Resource reservation and media scaling
techniques for multimedia communication

L. Delgrossi

Andersen Consulting Technology Park, 449 Route des Crêtes,
06902 Sophia Antipolis, France

Abstract

Emerging distributed multimedia applications demand efficient delivery of high volumes of digital audio and video data. As the current transport systems do not provide real-time services for the communication, it is necessary to experiment with new techniques. This paper comprises a short overview of two promising techniques: resource reservation and media scaling. It provides the motivation for the adoption of these techniques, and it shows how they can be exploited to facilitate the exchange of multimedia data across generic internetworks.

1 Introduction

Emerging distributed multimedia applications as video-on-demand and remote collaboration require efficient delivery of digital audio and video data. This is difficult to achieve with today’s communication systems because of the high data volumes that are involved and the time-critical nature of multimedia data.

Even when using efficient data compression techniques, a two-hours movie still requires several Gigabytes of storage and a transmission throughput of about 2 Mbits/s. That is, a few video streams are sufficient to fill a 10 Mbits/s Ethernet or a 16 Mbits/s Token Ring LANs. In the same way, although most PCs offer today multimedia capabilities, they still have limitations in their disks capacities and in the power of their processors. Multimedia traffic brings the systems of today to their limits.

Multimedia data is time-critical and has to be consumed at the appropriate time in order to maintain its correct semantics. Consider a camera that generates video frames and transmits them over the network to an end-system for playout. It is crucial that each frame reaches its destination and is available for playout at the appropriate time. If the frame is not available in time, its contents are no longer useful and the frame should be discarded.
As the existing transport systems cannot provide real-time guarantees on the data communication, new techniques have been recently proposed and experimented with. This paper provides a short survey of two promising techniques: resource reservation and media scaling. First, the communication scenario and the motivation for these techniques are introduced in Section 2. Then, resource reservation and media scaling are illustrated respectively in Section 3 and 4. Conclusions are summarized in Section 5.

2 Communication Scenario

Figure 1 describes a common scenario where a source is connected to a set of receivers through a router. In a typical multimedia application, the data flows continuously from the source to the receivers. The picture highlights the main resources involved in this communication: storage devices, CPU, and network.

This scenario suggests a connection-oriented communication model, where packets belonging to the same multimedia data stream (e.g. a single packet may contain one or more images or audio samples) share stringent requirements in terms of throughput and delay: the data has to be delivered to the destinations at a certain transmission rate within a specific time frame. Unfortunately, the performance achievable with today’s computer systems is hardly adequate, i.e. the systems do not provide sufficient capacity.

In his “window of scarcity”, represented in Figure 2, Anderson [AnDH90] shows that it was impossible to handle audio and video with the end-systems built in the 80s and that only in a distant future end-systems will provide abundant resources that enable fully interactive multimedia applications. In the meantime, the available resources are sufficient to handle audio and video data, but scarce.

To cope with this resource scarcity, it is necessary to improve the efficiency of multimedia systems: this can be done by achieving optimal utilization of the available resources. In fact, since resource capacity is not abundant, appropriate scheduling and careful resource management are essential to the provision of multimedia services in the next future.
There are several reasons why today’s systems do not make an efficient use of the available resources from the point of view of multimedia communications:

- the current systems do not take into account the needs of specific data flows; all packets are served on a best-effort basis, often following First-In-First-Out criteria, no matter whether they contain time-dependent data or not. On the contrary, it is necessary to be able to separate traffic types with different demands in order to provide each with the appropriate service.
- since access to the resources is free, applications compete to use them. This results in low-priority batch applications disturbing high-priority real-time applications, but also in real-time applications disturbing each other. It would be preferable to control the access to the resources, so that a few applications are served well rather than providing all applications with bad service.
- the desire to serve all requests led in the past to resource scheduling techniques that have the main goal to avoid the starvation of low precedence executions. This democratic approach works well when no applications have real-time demands, but it is inappropriate for time-critical applications.

The previous considerations suggest improvements in the way resources are scheduled and in general handled. A first approach consists of dedicating the entire resources to a specific task or application. In the past, solutions have been architected, where the entire PC is used for video playback or for traditional data processing. This approach is hardly feasible in a future perspective because it prevents from building fully integrated computing environments. A picture of the role played by performance, scheduling and resource management in multimedia systems is provided by Herrtwich [Herr91].

3 Resource Management

Resource management functions are used to control traffic through the resources and the resources internal behavior. These are the task of the Local Resource Manager (LRM). The LRM knows which resources are on the system and what capacity they can provide. It cooperates with the communication software when
data flows with specific service requirements need to be established. An example of resource manager can be found in [VoHN93].

![Diagram of Resource](image)

**Figure 3: Example of Resource**

Figure 3 sketches the internal behaviour of a resource. At its input interface the resource is presented with requests to be processed. They might be packets from the network or a set of instructions to be executed in case of the CPU. In the following, we just call them incoming messages. The resource internally serves the incoming messages and produces outgoing messages as output. For instance, packets are forwarded to the next hop in the network. In Figure 3, messages with different colour belong different data streams. With respect to this resource behaviour, the LRM has to be able to perform the following functions:

1. **Admission Control**: this test checks whether there is sufficient capacity left to handle a new data stream. Information is maintained into local data structures that describes the current load of the individual resources. If the available capacity is insufficient, the new data stream must be rejected. In Figure 3, admission control determines whether it is possible to use a new input queue.

2. **Quality of Service Computation** calculates the best possible performance the resource can provide for the new data stream under the current traffic conditions, e.g. throughput and delay values are here computed. The throughput and delay values determine the amount of incoming messages in a given data unit and therefore they determine the minimum length of the input queues.

3. **Resource Reservation**: this function reserves the resource capacities required to meet the desired service. This is done by updating the local data structures. In Figure 3, reservation means that an input queue has been associated with a specific data flow, i.e. q1 is associated to dark-coloured messages and q2 to light-coloured messages.

4. **Quality of Service Enforcement**: the service guarantees must be enforced by appropriate scheduling of resource access. For instance, an application with a short guaranteed delay must be served prior to an application with a less strict delay bound. The scheduling algorithm indicates which of the input queues in Figure 3 has to be served next.
The first three functions are executed during stream setup. They either allow for a new stream to be built and reserve the corresponding resources or they notify the application that the current load of the system makes it impossible to run the new stream. The last function, enforcement by scheduling, is on the contrary executed at run-time and it is the most difficult function to realize in practice. Unfortunately, it also is the core function for bandwidth management: reservation without enforcement of reservations is not useful.

Algorithms for CPU scheduling include the classical approaches for real-time processing, earlier-deadline first (EDF) and rate monotonic (RM) [LiLa73]. A deadline-workahead scheme is suggested by [Ande93]. Examples of network resource scheduling with focus on networks based on token-passing admission control schemes can be found in [NaVo92].

3.1 Reservation Strategies

Making use of resource reservations techniques immediately brings up issues on what strategies to adopt when reserving the resources’ capacity. Reservation techniques assume that this capacity can be partitioned into multiple fractions and that these fractions can be assigned to specific applications. Determining the amount of resources to be reserved is a critical task: in the following, some of the possible alternatives are presented.

The first problem is that, in order to determine a good estimate of the amount of resource capacity to be reserved, an application must be able to accurately predict the generated workload. This is less difficult with continuous media, because such media usually generate a periodic workload, but it can be extremely complicated for bursty workload, e.g. generated by real-time simulation applications.

A transport system providing full service guarantees needs to be prepared to handle the worst-case, i.e. maximum workload has to be assumed when reserving the resources. This way, the transport provider is certain to meet the users requirements under normal circumstances as long as the users do not exceed the agreed upon maximum workload. This approach takes into account the worst-case and it is therefore called pessimistic. It has the advantage of avoiding all conflicts among concurrent applications, but it leads to a poor utilization of the resources: the worst case may occur very rarely in practice, so that a possibly large portion of the reserved resources is almost unused.

Alternatively, an optimistic strategy can be adopted. Instead of reserving for the worst-case, the system reserves for the average case, e.g average workload. This strategy works well when all applications behave normally but it leads to conflicts in the case of different applications simultaneously experiencing their maximum workload: in this situation the QoS guarantees may be violated for a small time interval. The optimistic approach is convenient for those applications that do not require full service guarantees and it has the benefit of a better resource utilization. The real-time software to control a nuclear device needs a pessimistic reservation; for an audio-conferencing application where some occasional noise can be tolerated an optimistic reservation is sufficient. Pessimistic and optimistic strategies are discussed in [Herr91] and [Herr90].
Normally an application has no hints on how long the user is going to utilize a data channel, thus it explicitly informs the LRM of both beginning and end of a working session. Resources are released only after the explicit end signal is received by the LRM.

Recently, Resource Reservations in Advance (ReRA) techniques have been proposed [DSWW95]. They assume an application knows about the time and duration the resources will be in use. Applications communicate this information to the LRM any time before the beginning of the session. This scheme conveniently models business activities, e.g. regular one-hour meetings on Tuesdays can be reserved in advance.

4 Media Scaling

In some cases, it is not possible to exploit reservation-based techniques to provide multimedia applications with performance guarantees. Ethernet networks, for instance, allow free access to the medium and use a logarithmic algorithm to reduce the probability of collisions. This simple scheme, in general efficient, is hardly adequate for time-critical applications that require upper bounds on the medium access time.

Since the network capacity cannot be reserved, in such networks temporary congestion are possible. The desire to allow for multimedia communication over these networks even in case of congestion led to the design of media scaling techniques [DHHH93]. The idea behind media scaling is the opposite of resource reservation. Reservation techniques let the user specify how much bandwidth is needed and the network then sets this bandwidth aside for a certain stream. With media scaling, first it is determined how much capacity is available in the network and then the bandwidth of the media stream gets accordingly regulated.

Media scaling is based on the following assumptions: a) it is possible to sub-sample a data stream and to present only some fraction of its original contents, b) functions are available that are able to detect the data traffic variations over the network and in particular, the beginning and the end of network congestion.

When a network congestion is detected, e.g. by monitoring functions, the stream is reduced (scaling down) so that it requires less bandwidth. Receivers are presented with subsamples of the stream, i.e. with a lower quality. Still, this is preferable to nevertheless attempting to deliver the whole stream with the likely result of receiving no data at all and further contributing to the congestion. The stream can be delivered in its full quality (scaling up) later on, after the end of the congestion has been detected.

In general, it is useful to scale a data stream before it enters a system bottleneck; otherwise it is likely to contribute to the overload of the bottleneck resource. Scaling at the source is usually the best solution: there is no need for transmitting data in the first place if it will be thrown away somewhere in the system anyway.

The following section describes which kind of compression schemes lend themselves to transparent and non-transparent scaling.
4.1 Scalable Elements of Media Streams

In a multimedia system, scaling can be applied to both audio and video data. For audio, transparent scaling methods cannot be used easily because presenting only a fraction of the original data is easily noticeable to the human listener. Therefore, audio streams are predestined for non-transparent scaling methods, e.g., by changing the sampling rate at the source of the stream.

Scaling depends on the media encoding format. Not all formats lend themselves to the same kind of scaling. For video streams, the applicability of a specific scaling method depends strongly on the underlying compression technique. There are several domains of a video signal to which scaling can be applied:

- **Temporal scaling** reduces the resolution of the video stream by decreasing the number of frames transmitted within a time interval.
- **Spatial scaling** reduces the number of pixels of each image in a video stream. For spatial scaling, hierarchical arrangement is ideal because the compressed video is immediately available in various resolutions.
- **Frequency scaling** reduces the number of DCT coefficients applied to the compression of an image.
- **Amplitudinal scaling** reduces the color depths for each image pixel.
- **Color space scaling** reduces the number of entries in the color space, e.g., switch from color to greyscale presentation.

Obviously, combinations of these scaling methods are possible.

4.2 Issues on Media Scaling

Several issues are to be discussed when adopting media scaling techniques in a multimedia transport system. Deciding whether the network is congested or not is a difficult task. Even assuming that the system is able to detect that a data packet is late, the lateness of a single packet does not necessarily indicate a congestion. It is more appropriate to wait for a sequence of late packets, although in this case the length of the sequence becomes critical.

Once a congestion has been detected, it is necessary to decide how rapidly should the system respond, i.e., when the stream has to be scaled down. If the congestion is brief, it is wiser to drop some excess or late packets at the receivers instead of having the source reduce the stream. Unfortunately, it is in general not possible to foresee the duration of a congestion.

Monitoring functions can help detecting the beginning of a network congestion but are useless to detect the termination of a congestion condition. The only practical way is to declare that the congestion is terminated after a certain time has elapsed and no new beginning of congestion has been detected.

5 Conclusion

The transmission of digital audio and video data streams requires careful handling of the resources involved in the communication. Resource reservation techniques can be adopted to provide timely delivery of data. We described such techniques and discussed the different strategies that can be adopted when realizing them. Media scaling techniques can be used to adjust the volume of data.
flows based on the network load, in case it is not possible to exploit resource management functions.

References


