End-to-end QoS Provisioning through Resource Adaptation

D. G. Waddington and D. Hutchison
Distributed Multimedia Research Group,
Computing Department,
Lancaster University,
Lancaster LA1 4YR,
UK
e-mail: [dan, dh]@comp.lancs .ac.uk

Abstract

With the progression of multimedia middleware and guaranteed network services, developers are now presented with flexible frameworks for the development and deployment of distributed multimedia applications. New and more advanced applications are supporting end-to-end Quality of Service (QoS) guarantees through the configuration and management of distributed resources. As an effect of the sharing of network and end-system resources across multiple clients, coupled with their dynamically changing state, the general end-to-end availability of resources in a distributed environment is variable and potentially unpredictable. Thus, the provision of QoS constrained services in a distributed environment demands carefully controlled and co-ordinated management mechanisms. In this paper we discuss the requirements for QoS adaptation mechanisms and QoS-based distributed resource management, together with our approaches to QoS adaptation and policing, with issues concerning the incorporation of these mechanisms in our recently developed Distributed Resource Management Architecture (DRMA).

Keywords

End-to-end QoS, Resource Management, QoS Adaptation

1 INTRODUCTION

The growth of general networked computing performance is being closely followed by the exploitation of this capability by QoS constrained applications. Many of

---

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35388-3_42

H. R. van As (ed.), *High Performance Networking*

these applications, such as video conferencing and distributed collaborative environments, have dynamically changing and potentially unpredictable QoS requirements. This problem is exacerbated by the heterogeneous nature and varying capabilities of today's end-systems and global network infrastructures. As a result, conventional resource reservation and admission techniques cannot guarantee QoS without considerable over-booking and inefficient resource utilisation, something which is particularly undesirable in systems that are required to maintain high levels of resource sharing.

To address this problem, and avoid any adverse impact to the end-user, distributed applications and their infrastructures need to become adaptive. This means that either applications must tolerate fluctuations in resource availability or that the supporting infrastructure can itself mould, through distributed QoS management, to the dynamically changing requirements of the applications. Furthermore, certain applications, especially distributed multimedia applications with relatively demanding QoS requirements, simply cannot operate outside strict resource requirements, and therefore the need for QoS management arises. For example, Video on Demand (VoD) applications often incur varying resource requirements through changing counts of 'viewers' with different end-system capabilities (set-top boxes to high performance workstations). Furthermore, because of the heterogeneous nature of network connections and end-system resources, offering better guarantees and higher performance is often a infeasible and uneconomic solution. If QoS is not carefully managed, and resource utilisation scrupulously co-ordinated, then the desired levels service across the application are difficult to maintain. Other effects of poor QoS management include loss of synchronisation through delayed updates, deadlocks in shared resources, and failure of mission-critical applications through resource unavailability. The term 'QoS management' encompasses the maintenance of a required level of service through the co-ordinated configuration and control of end-to-end resources. Because of the obvious proliferation and acceptance of distributed object computing, and more importantly its usefulness in addressing the problem of QoS management in an open distributed system, proposals have already been made for object-based management frameworks (Dang, 1995)(ISO, 1997)(Waddington, 1997). However, up till now, much of the work has only addressed issues of QoS specification and the projection of useful QoS abstractions and services to the application programmer.

In this paper we discuss our approach to distributed QoS adaptation, furthering Lancaster's original work on QoS maintenance and adaptation in the QoS-A framework (Campbell, 1994). We now place more emphasis on the provision of QoS guarantees in distributed processing environments, whilst also addressing the problems of QoS-driven resource management and adaptation in the middleware infrastructure. In section 2 we discuss the general requirements for distributed resource management and the implications of supporting QoS constrained applications. Then, in section 3, we introduce the rationale for adaptive

QoS-based resource management, together with support for resource monitoring and QoS degradation predication, and its use in instigating adaptation processes. In section 4 we discuss some of the issues concerning the engineering and implementation of QoS adaptation into distributed multimedia middleware platforms, and in particular our recently developed Distributed Resource Management Architecture (DRMA) (Waddington, 1997). Finally in section 5 we present our conclusions.

2 QOS PROVISIONING THROUGH RESOURCE MANAGEMENT

In order to sustain the QoS requirements of a given continuous media application, all resources involved in the handling and processing of data from end-to-end, must be carefully co-ordinated and managed. The end-to-end QoS is some function of the resource utilisation of a distributed application, from client through network to server.

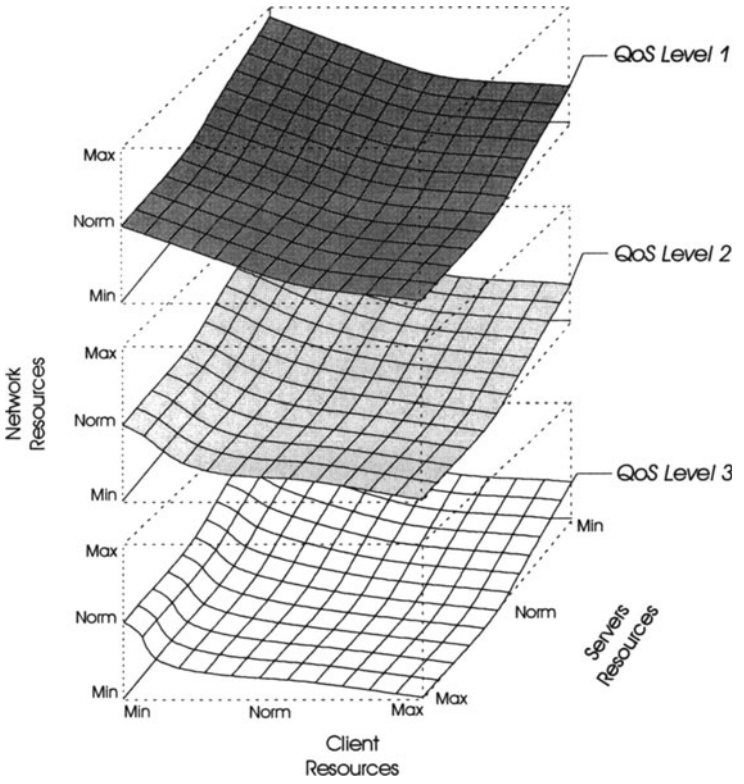

Figure 1 Resource Capacity Regions for different levels of QoS

Any application which needs to provide a QoS-bound service must maintain its distributed resource utilisation within a finite space, which we call the *resource*

capacity region. Figure 1 illustrates this relationship between resource usage in the end-systems and the network. The region's surface represents the balance of resources required to sustain a particular end-to-end QoS; hence each level of service has associated a different capacity region. For each level of service, provided that an application maintains its resource utilisation within the defined region, QoS is sustained (note that each capacity region associated with a particular level of QoS is represented by its own graph).

A concept of capacity regions was proposed by Columbia University's work on meeting end-to-end QoS guarantees over high performance packet-switched networks. (Hyman, 1995) introduces the concept of a *schedulable region* which describes a finite space representing the number of calls of a given class a particular network link can support. This concept was later extended (Lazar, 1995) to incorporate capacity regions for end system resources, known as *multimedia capacity regions*, where the summation of the two capacities is used to determine end-to-end call admission.

If an application is unable to maintain its utilisation within the region, possibly due to the unavailability of resources, then degradation in the application QoS will occur. On the other hand, if an application's resource utilisation is not close to the surface of the capacity region, then resources are not being used optimally which may result in the degradation of QoS for other applications sharing the resources. Thus, distributed applications providing end-to-end QoS should strive to maintain resource utilisation as close as possible to the balance defined by the surface of the resource capacity region. So far, our discussion of the resource capacity regions has been limited to distributed applications based upon single client/network/server scenarios. However, the concept can be readily extended to multi-point communication scenarios, resulting in additional dimensions to the capacity graph (we are not limited by 3-dimensional space, as capacity regions are actually realised as non-visual multi-variable relationships).

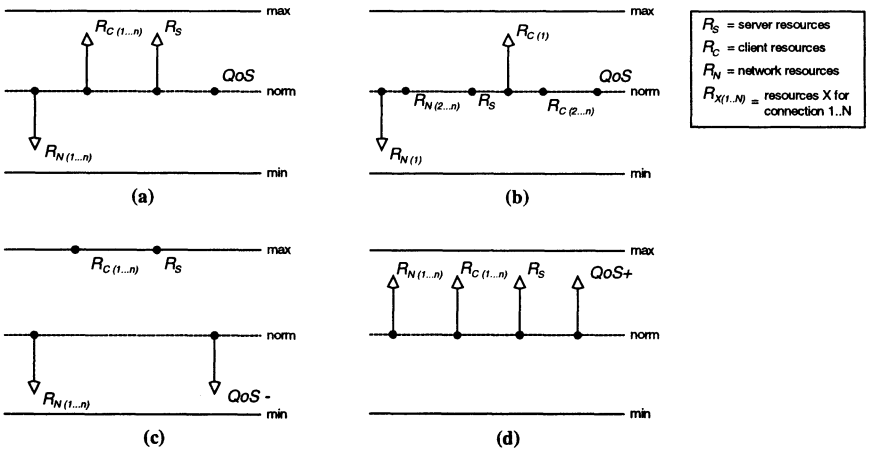

Figure 2 Example Adaptation Scenarios in One-to-Many Relationships

QoS adaptation is the process of maintenance control, facilitated through alterations to either the balance and distribution of resources or to the application's level of service, on short time scales. Adaptation processes often occurs as a result of *QoS notifications*, usually emitted from QoS monitoring mechanisms, which indicate a change in the observed service affected through the availability of some element of the end-to-end resources. Notifications may indicate a imminent lack of resources and hence reduction in service quality (QoS degradation) or a failure to maintain service quality through a complete loss of resources (QoS failure). Whether degradation or actual failure occurs, QoS adaptation is required to either adjust the balance of resources to maintain graceful degradation, or recover service quality, or alternatively inform the end-user of the need to alter to a new level of service (change in level of service could entail dropping one or more media channels, or reducing information resolution). Ideally adjustments in resource balance, affected by the adaptation mechanisms, will satisfy the function of resource utilisation described by the surface of the resource capacity region. In doing so this is likely to involve one or more entities of the resource set (client, server or network) increasing their own resource utilisation to counteract deficiencies in the failing entity.

It is also important to realise that the rules of adaptation that we apply to simple client-network-server applications can be readily scaled to applications which communicate in one-to-many and many-to-many relationships. Figure 2 presents some example adaptation processes in a one-to-many environment. Figure 2(a) illustrates a loss in the network resources (for all connections) which is counteracted by an increase in client/server end-system resource utilisation (perhaps through the incorporation of compression techniques). In examining adaptation processes in multi-party relationships there are additional factors which must be considered, for instance the number of remote end-systems a particular server is communicating with. The example scenario described in figure 2(b) shows a drop in resources for a single connection, whilst other network resources remain stable. In this scenario it does not make sense to increase utilisation of the server and thus, of all the clients, in response to a single client failure. Therefore, an alternative action would be required, such as increasing the resource utilisation for the individual client (maybe by employing data reconstruction or forward error control techniques). If a particular set of resources fail and counteractions cannot be made, then a reduction in QoS is inevitable(see figure 2(c)). In such a scenario, the adaptation process is required to release other shared resources or inform the end-user of a need to change the level of service. Finally, figure 2(d) illustrates that in order to effect an increase in end-to-end QoS then an increase in all of the end-to-end resources is required.

2.1 Hierarchical QoS Management

From an engineering perspective, QoS management in a distributed system is a substantially complex task. An approach which has been proposed in the Distributed Resource Management Architecture (DRMA) (Waddington, 1997) is hierarchical QoS management. This technique breaks down the task of managing end-to-end resources by dividing the problem into a set of finer-grained point-to-point requirements which are structured as hierarchical bindings (see figure 3). By doing so, mapping and monitoring processes become distributed from end-to-end, enhancing scalability and avoiding problems of centralised control. The hierarchical management approach is also suited to the engineering of adaptation mechanisms. At the upper levels of the structure, adaptation mechanisms are responsible for coarse-grained actions, including reconfiguration and change of service. Descending towards the leaves of the hierarchy, adaptation mechanisms become finer-grained, usually in the form of atomic resource control. The processes responsible for the adaptation actions are maintained within binding components (bindings are simply communication abstractions between one or more potentially distributed object interfaces). Furthermore, each individual binding is responsible for maintaining, through monitoring and adaptation, the point-to-point QoS characteristics which are defined by its interfaces.

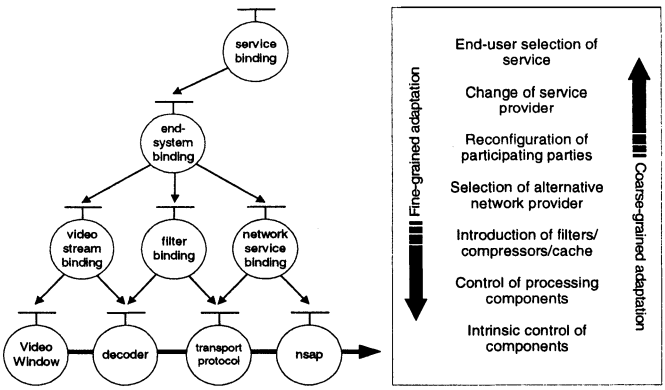

Figure 3 Hierarchical QoS Adaptation

2.2 Core Distributed Resources

The term 'resource' is inherently vague. Resources can exist at varying levels of abstraction. For instance at the highest level the term could include an end-system or a network node. At a lower level, a resource could represent a physical device or some system wide shared resource such as a network connection or access to a physical disk. However, we believe that there is a finite set of resource, in the end-

system and the network, in terms of which all other resources can be described. Therefore we propose that, at least to begin with, we should concentrate on monitoring a limited set of resources and make adaptations accordingly.

Table 1 Core Distributed Resources

Area	*Resource*	*Description*
End-System	CPU	Processor cycles
	Physical Disk Access	Disk I/O requests
	Memory	Paged and non-paged memory usage
	Cache	File and I/O caching
	Auxiliary Memory	Device buffers, video memory
	I/O Devices	Serial/parallel ports
	Peripherals	Video camera, microphone, etc.
	NSAPs	End-system access Ports
Network	Bandwidth	Traffic throughput
	Buffer Space	Router buffer requirements
	Switch/Router Ports	Network channels/paths

The core resources, as described in the above table, are shared by applications across the distributed environment. This finite set covers the majority of resources which are shared in a distributed system. By careful management and control of these, we can begin to prioritise resource utilisation and hence offer QoS-guarantees.

2.3 Resource Scheduling

In order to successfully share resources across distributed applications and, furthermore, offer sufficient guarantees on their availability (an obvious requirement for time critical continuous media applications), resources need to be scheduled. Scheduling is the process of determining the availability of resources at a particular instant in time. Through resource reservation, an application can request resources and in return the system can determine whether sufficient resources are available to service the given request.

Scheduling algorithms can only be effective if they are used in advance of the admission of the resource. However, the time scale between resource admission testing and admission is dependent upon the scheduling algorithm. Many algorithms, of varying complexity and usually focused at either the network or the end-system, have been suggested (Hyman, 1995)(Anderson, 1993). However, in scheduling resources for multimedia applications the use of exhaustive or statistical techniques is often inappropriate (applications such as VoD can be statically

analysed, but dynamically changing applications such as video conferencing cannot). It is suggested that simple heuristic scheduling techniques are preferable, offering a low processing overhead, and thus being more suited to resource requirements which vary in real-time. A basic model for resource reservation is offered by (Wolf, 1995). In this model, resources are *requested* by the application to the individual resource manager (i.e. network or end-system). The resource request describes the resource requirements of the application and the duration of the requirement. Provided that there are sufficient available resources, the resource manager returns a *confirmation*, otherwise the request is refused and a *failure* is returned. On successful completion of the negotiation phase, the client contacts the resource manager at the point when previously reserved resources are required. The *demand* is acknowledged, and the client can then use the resource for the duration of the reservation. This technique of reservation in advance does require that the duration of the reservation can be calculated *a priori* and that the resource usage is scheduled from the reservation request. There are techniques, such as partitioning, which can be used to couple with non-advance resource reservation systems; however we feel that the reservation in advance scheme is suitable for our purposes.

3 ADAPTIVE RESOURCE MANAGEMENT

Many of today's applications continuously adjust their resource requirements. This dynamic nature is particularly evident in distributed multimedia applications which demand strict levels of QoS. Furthermore, the problem is intensified in general purpose distributed environments because resources, in the network and the end-system, must be shared across multiple contending applications. Some operating systems, especially real-time systems, employ resource scheduling techniques (as previously discussed) in an attempt to alleviate the problem. Reservation and admission does allow a system to offer firm guarantees provided that the resource can be guaranteed, e.g. wireless communication bandwidth cannot always be guaranteed. However, many applications, such as distributed games, have varying resource requirements which cannot be predicted. One solution is to over-book resources and hope that the application does not demand resources beyond those made in the reservation. This is not an ideal solution as it is likely to result in the inefficient use of resources. So what is the rationale behind the use of resource reservation and admission techniques in a general purpose operating system at all? Why not simply rely directly upon monitoring and adaptation? In response, we suggest that compared with adaptation processes, resource reservation and admission processes are relatively lightweight and their processes demand little of the system. Furthermore, the majority of applications know their resource requirements *a priori*, and therefore are suited to a model of admission and reservation.

Coupled with the problem of varying client requirements, is the indeterminate nature of end-to-end resource availability. This difficulty is particularly evident within wide-area QoS guaranteed networks, which create an end-to-end connection through the concatenation of multiple hops. Because each hop in the connection continuously change state, QoS routing techniques must be employed in determining a suitable end-to-end route. As a result of the inability to determine the exact state of end-to-end resources, we propose the use of two-tiered 'loose' reservation and admission in order to maintain a best-effort management of resources. Loose resource reservation implies that approximate state metrics are used in admission and reservation, and that only statistical guarantees can be made. In the event that an applications resource requirements do change, then adaptation techniques are used to maintain QoS. Thus, the resulting system combines the benefits of resource reservation and admission, with the flexibility of monitoring and adaptation.

3.1 Monitoring and Prediction

Monitoring is the process of observing the utilisation of resources and/or QoS characteristics in the system. It is the responsibility of the monitoring process to observe events and provide messages indicating the occurrence of QoS contract violations. There are two approaches to monitoring, *intrusive* and *non-intrusive*. Intrusive monitoring means that the monitoring process takes periodic samples of resource availability and utilisation; this in turn means that resources are consumed by the monitoring process itself, an overhead which is sometimes unacceptable. Alternatively, monitoring processes can rely upon indications of events which are not within the bounds of an agreed QoS contract. For instance, a monitoring process may receive indications from a media decoder concerning the dropping of video frames. This approach means that the monitored data set is much smaller and often aperiodic, however, resources are only consumed by the monitoring process in the event of QoS degradation or failure (some researchers may argue that this would cause deadlocks since a system should not consume more resources at a point of QoS failure).

Many traditional QoS-based adaptive systems use indications of QoS failure to initiate adaptation actions. As a consequence any resource management and adaptation which is carried out by the system is, more often than not, readily noticeable to the end-user. To avoid such disjunction, it is suggested that we should attempt to predict the need for adaptation. Some sources of monitoring, such as the CPU usage of a video decoder, are often unpredictable[1]. You can see from the results in figure 4 that the utilisation of resources is often application and implementation dependent. The resource utilisation from the playback of a local

[1] The figures were taken from PerfMon on Windows NT 4.0 whilst decoding videos of comparable content through DirectShow 2.0 software decoders.

MPEG video is relatively periodic, however the playback of an Indeo video which can be considered to be a similar application, uses CPU resources much more sporadically. Furthermore, even if resource utilisation patterns can be identified, they are often too fine grained to be useful. Nevertheless, prediction techniques are useful for coarser grained trends in resource utilisation and/or QoS characteristics. In such cases we can introduce simple statistical prediction algorithms, such as extrapolation and regression, to make an estimation of the probability of a failure occurring. Prediction techniques do depend upon the resource being monitored and the process[2] variants which are causing the system to degrade. Variants are either *system-driven*, such as caching techniques, or *user-driven*, such as a new selection of service. User-driven variants are often stochastic processes (meaning that they are random and have a mean of zero), and are therefore very difficult to predict in the short term. However, system-driven variants often result in a relatively predictable pattern, allowing statistical prediction algorithms to be used to extrapolate future resource variations and hence enable adaptations to be employed before the point of failure.

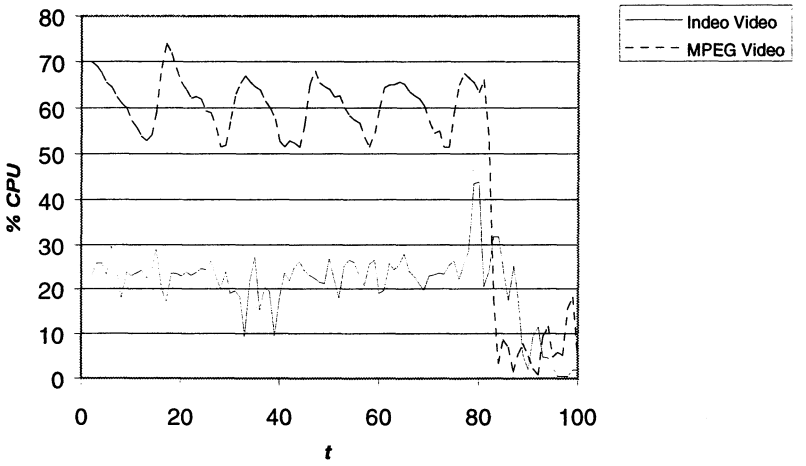

Figure 4 Example CPU Utilisations

3.2 QoS Requirements and Benefit Functions

The objective of any general purpose operating system is to share resources across multiple applications, in a fair and efficient manner. In doing so, the ultimate goal of the system is to fulfil the end-user's requirements, whether that be displaying video without interruption or maintaining a data backup in the event of a system crash. End-user requirements can be defined as *subjective* or *objective*. An example of a subjective end-user requirement is the impairment of transmitted

[2] In this context the term process is used to denote the generation of monitoring information.

video, often used in subjective testing in the engineering community. It is possible to use such perceptual QoS metrics to help optimise multimedia communication services (Verscheure, 1996).

Alternatively requirements can be classified as objective. These requirements are directly associated with a particular QoS metric and hence are more easily described and specified; examples include frame rate and network bandwidth. For this reason, traditional QoS-based resource management systems tend to concentrate on the management of objective requirements. However, there does exist a direct relationship between subjective and objective QoS. Furthermore, this relationship is particularly important in defining, and prioritising, which objective QoS requirements contribute to the overall goal of the system, satisfying the end-user. The relationship between the two can be quantified as a functional expression, known as a *benefit function*, which is a technique originally proposed by (Davis, 1994). The generalised abstraction allows the specification of arbitrary objectives (or subjective QoS requirements), and their relation to resource utilisation and/or objective QoS requirements. In turn, the function can be used by a resource manager to determine which adaptation processes are most beneficial to the end-user. An example of a use of a benefit function is in describing the relationship between frame jitter and frame size and overall subjective quality of service. From the video benefit function shown in figure 5 (Davis, 1994), it is apparent that once finite thresholds of frame jitter and frame rate are reached (0.2 and 0.7 respectively) the benefit of using more resources to increase the resulting subjective QoS is negligible.

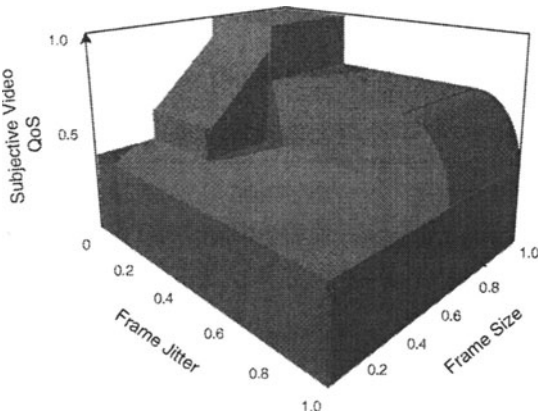

Figure 5 Video Benefit Function

1.1 Adaptation Mechanisms

To structure our model, we now define the mechanisms required to support QoS adaptation in a distributed environment. Many resource fluctuations in a distributed

system can be handled implicitly by the processing entities themselves. For example, a video decoder which receives a burst of frames and cannot process these before the next frames are expected may decide to drop a portion of the frames from the burst. In such a case, because the adaptation is very fine grained, the likelihood of the adaptation process being noticeable to the end user is small. However, more sizeable fluctuations may become apparent through resource monitoring and prediction as previously discussed, and given that a system monitoring process has indicated that QoS degradation is occurring, or QoS failure is imminent, ideally the system should adapt its resource utilisation in an attempt to maintain the end-user's level of service. As discussed in section 2, the process of adaptation, particularly in a distributed environment, involves addressing the balance of resources between the clients, servers and the network connections; the role of adaptation processes is to initiate such a balancing.

Table 2 Adaptation Mechanisms

Mechanism	*Technique*	*Resource Usage Shift*
Resource Control	Static Rate Shaping	client/network → null
	Dynamic rate shaping	client/network → server
	Cache Optimisation	server → server
	Priority Adjustment	client_a → client_b
	Scaling	network → server
	Network Service Control	client/server → network
		network → client/server
End-to-end Reconfiguration	Dual Codec Insertion	network → client/server
	Client Side Coder Insertion	network/server → client
	Server Hand-off	server_a → server_b
	Network 'swap in'	network_a → network_b
	Service provider fault action	network_a → network_a
Explicit Change of Service	Audio channel drop	client/network/server → null
	Video Adjustment	client/network/server → null

We identify three classes of adaptation mechanisms: *resource control*, making fine grained adjustments to individual resources in the distributed system; *reconfiguration*, altering the topology of the end-to-end processing; and *change of service*, allowing the user to prioritise services and adjust as necessary. The majority of adaptation mechanisms fit into one of these classes. Each mechanism has a certain granularity, and each may affect the resource distribution in a slightly different manner. Table 2 offers some examples of adaptation mechanisms and indicates the resulting shift of distribution in resource utilisation. The actual choice

of adaptation mechanism in response to received monitoring indications, is dictated by a set of system-wide *adaptation policies*. An adaptation policy describes, in conjunction with the applications resource capacity regions, what adaptation processes (control, reconfiguration or change of service) should be executed in response to various QoS scenarios. Furthermore, policies can be used in conjunction with the previously discussed benefit functions, allowing the prioritisation of adaptation mechanisms. Policies are applicable from end-to-end and are associated with any of the core distributed resources and their processing configuration. We suggest that the specification of policies should employ open interfacing techniques, aiding extensibility and readily understood programming level abstractions.

4 IMPLEMENTATION PERSPECTIVES

This section is concerned with the incorporation of the discussed QoS adaptation techniques into the Distributed Resource Management Architecture (DRMA) currently being developed by Lancaster University and BT Labs (Waddington, 1997). The DRMA platform focuses on the deployment of distributed multimedia applications over multi-service ATM networks and offers a framework for the hierarchical management of end-to-end resources. In addition, the implementation exploits open interfacing and distributed object techniques to aid scalability, flexibility and extensibility.

4.1 Adaptive Bindings

Within the DRMA, adaptation, monitoring and control processes are all engineered as a set of hierarchically structured *binding objects* (see figure 3). A binding object is a particular class of component which is used to abstract the functionality of a 'binding' between one or more source and sink components; a binding represents any form of communication (in the network or in the end-system) between these components. The combination of binding objects and other processing components is used to form the distributed application. Because the binding objects are hierarchically distributed, the adaptation and monitoring mechanisms also become distributed, thus avoiding the problem of centralised control and management. Each binding object maintains a set of interaction interfaces which describe the level of service offered through the binding communications, and furthermore define the point-to-point QoS constraints. The role of the binding object is to maintain the desired QoS, as specified by its interfaces, and in doing so it may, if required, employ monitoring and adaptation mechanisms. More detail on distributed binding objects is given in (Waddington, 1997).

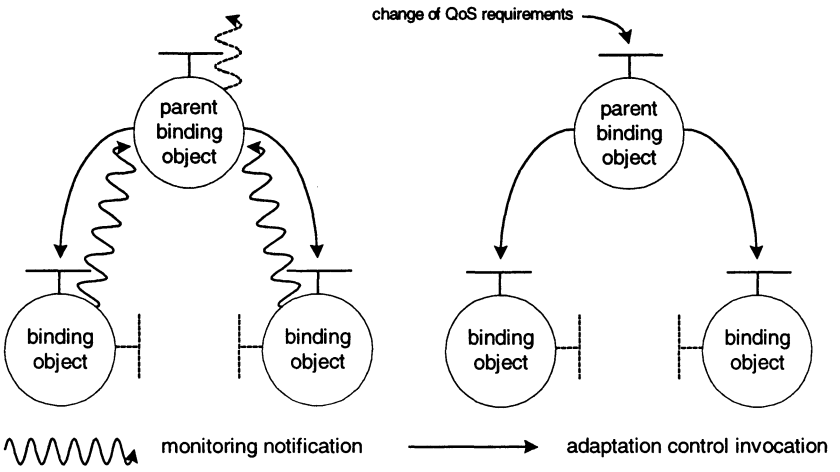

Figure 6 Explicit Hierarchical Adaptation

At the lowest level of the binding hierarchy, atomic processing components are linked together to form an end-to-end processing chain. Because adaptation and control at this level are very fine grained, the adaptation processes are both implicit (autonomous control adjustments) and explicit (control adjustments from parent bindings). As a consequence, care must be taken to avoid implementing adaptation policies which conflict or which may result in continuous counteractions. Explicit adaptation actions are usually triggered as a result of either degradation notifications or higher level changes in service requirements (see figure 6). In the former scenario, indications of QoS failure or degradation are passed from the child to the parent bindings. Adaptation policies are then parsed to determine what actions should be taken, and then any necessary adaptation processes are executed by the local binding object. If the binding object is unable to correct the degradation through adaptation (by a lack of suitable adaptation policies), or attempts at adaptation have failed, then the QoS notification is forwarded to the parent binding object (all notifications are passed through clearly defined QoS-feedback object interfaces). This process is continued until either successful adaptation actions are executed, or in the event of un-correctable failure the end-user is notified.

Example Adaptive Binding

As previously discussed in section 2.1, coarse grained adaptations occur within the higher level binding objects, such as the service binding. We now describe an example adaptive binding which is used to provision a multimedia streaming service across multi-domain ATM network. The binding hierarchy, as illustrated in figure 7, is composed of a service binding which delegates the end-to-end communications to concatenated end-system and network bindings.

During the initial setup, the service binding is responsible for mapping application level QoS requirements onto end-system and network level requirements (a process often referred to as QoS mapping). It is likely that many such mappings exist for a particular end-to-end service. Nevertheless, in determining the choice of end-to-end resource, the service binding object must make consideration of current resource availability (determined through monitoring) and their cost, a metric which is particularly relevant to network resources. Once a mapping has been determined the binding objects are instantiated and their QoS requirements exchanged. The next phase of the setup involves each individual binding object carrying out further QoS mapping to determine what resources are required to maintain its previously agreed point-to-point requirements. If there are insufficient resources, then the binding object must indicate an admission failure to the parent service binding. In our prototype implementation, the end-system binding uses statically defined mapping information to create a chain of end-system processing components (such as decoders and renderers) which meet the QoS requirements previously requested. In admitting the components, the end-system binding must also ensure that there are approximately (because we are using loose reservation) sufficient available CPU, memory, and other end-system resources. The setup of the network binding objects is slightly different. The network binding must employ some form of connection setup and routing, to establish a point-to-point QoS constrained network service. In our prototype implementation, we have used the ATM forum's UNI/NNI based signalling to create a guaranteed service to a Winsock2/AAL5 protocol stack.

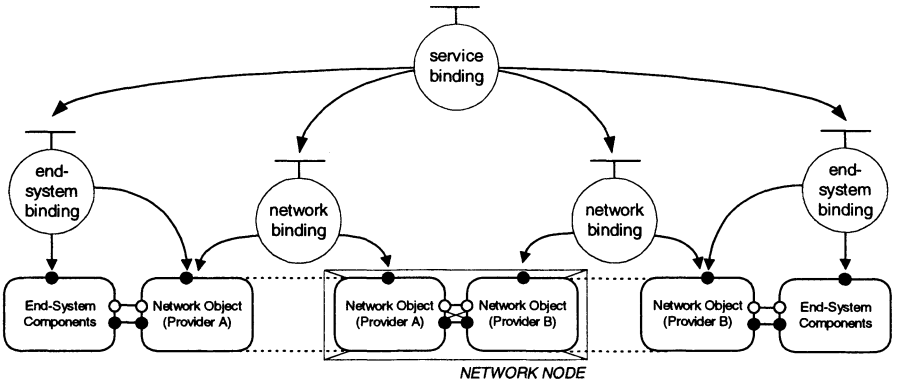

Figure 7 Example Adaptive Binding

During the lifetime of the service binding, their are potentially changes in the state of end-to-end resources which may in turn require QoS adaptation. For instance, one likely scenario is that the level of network service that was initially reserved is not sufficient for the application's streaming requirements (maybe video frames are being lost). Such mismatches in reserved or available resources, and the actual required resources, are indicated through QoS monitoring fed back from the

network objects (see section 4.1). On receipt of QoS degradation notifications, it becomes the responsibility of the service binding to initiate QoS adaptation. The choice of adaptation is made through a finite state machine representing the originally discussed resource capacity region. This is used to determines which adaptation policies are valid for the given level of service. If the binding is unable to make any suitable adaptation, then the component must indicate QoS failure to the application. Within this context likely adaptation scenarios would be adjustments in the network QoS (maybe through the setting up of a new ATM connection and carrying out a hot swap) or alternatively the incorporation of compression components in the end-systems. In some cases adaptation actions may take the form of reconfiguration. For example, consider the scenario where the initially chosen network service provider can only support a limited 25Mbps connection and a requirement of 26Mbps becomes evident. In this case the service binding object may choose to use an alternative network service provider and release the previously allocated network resources. Finally, in parallel with the previously discussed coarse-grained adaptations, the system is likely to experience various fluctuations in low level QoS, which are counter-acted through intrinsic fine grained adaptation actions.

5 CONCLUSION

We have proposed a general model for the support of QoS-based adaptation and resource management in distributed multimedia systems. Our perspective is across the complete end-to-end co-ordination and control of network and end-system resources, supporting end-to-end QoS constrained processing and communications. The general model of QoS adaptation incorporates the following principles:

- QoS provisioning through distributed resource management;
- Hierarchical approach to QoS management, leading to scalability;
- Combination of 'loose' resource scheduling and dynamic adaptation policies;
- Adaptation through both monitoring and prediction;
- Indications of end-user requirements through functional modelling of benefits.

We have also made progress towards the implementation of our proposed QoS adaptation mechanisms, using QoS adaptation techniques in the development of the prototype Distributed Resource Management Architecture (DRMA); DRMA provides a distributed platform for the development and deployment of QoS-constrained continuous media applications. The system, based on Windows NT, uses distributed object programming methods and hierarchical QoS management to support adaptive bindings which transparently encapsulate the proposed resource monitoring and adaptation mechanisms. Our future work is directed towards a fuller implementation of the Distributed Resource Management Architecture together with further support for QoS management and QoS adaptation, in both the network and end-system bindings.

6 ACKNOWLEDGEMENTS

We would like to acknowledge the kind support of BT Labs in funding this research under their Management of Multi-service Networks University Research Initiative (BT-URI).

7 REFERENCES

Anderson, D.P. (1993) Metascheduling for Continuous Media, ACM Transactions on Computer Systems, Vol. 11, No. 3, pp. 226-252.

Bolot, J.C. and Turletti, T. (1996) Adaptive Error Control for Packet Video in the Internet, IEEE Signal Processing Society, Proceedings of the International Conference on Image Processing (ICIP), Lausanne.

Campbell, A. and Coulson, G. and Hutchison, D. (1994) A Quality of Service Architecture, ACM Computer Communications Review.

Chatterjee, S., Sydir, J. and Sabata, B. (1997) Modelling Applications for Adaptive QoS-based Resource Management", Proceedings of the 2nd IEEE High Assurance Systems Engineering Workshop, Bethesda, Maryland.

Dang Tran, F. and Perebaskine, V. (1995) TORBoyau: Architecture and Implementation, General Leclerc, Issy-les-Moulineaux, France.

Davis, M. and Downing, A. (1994) Adaptable System Resource Management for Soft Real-Time Systems, Symposium on Command and Control Research and Decision Aids, Monterey, California.

Degermark, M., Kohler, T., Pink, S. and Schelen, O. (1995) Advance Reservations for Predictive Service, Proceedings of 5th International Workshop on Network and Operating System Support for Digital Audio and Video, Durham, New Hampshire.

Edwards, C., Hutchison, D. and Waddington, D. (1998) Open Interface Support for Heterogeneous Network Services, to be presented at the Third European Conference on Multimedia Applications, Services and Techniques (ECMAST'98), Berlin.

Huard, J., Inoue, I., Lazar, A. and Yamanaka, H. (1996) Meeting QoS Guarantees by End-to-end QoS Monitoring and Adaptation, Workshop on Multimedia and Collaborative Environments of the Fifth IEEE International Symposium On High Performance Distributed Computing, Syracuse, NY.

Hyman, J., Lazar, A. and Pacifici, G. (1991) Real-time Scheduling with Quality of Service Constraints, IEEE Journal on Selected Areas in Communications, Vol. 9, pp. 1052-1063.

ISO/IEC JTC 1/SC21 (1997) "Quality of Service in ODP – Attachment 1"

Lazar, A., Ngoh, L. and Sahai, A. (1995) Multimedia networking abstractions with quality of service guarantees, Proceedings of the SPIE Conference on Multimedia Computing and Networking, San Jose, CA.

Nahrstedt, K., Hossain, A. and Kang, S. (1995) Probe-based Algorithm for QoS Specification and Adaptation, University of Illinois, QoS Workshop, Paris.

Ott, M., Michelitsch, G., Reininger, D. and Welling, G. (1997) An Architecture for Adaptive QoS and its Application to Multimedia Systems Design, to appear in Special Issue of Computer Communications on Building Quality of Service into Distributed Systems, C&C Research Laboratories, NEC, USA.

Verscheure, O. and Hubaux, J. (1996) Perceptual Video Quality and Activity Metrics: Optimization of Video Service Based on MPEG-2 Encoding, Springer Series LNCS 1185, Proceedings of 3rd International COST237 Workshop, Barcelona, Spain.

Waddington, D., Edwards, C. and Hutchison, D. (1997) Resource Management for Distributed Multimedia Applications, Second European Conference on Multimedia Applications, Services and Techniques (ECMAST '97), Milan.

Waddington, D. and Coulson, G. (1997) A Distributed Multimedia Component Architecture, Proceedings of the 1st International Workshop on Enterprise Distributed Object Computing, Gold Coast, Australia.

Wolf, L., Delgrossi, W. L., Steinmetz, R., Schaller, S. and Wittig, H. (1995) Issues of Reserving Resources in Advance, Proc. 5th Intl. Workshop on Network and Operating System Support for Digital Audio and Video, Durham, New Hampshire.

8 BIOGRAPHY

Daniel Waddington is a final year Ph.D. student, with the Distributed Multimedia Research Group. He is currently working as a Research Assistant on the British Telecom University Research Initiative (BT-URI) project. On the BT-URI, Daniel is looking at issues of end-to-end QoS management and a framework for service provision. His primary Ph.D. research interests are Distributed Object Computing and its support for distributed multimedia services.

David Hutchison is Professor of Computing at Lancaster University and has worked in the areas of computer communications and distributed systems for the past 15 years. He has completed many UK and European funded research contracts and published over 100 papers as well as writing and editing books on these areas. He has just finished a years sabbatical leave as a visiting academic at HP Labs in Bristol, UK, at EPFL in Lausanne, Switzerland, and at BT Labs in Ipswich, UK.