Layer Integrated Quality of Service Management[⋆]

Tino Hutschenreuther[1] and Sebastian Schönberg[2]

[1] Chair Computer Networks
[2] Operating Systems Dept. of Computer Science
Dresden University of Technology, Mommsenstraße 13, 01062 Dresden

Abstract. Emerging high speed networks give rise to new classes of applications like teleconferencing or video-on-demand which demand Quality of Service (QoS). To provide QoS support on an end-to-end basis, the demand of integrating network, transport, and operating system appears since applications compete for network access, processor time and QoS guarantees. This paper discusses QoS requirements to the transport and operating system. Several approaches for QoS management schemes are discussed and problems identified. A possible solution is presented and explained which operates on end-system management on the basis of an admission control for resources. Further QoS specification on the basis of a structured description of the application data contents at the transport system interface is presented. A generic description is presented and the resulting management functionalities inside the transport system are discussed.
Keywords: Quality of Service, QoS specification, QoS management, operating system management

1 Introduction

The provision of a specific quality of service to distributed multimedia applications such as conferencing, application sharing or virtual desktops imposes new requirements on data transmission and the management of the end-system. Such requirements are for instance high throughput, fast transfer rates, including error-free delivery, and time guarantees. The network must not only provide fast data transfer but also guaranteed delivery. Continuous media, like audio and video have to be delivered error-free to the user under well defined time constraints to achieve the desired quality during the presentation. Achieving guaranteed, end-to-end delivery in networked computer systems will require the solution of several multimedia specific control management problems at end-system and network levels. The layered architecture of end-systems implies considerable data movement. Different communication layers may also have different protocol data unit sizes. If the upper layer wants to transmit a large data size, the data units have to be broken into the size required by the underlying layer. This segmentation is performed by the sender, and the underlying layer's data units must be reassembled at the receiver. Generally speaking, every communication layer has its own understanding for the quality of service. That implies an adequate mapping between these different layers. Over the past years, substantial research effort has been made in designing architectures supporting quality of service. Researchers also dealt with new transport protocols like VMTP [8] or XTP [16] to give adequate support for various kinds of traffic over networks which offer no guarantees. With ATM [2], a guaranteeing network technology with huge available bandwidth has been developed. It provides specific support for connection management, flow control, congestion avoidance, segmentation, reassembling and routing making similar functionality in transport protocols redundant. Several approaches have been developed addressing the enhancement of architectural frameworks with QoS [1, 5, 11, 12, 10, 6, 4]. But current architectural frameworks lack a QoS-based application programming interface (API) and transport mechanisms that support end-to-end QoS [12]. Hence applications use application level protocols to take care of their data and mostly negotiate their demands directly with the network (pure ATM interface) [12]. This leads to a decentralized data management at the end-system which suffers from concurrent access to processor, memory and network. In such situations, end-systems can slow down or even stop executing applications or receiving data from the network due to high buffer utilization, exhaustive processor usage or bad application memory management. Since receiving and sending data has high priority in end-systems, the exchange of large data amounts can lead to disturbance of normal application behavior. In spite of faster processors and better hardware support the end-system becomes the bottleneck within the entire environment.

1.1 Abstraction layers

There are several abstraction layers below the visible interface at the end-system: user level; application level and system level, including communication and operating system services. Since all layers include services, a QoS configuration has to be considered in all layers. The used QoS specifics within these levels are:

⋆ This work is sponsored by SFB358 of the German research community

User level: Here, quality of service is usually described in terms of media quality and media relations. Media quality includes the characteristics of source and destination and the "human feeling". Media relations specify relationships among the streams transmitted, such as inter- or intra stream synchronization.

Application level: Within the application level quality of service is described in application oriented parameters of the media, resulting from the user requirements and application used. Such parameters are for instance frame rate, picture size or the requested quality in terms of coding formats or bit depth.

System level: QoS parameters describe communication and operating system requirements, which must be derived from the application QoS. These parameters need to be specified in qualitative and quantitative terms. Quantitative criteria are those which can be measured, while qualitative criteria specify the expected service. The system level is divided into two parts which have to be handled together to ensure the running of networked applications.

- **Communication level:** The network QoS parameters have to be specified in terms of network load and network performance. Hence, calculated traffic parameters depend on network QoS parameters and are specified in a traffic contract. Here QoS is expressed in terms of latency, throughput, loss and burst characteristics.
- **Operating system level:** These requirements are the end-system resources required for execution of the processes. Such resources are the CPU, access to devices, main memory and system bus bandwidth. In order to give adequate support to the user, the requirements have to be mapped according to the system QoS parameters.

The aim of this work is the establishment of an application related interface for QoS specification. From that, a systematic mapping onto end-systems and network resources, as well as the management of the available resources can be derived. The next section illustrates the current situation in end-system management and discusses other approaches which tackle this problem. After that our approach of an integrated QoS management in end-systems is presented and the interfaces and strategies are discussed. A future outlook concludes this paper.

1.2 Current Situation in End-systems

Applications can specify their requirements facing the communication subsystem. Mostly only quantitative parameters like throughput or delay can be specified at a very low level. Applications are not able to specify their requirements to the operating system, because of the lack of knowledge about it. Furthermore, there is hardly a possibility to specify all resources in their quantity, needed for execution of the task. For transmission of data further processes rather the initiating are necessary, so indirect resources are needed, which cannot be influenced by the application process. This is for instance the memory used by or the CPU-time needed for the execution of the communication protocol. A first step in the right direction was made with the AQUA-framework [17], where operating system resources become integrated with the QoS management in the end-system. But it is not sufficient to schedule one resource, rather the consideration of all available resources and the granularity of the specified requirements is important for the powerful management of all applications in an end-system. Similar approaches dealing with resource management and QoS mapping can be found in [17, 7, 3] and [14] . But to some extend they try to restrict the applications in their behavior instead of introducing a resource management.

2 Integrated Management of QoS

Integrated management of QoS means the consideration of the user requirements, operating system constraints, network resources and application requirements. All of these requirements have to be brought together into one model to give adequate support to applications at the end-system.

2.1 Central Resource Management Component (RMC)

The intention of this central RMC comes from the problem of addressing both operating system and network management.

In order to give adequate support to the application, an integrated approach is necessary, because of the use of indirect resources and the mutual dependence of the different management components. So a clumsy scheduling of resources and network parameters for one application can waste resources and in spite of that not satisfy the user requirements. This RMC works as an admission control for applications which need guarantees for execution and transmission bandwidth. It can be stated that this approach is not suitable for an overall scheduling at the end-system. This approach is only necessary for some applications like video applications or the collaboration of

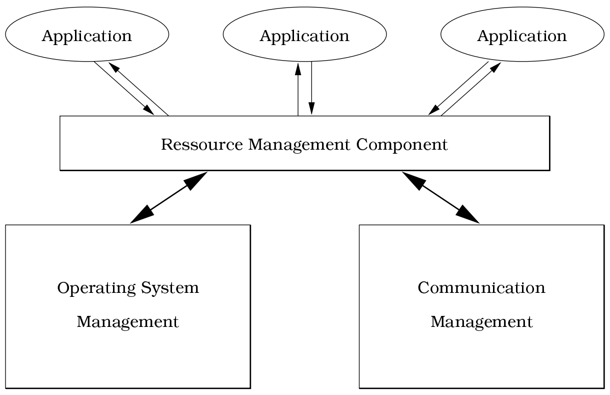

Fig. 1. Model of the Resource Management Component

several people at a shared application. All other applications which are not able to specify their requirements or do not need guarantees from the end-system, are defined as background applications and have to share the available resources, not used by the scheduled applications.

As shown in Figure 1, the RMC is responsible for the scheduling and allocation of operating system and network resources. Furthermore, a systematic mapping of quality of service parameters can be achieved by providing an application-oriented interface for describing the requirements. From these requirements the specific demands of the application has to be derived in connection with the knowledge about indirect resources which are needed for the execution of the application.

2.2 Operating System Management

The operating system has to provide an interface for the specification of the application requirements in terms of needed CPU-time, memory and devices. Therefore an operating system is needed, where the specification of such parameters is possible and the operating system itself has knowledge about the execution time of its own processes. A further important feature is the support of real-time applications by introducing a real-time personality for efficient access to the operating system resources. Furthermore the operating system should provide information about the capacity of system resources and currently running applications. This information can be used by the RMC to determine the importance of application requests and the availability of system resources or devices. On start, every application negotiates the resources it needs by passing a set of QoS parameter to the RMC. This set consists of qualitative parameters such as

- video frame rate
- video picture size
- color model
- audio quality (like phone, radio, CD, mono/stereo) or quantitative parameters like
- bus bandwidth,
- CPU,
- memory.

By use of a mapping database, the RMC converts the qualitative to quantitative parameters. These parameters should be as machine independent as possible. For scheduling, a CPU specific description can be derived by an algorithm-specific description, called Process-Mix (PM) (1). This consists of a proportion of the rate between Integer (IPU), Floating-point (FPU) and Memory Instruction (MPU). This triple is either well known (e.g., for MPEG decoding) or can be determined by a simple instruction profiler which scans the appropriate part of the program. Weighted with a CPU-specific triple, a per-instruction-code (PIC) can be calculated (2). This PIC multiplied by the given number of instructions divided by the CPU frequency is the part of the CPU this application takes. In combination with the frequency how often this code is executed per time the reservation for the scheduler is made (3).

$$PM = \{IPU, FPU, MPU\} \tag{1}$$

$$PIC_{CPU} = PM \times CPU \tag{2}$$

$$Sched_{Res} = PIC_{CPU} * \frac{Instructions}{Freq_{CPU}} * loops \tag{3}$$

2.3 Network Management

The integrated management of quality of service is necessary for an efficient support of applications in the end-system and end-to-end between the involved hosts [9, 15]. In our model an application level interface is described, which allows a high level specification of the quality of service demands of the application. These requirements are mapped down to the network and are translated into network parameters of the underlying layers. From there a reservation of bandwidth and the negotiation of the parameters with the corresponding end-system can be done. Content depended data handling The OSI reference model insists on the separation of application dependent and transport dependent layers. So the knowledge on application specifics is hidden from the transport system. This is partly negated by specifying QoS parameters for connections. But the transport system can only adapt the bandwidth and buffer space according to the adjustments of the application. This does not prevent the receiver of a data stream from handling useless data, most important out of time data. Our approach favours a central scheduling inside the transport system to control the load at the end-system generated by network access. Network resources are countable. Therefore an intelligent resource and an admission control are needed inside the transport system to prevent instable conditions due to lack of buffers and processing capabilities. This central scheduling can only be efficient with the knowledge of the data content transported through the network. With this knowledge and on the basis of a priority scheme the sending entity can decide to drop less important data in times of network congestion. The receiving entity can make decisions when data is lost or is being corrupted inside the network without transporting all data to the application. So a better end-system performance can be achieved and more resources can be made available for user processes. In our opinion the quality of service parameters, normally given to the transport system (bursts, bandwidth and rate), are not enough to handle the data in a content specific way. Most data items have several parts, because applications usually take a sort of containers to transport the needed information. A container can be a header, in which the structure of the following information is coded, a real container like it is used in RTP, or a similar kind of package, in which the information is transported. Furthermore application data itself is structured. Usually there are parts which are independent, and parts which depend on each other. This is very application specific and depends on implementation, standards and used formats. Most applications use the same structure for every data item transmitted, so that the structure has to be given to the transport system only once at connection setup time. By providing this information additionally to the normal Quality of Service parameters the transport system is able to handle delay and loss constraints in a more specific way according to the requirements of the application. The application has to provide a description of their Application Data Unit (ADU) in terms of type, structure, dependencies and priorities. The following structures give a general ADU definition.

```
define enum { repeat, repeat_if_possible, no } ERR;

typedef struct  {
        short   priority;               /* relative priority inside the ADU */
        short   depend[5];              /* array with segment numbers */
        ERR     err;                    /* way of error control for the segment */
} SEG;

typedef struct  {
        short   num;                    /* number of segments */
        bool    adu_type;               /* type of the ADU */
        u_long  delay;                  /* max_delay */
        SEG     seg[num];               /* array of information about segments */
} ADU;

typedef struct  {
        long    max_frame_size;         /* max. size of the frame, given to the transport layer*/
        long    max_frame_rate;         /* rate in sec-1 */
        float   loss_rate;              /* overall loss rate for the connection */
        ADU     adu;
} QoS;
```

The ADU description consists of an ADU type, a delay and a varying number of segment descriptions. The adu_type can be used to specify distinct application coding formats like MPEG, MJPEG or RTP, if there is a database where such descriptions are stored. At the moment the adu_type is used to define whether an ADU is structured or not. The delay is specified for every ADU type to allow different delays for different types of an ADU. So it is possible to distinguish between data and control information inside the transport system and use different connections for them. The segment description contains information about the priority of the segment inside the ADU, which can be used for discarding strategies inside the transport system. Discarding of whole ADUs could occur if the transmitting application broke the QoS contract, the network is too busy, not all parts of the ADU are transmitted error free or the receiving application cannot consume at the transmitted rate. Further depend characterizes the dependencies between the various segments inside an ADU. If a packet of an ADU is lost and can not be retransmitted in time, the protocol can observe the dependencies and drop also the resulting useless packets. The retransmission algorithm at the receiver side depends on the error control given for a segment, a more fine grained approach than only on the basis of a loss_rate and gap_loss of the whole media stream [13]. On the sender side the application negotiates a QoS contract with the transport system by providing this structure additionally to the normal QoS parameters for the connection. During transmission the application only has to provide the ADUs and an array with the length of the segments. So the transport system can execute an user oriented segmentation in the case of an underlying ATM network into AAL5 packets by utilizing the length of the segments and set priorities for the resulting AAL5 PDU's according to the corresponding segment descriptions. If the network load is too high to send all packets within the requested time, low-priority packets can be discarded. If one AAL5 PDU is lost on receiver side, the receiver can detect the corresponding segment and the dependencies and executes the error control provided for the segment. If there is no time left for retransmissions the whole segment and all depending segments are discarded or if necessary the whole ADU. This can also apply if the receiving application cannot consume the stream at the required rate. Then either the whole ADU or parts of it are discarded.

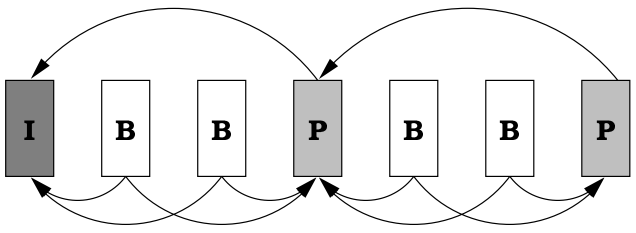

Fig. 2. Dependencies within a GOP in MPEG

With this structure various descriptions can be made. One example is given now for the description of a MPEG stream. There are several opportunities. On the one side the application can describe the composition of the frame as an ADU and on the other side you can define a Group Of Pictures (GOP) as an ADU. Then you describe a frame as a segment of the ADU. Figure 2 shows the dependencies within a GOP. If P-frame is lost at receiver side, all B-frames using references of this P-frame can be discarded. Also the following P-frame can not be decoded and may be omitted. But the description given by the application should comply to the transmitted structure. For example, an input sequence of IBBPBBP will be arranged in the output sequence as IPBBPBB for transmission. This is only one example, the description can also be applied to other structured data, like mails with attachments, web pages, every video or audio coding format and so on.

3 Conclusions

In this paper, we presented an approach for central admission control in end-systems. Guarantees are only provided for applications, applied for resources at this component. All others are defined to be background applications, which have low priority and will be executed when time permits. This approach allows the high level specification of resources and behavior of the application. This is a necessary requirement for managing end-system and network resources effectively. With this knowledge about running applications the admission of new applications on basis of their resource requirements becomes possible. So a running out of resources of the end-system or the network can be prevented and applications don't obstruct each other. The inclusion of a distributed network resource management to this approach is planned in the future.

References

1. A.A. Chien and J.H. Kim. Approaches to quality of service in high-performance networks. In *Parallel Computer Routing and Communications Workshop*, July 1997.
2. ATM traffic management specification, Oktober 1996.
3. Y. Bao. fficient resource management for satisfying diversified qos guarantees. Technical report, Department of Computer and Information Science, University of Delaware.
4. I. Barth, G. Dermler, and W. Fiederer. Levels of quality in cinema. Technical report, University of Stuttgart, Institute of Parallel and Distributed High Performance Systems.
5. C.R. Becker and K. Geihs. Maqs - management for adaptive qos-enabled services. Technical report, SFB 403, 1997.
6. A. Campbell and G. Coulson. Implementation and evaluation of the qos-a transport system. In *Protocols for high speed networks V*, pages 201–218, Sophia Antipolis, France, 1996. PfHSN.
7. A. Campbell, G. Coulson, and D. Hutchison. A quality of service architecture. *Computer Communications Review*, 1(2):6–27, Apr 1994.
8. D. R. Cheriton. *RFC1045: VMTP: Versatile Message Transaction Protocol – Protocol Specification.* SRI Network Information Center, February 1988.
9. S. Fischer and R. Keller. Quality of service mapping in distributed multimedia systems, 1995.
10. COMET Group. xbind: `http://www.ctr.columbia.edu/comet/xbind/xbind.html`.
11. A. Hafid and G. v.Bochmann. An approach to quality of service management for distributed multimedia applications. In *Third IFIP International Conference on Open Distributed Computing*, Brisbane, Australia, 1995.
12. Huard and et.al. Meeting qos guarantees by end-to-end qos monitoring and adaption. In *(HPDC-5)*, Syracuse NY, August 1996.
13. J.-F. Huard. kstack: A user space native-mode atm transport layer with qos support. Technical Report CU/CTR 463-96-29, Columbia University New York, 1996.
14. K. Nahrstedt and J.M. Smith. End-to-end qos guarantees: Lessons learned from omega. Technical Report UIUCDCS-R-96-1957, UILU-ENG-96-1720, University of Illinois, May 1996.
15. D.J. Wetherall and et.al. Ants: A toolkit for building and dynamically deploying network protocols. In *IEEE OPENARCH*, San Francisco CA, Apr 1998.
16. Xpress transport protocol specification, 1995.
17. R. Yavatkar and K. Lakshman. Integrated CPU and network qos management in an end-system. In *IWQOS'96*, pages 167–178, 1997.