A FLEXIBLE APPROACH TO TRANSACTION MANAGEMENT IN DISTRIBUTED, MULTIMEDIA SYSTEMS

M. BEATRIZ F. DE TOLEDO, GORDON S. BLAIR*

This paper discusses the potential impact of multimedia on one classic distributed systems problem – the provision of transaction mechanisms to support fault tolerance and concurrency control. Several features of distributed multimedia sytems are highlighted: the real–time requirements of continuous media types, the cooperative styles of working and the engineering requirements across the range of media types. It is argued that, given these demands, multimedia will require a more flexible approach to transaction management. A framework for flexible transaction management is described. This framework is based on the ANSA architecture and allows applications to specify particular policies for concurrency control, recovery and real–time behaviour.

1. Introduction

The emergence of multimedia has provided a great stimulus to distributed systems research. Many of the traditional problems of distributed systems are made considerably more difficult with the introduction of multimedia services. This paper discusses the potential impact of multimedia on one classic distributed systems problem, namely the provision of transaction mechanisms to support fault tolerance and concurrency control.

The traditional approach to transactions is to provide specific mechanisms to deal with the issues of synchronization and recovery. In such systems, the algorithms for synchronization and recovery tend to be fixed by the systems and hence they suffer from a lack of flexibility. However, as transactions started to be used in other environments, a revision in this traditional model became necessary. In particular, for multimedia environments where applications can have widely different requirements, a more flexible model is required.

*Distributed Multimedia Research Group, Computing Department, Lancaster University, Bailrigg Lancaster LA1 4YR U. K.

A new flexible model for transaction management is proposed; this approach gives applications more control on transaction management enabling them to state policies for services such as synchronization and recovery. The model also provides facilities to deal with real–time transactions (a vital concern in multimedia computing).

The paper is structured as follows. Section 2 traces the emergence of distributed multimedia computing. This is followed by section 3 with a commentary on the various styles of multimedia information. Section 4 then discusses the impact of multimedia on the field transaction management. Section 5 proposes a new model for transaction management with the aim of achieving the flexibility required by multimedia applications. A simple example of use of transaction services is presented in section 6. Finally, section 7 makes some concluding remarks.

2. The Emergence of Distributed Multimedia Computing

Distributed systems have been one of the major growth areas in computing over the past decade. This interest can be attributed to two factors:

Enterprise Demands

Most medium and large scale organisations are distributed, with management, equipment, skilled personnel and administration spread across a number of geographically dispersed sites. Therefore, they have a great interest in techniques to integrate their computing and communications infrastructures and allow network wide access to company resources.

Technology Developments

Both computer and communications technology have advanced considerably over the past decade. It is now economically feasible to have a large number of workstations spread across a large organisation. In addition, it is possible to connect such computer facilities with high speed local area and wide area digital communications systems.

As a consequence, there has been considerable research and development in distributed systems both in industry and in academia. Recently however the new field of distributed, multimedia systems has emerged. Again, this can be seen as a natural development resulting from demands from industry/commerce and new technological developments.

From above, there is a realisation that facilities such as electronic mail and remote access to computing services are not enough. Industry and

commerce are looking for more sophisticated communications services such as voice, image and video in addition to traditional text based services. From below, technological developments are starting to make distributed, multimedia systems feasible. For example, local area network technologies such as FDDI (Heywood and Greenfiel, 1990) and the fast Cambridge Ring are now being developed. In addition, wide area Integrated Service Digital Networks (ISDN) (Ludwig, 1989) are becoming available.

3. What is Multimedia?

Multimedia comprises information such as audio and image, in addition to traditional data such as text. A fully fledged system is then one which can store, transmit and manipulate the various media types. In more detail, the following media types can be identified:

- text
- raster graphics
- vector graphics
- audio
- video (moving raster)
- animation (moving vector)

The latter three categories are particulary interesting and are generally refered to collectively as *continuous media* types to reflect the fact that they can continue over a period of time (Anderson *et al.*, 1990). Continuous media therefore make specific real–time demands of the underlying system infrastructure. For example, a network must guarantee to deliver audio at a particular rate for the quality to be acceptable.

One of the key challenges of multimedia is to achieve *integration* of the various media types. This is particularly difficult given the wide range of characteristics of each of the media types listed above. This makes the problem of managing multimedia information in a distributed system extremely difficult. This is our prime motivation in seeking more flexible management strategies for distributed multimedia computing.

4. The Impact of Multimedia on Transactions

Transaction mechanisms have traditionally been concerned with the maintenance of the consistency of a group of objects in spite of concurrent access and the possibility of partial failure of parts of the system. A wide range of techniques (two–phase locking, timestamps, logs, versions, etc) have been developed to assist in this task. The original model provides integrated,

fixed mechanisms to treat concurrency control and recovery and thus guarantees the properties of serializability and atomicity. For the traditional model refer to (Bernstein and Goodman, 1981; Ceri and Pelagatti, 1984; Eswaran *et. al.*, 1976).

More recently however there has been greater interest in less rigid techniques which are more geared towards the semantics of the application domain and hence tailored towards the cooperative nature of the task. This is having a major impact on the design of transaction mechanisms with techniques such as non–serializable transactions starting to appear (Skarra and Zdonik, 1989). Multimedia systems will require other alternative designs for transaction services.

The main factors which influence the design of a transaction mechanisms are discussed below:

i) real–time requirements of multimedia

It is common for multimedia services to have stringent real–time requirements. This is particularly true for continuous media types (as discussed above). It is already recognised that real–time requirements can severely affect transaction mechanisms. For example, transaction aborts or delays through locking can prejudice the required real–time behaviour of an interaction. Similarly, it is often not possible to recover in the conventional sense from a real–time transaction because of the real world interactions involved. Transaction mechanisms are therefore required which cater for a variety of real–time constraints in a distributed environment.

ii) cooperative modes of work

Computer Supported Cooperative Work (CSCW) is emerging as a major topic in distributed, multimedia computing (Rodden and Blair, 1991). In CSCW, emphasis is placed on support for the cooperative nature of the task rather than on the control of competitive access to resources (Bancilhon *et. al.*, 1985; Bannon and Shmidt, 1989; Fernandez and Zdonik, 1989). This inevitably effects the design of transaction mechanisms. Hard locks are now seen as inappropriate for many classes of interaction; alternative synchronisation mechanisms based on soft locks or version histories are preferred. Such developments will lead to synchronisation mechanisms more closely geared toward 'floor passing strategies' or 'joint editing metaphors' as found in CSCW applications. It is important for developers of distributed, multimedia infrastructures to be aware of these trends.

iii) dealing with variety

The key characteristic of multimedia computing is that a range of services with widely differing characteristic can co–exist in a single system. A typical multimedia system is a very rich environment with many dimensions. For example, a system may have a high performance audio–visual store, a range of sound and vision devices and a spectrum of communications protocols to carry the various forms of multimedia traffic. In contrast, a traditional distributed system has a fairly small number of storage services, communication services and devices. This variation in services has major consequences for the management of a distributed system.

Distributed systems in the future will have to cater for at least some aspects of multimedia computing. Therefore it is important to address the issues raised by the above problems. Inevitably this will mean a major change to the design of transactions. In our estimation, more flexible management should replace the rigidity of traditional transactions. More specifically, it will be necessary to tailor particular transaction mechanisms for the requirements of given classes of application. As highlighted above, it is also necessary to cater for real–time requirements within the transaction framework.

5. Towards Flexible Transactions

5.1. Background

The framework for transaction management was designed in the context of related research in the Distributed Multimedia Research Group, Lancaster University. The main goal of this research is to develop a distributed systems infrastructure to support interactive multimedia applications.

The approach taken in the research is to extend an existing distributed systems platform, namely the ANSA testbench, to cater for multimedia. The ANSA testbench is an experimental system designed for Open Distributed Processing (ODP). To cater for heterogeneity, the ANSA testbench layers a platform on top of the host environment providing a unified abstraction of the underlying system.

The testbench is based on the client–server model of distributed processing. More specifically, all services are treated uniformly as objects accessible through one or more interfaces. An interface describes the operations defined on an object, the parameters and results of operations, and a range of properties associated with the object, e. g. levels of distribution transparency. The testbench also provides a system *trader* which acts as a registry

of available services. Clients wishing to access a service must *import* an object interface by specifying a set of requirements in terms of operations and properties. The role of the trader is to choose a suitable candidate among the registered services. Once an interface has been selected, the system must bind the requester object to the object implementing the service and thus allowing operations to be invoked. In the ANSA testbench, this binding is provided by the remote procedure call protocol, REX.

Research at Lancaster has made a number of extensions to the ANSA testbench to support multimedia (Blair *et al.*, 1991). Firstly, it was necessary to implement key multimedia services, requiring real–time response, on a separate *multimedia network interface* (Ball *et al.*, 1990). Secondly, the basic architecture was extended to include *streams* as abstractions for high speed isochronous protocols and a more *flexible approach to trading* to accommodate multimedia services (MaCartney and Blair, 1990). With respect to distributed management, it was necessary to provide a more flexible framework suitable for applications with different requirements. The main features of this management framework are described in the following section.

5.2. A Model for Flexible Transaction Management

5.2.1. General Approach

The main features of the transaction model are described below:

Support for Real–Time

Traditionally, transactions deal with two separate but closely related issues, namely synchronization and recovery. This is extended in our model to include real–time. The issue of real–time transactions is vast and complex. We focus on one particular aspect of real–time, i.e. the guarantee of resources to achieve the necessary performance. This is seen as analogous to the guarantee of serializability to achieve correct concurrent behaviour. Thus, the model supports operations to gain and release resources to mirror operations to gain and release locks.

Separation of Mechanism and Policy

The problem with existing transaction mechanisms is that management responsibility is completely delegated to the underlying system. This means that the system is responsible for making policy decisions. The key to our approach is to separate out policy from the mechanisms required to support

the policy. In particular, objects support mechanisms to achieve concurrency control, recovery and real–time guarantees. The exact policy stating how to use these mechanisms is provided by higher levels of authority.

Co–existence of Different Approaches

To increase the level of flexibility, objects can use a variety of mechanisms to achieve concurrency control, recovery and real–time guarantees. For example, it is possible for some objects to use locking algorithms whereas others adopt timestamp approaches. Similarly, recovery techniques based on logging can co–exist with techniques based on shadows. In order to achieve this generalisation, all objects must provide a generic interface mapping on to the range of possible mechanisms. Such an interface has been defined for the three areas, e. g. for locking the operations are request_access_permission and release_access_permission. These operations can clearly be implemented in a variety of ways.

5.2.2. Incorporating Flexible Transactions in ANSA

In order to support generic objects, the ANSA notion of a *factory* was extended. Factories in ANSA (ANSA, 1990) are used to create instances of objects of a given type. To cater for different choices of mechanisms, factories are parameterised. The same factory is now able to create objects with different characteristics depending on the parameters given. A parameter expresses a creation policy for a service and thus determines which mechanisms implementing this service are to be linked to the created object. For example, the support environment may provide locking and optimistic concurrency control mechanisms; according to the specified policy (pessimistic or optimistic concurrency control), one of the mechanisms is linked to the created object.

Parameterised factories support the creation of individual objects with a variety of characteristics. This is one level of flexibility provided by the model. A second level of flexibility is provided by the approach to transaction management (across a group of objects). Applications have a choice in the way they access objects. In particular, they can opt for either *transparency* in dealing with concurrency, recovery and real–time issues or *non–transparency.*

For non–transparency, the application itself is responsible for managing the required objects. The application must therefore invoke the various operations for concurrency, recovery and real–time directly. For example, the application would have to manage locks correctly.

Non–transparency imposes a considerable burden on applications. Therefore, application writers may decide to opt for transparency. In this case, a *management policy* may also be specified. This can be used, for example, to state if the application requires serializability or recovery. It is important that this policy statement is compatible with the creation policies of the objects participating in the transaction.

If transparency is selected, a *transaction manager* is created by a factory (as any object in the system). In this case, the parameters specified at creation time express the *management policies* for the transaction and are meant to force constraints on the objects involved in the transaction. Every time a new object joins a transaction, a consistency check must be made between the characteristics of an object and the policy of the transaction; the application is only allowed to proceed if there is no incompatibility. The interface for the transaction manager object consists of operations for transaction control such as begin transaction, end transaction, abort, checkpoint and also reserve a group of resources if real–time control is necessary. The transaction manager will be responsible for invoking the appropriate operations on underlying objects, thus removing this management burden from applications.

6. Transaction Services – an example of use

Applications rely on services exported to the trader for transaction management. As mentioned in previous sections, services are related with concurrency control, storage and real–time. In addition to these services, factories for creating reliable objects and transaction managers are provided.

If an application is interested in transparent transaction management, it must import the Transaction Factory for creating a transaction manager. The import is done through a prepc (ANSA, 1990) statement:

```
! itf <- ANSAImport ("TransFactory", context, constraints)
```

and then a transaction manager can be created by invocating the Transaction Factory create operation. The create operation will return an interface reference to the created transaction manager.

```
! {itm} <- itf$create (policies for transaction management)
```

Once the transaction manager is created the transaction is initiated by the begin operation and terminated by either end (if successful – objects are

updated) or abort (updates are discarded). Within the transaction, objects may be created and operations invocated on them. The execute operation tests conflicts (in case of pessimistic concurrency control) and invocates the specified operation on a given object.

! {} <– itm$begin()

! irf <– ANSAImport ("ReliableFactory", context, constrains)

! $\{o_1\}$ <– irf$create (*policies for services*, $type_1$)

! $\{o_2\}$ <– irf$create (*policies for services*, $type_2$)

! $\{result_1\}$ <– itm$execute (op_{o1}, o_1)

! $\{result_2\}$ <– itm$execute (op_{o2}, o_2)

! {} <– itm$end()

In case an application does not choose transaction management transparency; it will have to invoke operations for concurrency control and storage itself.

7. Concluding Remarks

This paper discusses the characteristics and requirements of multimedia systems which affect transaction management. It is argued that these requirements demand a revision of the traditional model in order to achieve more flexibility.

With this aim, a new model for transaction management is proposed. In this model, applications have more control over transaction management through the specification of policies for concurrency control and recovery. In addition, real–time issues are dealt with under the umbrella of transaction management.

In brief, the design of the framework for transaction management is mainly concerned with achieving more flexibility through the separation of policies and mechanisms, the independence of the supported objects, the provision of several mechanisms in the implementation of objects and also with the possibility of choice between transparency or non–transparency for transaction management.

Work on flexible transaction management is continuing with an examination of the policies required in the model and also the constraints a transaction can impose on its participating objects. The generic interfaces for objects and transaction managers have already being defined. Implementation work is about to be started. In particular, a prototype is being developed on the extended ANSA platform developed at Lancaster.

References

Anderson D.P., Tzou S.Y., Wahbe F., Govindan R. and Andrews M. (1990): *Support for continuous media in the DASH system.*– Proc. of the 10th Int. Conf. on DISTRIBUTED COMPUTING SYSTEMS, Paris.

ANSA (1989): *ANSA Reference Manual, Release 01.00.*– APM Ltd., Cambridge, U. K., March.

ANSA (1990): *DPL Programmer's Manual.*– APM Ltd., Cambridge, U. K.

Ball F., Hutchison D., Scott A. and Shepherd D. (1990): *A Multimedia Network Interface (MNI).*– 3rd IEEE COMSOC Int. Multimedia Workshop, Bordeaux, France.

Bancilhon F., Kim W. and Korth H. (1985): *A model of CAD Transactions.*– Proc. of the 11th Int. Conf. on VLDB, Stockholm.

Bannon L. and Schmidt K. (1989): *CSCW: Four characters in search of context.*– Proc. of EC–CSCW, Gatwick Hilton, 1989.

Bernstein P.A. and Goodman N. (1981): *Concurrency control in distributed database systems.*– ACM Computer Surveys, v.13, No.12.

Blair G.B., Coulson G., Davies N. and Williams N. (1991): *Incorporating multimedia in distributed open systems.*– Proc. of EurOpen'91, Tromsoe, Norway.

Ceri S. and Pelagatti G. (1984): *Distributed Databases–Principles and Systems.*– New York: McGraw Hill.

Ellis C.A., Gibbs S.J. and Rein G.L. (1991): *Groupware– Some issues and experiences.*– Comm. ACM, v.34, No.1.

Eswaran K., Gray J., Lorie R. and Traiger I. (1976): *The notion of consistency and predicate locks in a database system.*– Comm. ACM, v.19, No.11.

Fernandez M. and Zdonik S. (1989): *Transaction groups: A model for controlling cooperative transactions.*– Workshop on Persistent Object Systems: Their Design, Implementation and Use.

Heywoood P. and Greenfiel D. (1990): *FDDI: Just Say Not Yet.*– Data Communications International.

Ludwig L.F. (1989): *Multimedia in ISDN and BISDN: A Paradigm shift driven by evolving user technology and applications.*– Bellcore Digest of Technical Information.

MaCartney A. and Blair G.S. (1990): *Flexible trading in distributed multimedia systems.*– Computer Networks and ISDN Systems, (submitted to print).

Rodden T. and Blair G.S. (1991): *CSCW and distributed systems: the problem of control.*– Proc. of the European Conf. on Computer Supported Cooperative Work (ECSCW'91), Amsterdam.

Skarra A. and Zdonik S. (1989): *Concurrency control and object-oriented databases.*– Object–Oriented Concepts, Databases and Applications, (Ed: Kim W., Lochovsky F.H.).– ACM Press Frontier Series, Addison–Wesley.