

[8] Y. Bernet et al., "A Framework for Differentiated Services," Internet Draft, draft-ietf-diffserv-framework-02.txt, Feb. 1999.

[9] Multiprotocol Label Switching Architecture, RFC 3031, January, 2001.

[10] "SBM (Subnet Bandwidth Manager):A Protocol for RSVP-based Admission Control over IEEE 802-style networks', RFC 2814, May, 2000.

[11] K. Nahrstedt and J. M. Smith: "The QOS Broker", IEEE Multimedia, Vol. 2, No. 1, pp. 53--67 (1995).

[12] P. Y. Wang, Y. Yemini, D. Florissi, J. Zinky, ``A Distributed Resource Controller for QoS Applications", appeared in NOMS 2000, Hawaii, April 2000.

[33] J. Saltzer, D. Reed, D. Clark, "End to End Arguments in System Design", ACM Transactions in Computer Systems, November 1984. [Online] http://www.reed.com/Papers/EndtoEnd.html

[34] P.V.Rangan, ``Video conferencing, file storage, and management in multimedia computer systems", Computer Network and ISDN Systems, 25, pp. 901-919 (1993).

[35] M. Altenhofen, et al., ``The BERKOM Multimedia Collaboration Service", in Proceedings of the ACMMultimedia 93, ed. P.Venkat Rangan, pp. 457-464, ACM Press, Anaheim (1993).

[36] T.Gutekunst, T.Schmidt, G. Schule, J. Schweitzer,and M. Weber, ``A Distributed Multimedia Joint Viewing and Tele-Operation Service for Heterogeneous Workstation Environments", in Proceedings of International Workshop on Distributed Multimedia Systems, Stuttgart (1993).

[37] K. Nahrstedt, J. M. Smith, ``Application-Driven Approach to Networked Multimedia Systems", in Proceedings of the 18th Conference on Local Computer Networks (1993).

[38] B. Braden, D. Clark, S. Shenker. ``Integrated Services in the Internet Architecture: an Overview". RFC 1633. Integrated Services Working Group of the IETF, June 1994. Available as ftp://ftp.isi.edu/in-notes/rfc1533.{ps,txt}

[39] ITU/ISO, ``Information Technology - Open Distributed Processing - Reference Model: Foundations", ITU-T Recommendation X.902 (Nov. 1995). Also ISO/IEC International Standard 10746-2.

[40] A. Vogel, B. Kerhervé, G. v. Bochmann, J. Gecsei, ``Distributed Multimedia Applications and Quality of Service – A Survey", IEEE ..

[41] K. Nahrstedt, J. M. Smith, ``Design, Implementation, and Experiences of the OMEGA End-Point Architecture", IEEE Journal on Selected Areas in Communications, pp 1263-1279, Vol. 14, No. 7, Sep 1996.

[42] G. Coulson, G. Blair, P. Robin, and D. Shepherd. ``Supporting Continuous Media Applications in a Micro-Kernel Environment", in O. Spaniol, editor, *Architecture and Protocols for High-Speed Networks*. Kluwer Academic Publishers, 1994.

[43] M. B. Jones, P. J. Leach, R. Draves, J. S. Barrera. ``Modular Real-Time Resource Management in the Rialto Operating System". In *Proceedings of the 5th Workshop on Hot Topics in Operating Systems (HotOS-V)*, May 1995.

[44] I. M. Leslie, D. McAuley, R. Black, T. Roscoe, P. Barham, D. Evers, R. Fairbairns, E. Hyden. ``The Design and Implementation of an Operating System to Support Distributed Multimedia Applications". *IEEE Journal on Selected Areas In Communications*, 14(7),1280-1297, September 1996.

[45] J. Nieh, M. S. Lam. ``The Design, Implementation and Evaluation of SMART: A Scheduler for Multimedia Applications". In *Proceedings of the 16th ACM SIGOPS Symposium on Operating Systems Principles, Operating Systems Review*, pages 184-197, Saint-Malo, France, October 1997.

[46] R. Rajkumar, K. Juvva, A. Molano, S. Oikawa. ``Resource Kernels: A Resource-Centric Approach to Real-Time Systems". In *Proceedings of the SPIE/ACM Conference on Multimedia Computing and Networking*, January 1998.

[47] T. F. Abdelzaher. “QoS-Adaptation in Real-Time Systems", PhD thesis, University of Michigan, Ann Arbor, Michigan, August 1999. [Online] http://kabru.eecs.umich.edu/papers/thesis/zaher_thesis.ps.gz.