Multimedia Communications Session Management

Terrence P. McGarty

Abstract: Multimedia Communications involves itself with the communications of highly distributed multimedia data objects that require precise timing at and between multiple locations. This paper proposes a way to handle this level of communications through enhancements made at the Session Layer of the OSI protocol standard. The approach taken starts with a definition of multimedia data objects and then develops the required elements for the Session Layer. Detailed implementations are presented and discussions on their performance comparisons are discussed..

***Index Terms*—Session Management, OSI, Multimedia Communications, Protocol Management.**

I. Introduction

Multimedia Communications is a discipline that combines the ideas of the human senses, disparate storage and data structures, varying interfaces and complex communications systems. The basic concept of a multimedia environment has evolved from that of the single media data focused world of the computer specialist to the need to provide a fully integrated system for a human user to interact with using information stored on many different storage media. Multimedia consists of a matching of the three elements of the senses, the storage media and the interface devices.

It has been argued elsewhere (see McGarty, 17) that multimedia should not be confined to merely the storage of information of multiple storage devices. Rather, multimedia must include the senses and the interfaces as well. In fact, for the purpose of this paper we define multimedia as the confluence of storage, senses and interfaces. Specifically, multimedia relates to constructs of not only information storage but also information processing and communications. It encompasses all of the senses, although we currently only focus on the senses of sight, sound and touch. The definition that we take of multimedia in this paper is an expansive definition. It has been taken to provide a basis for the next step which is multimedia communications, which takes the multimedia paradigm and adds multiple human elements and as such transcends the prototypical computer communications view of the world.

Manuscript received _______________.

T.P. McGarty is with The Telmarc Group, Florham Park, NJ 07932 (tmcgarty@telmarc.com or mcgarty@mit.edu)

When we introduce the communications concepts, we do so in the context of having multiple users share in the use of the multimedia objects. Thus multimedia communications requires that multiple human users have sensory interfaces to multiple versions of complex objects stored on multiple storage media. In contrast to data communications in the computer domain, where humans are a secondary after thought, and optimization is made in accordance with the machine to machine connection, multimedia a communications is a human to many other human communications process that must fully integrate the end user into the environment. Multimedia communications thus generates a sense of conversationality, it is sustainable over longer periods, and it has an extreme fluidity of interaction.

Various authors have recently addressed the issue of multimedia communications with an architectural approach. (See Little and Ghafoor, Nicolaou, and Steinmetz). The current approaches focus on one of two extremes, either on broadband communications and the transport mechanism or on the multimedia storage aspects of the system design. Little and Ghafoor have attempted to integrate the presentation and data object side of the problem and have at a higher level, attempted to address the communications issues. Nicolaou has developed a communications architecture that follows the OSI standards but in attempting to introduce the multimedia issues has been forced to introduce several new constructs. Various other researchers in this area have focused on the lower protocol layers and have specifically been concerned with transport layer problems and below.

One of the major challenges to multimedia communications is that today there are broadband architectures that are developed that provide higher speed communications using direct extensions of the techniques developed in the data world of packet communications. Specifically such techniques as ATM and SMDS, as well as FDDI are direct offshoots of local area networks and packet technology. The fail to understand the paradigm that we are developing in this paper that relates to the structure of the multimedia object and the conversationality of multimedia communications.

In this paper, we concentrate on three issues in the area of multimedia communications; the data objects, the conversationality of the interaction and the overall communications architecture. We first note that the data structures in multimedia environments are dramatically different than those in normal computer data communications. Specifically, Mullender has shown that typical data file sizes that are transferred in a UNIX environment are on the order of 2K bits whereas in a multimedia environment the file size may average 100 Mbits. Secondly, a multimedia environment needs to handle real time data interaction such as that in real time voice and video. As is well known, such transport protocols as TCP/IP are not adequate from a delay perspective to support these types of data objects.

The conversationality aspect of the multimedia environment is key to effective communications. In this paper we focus on utilizing the Session layer from the OSI format for the delivery of the multi-user conversationality. Historically, the session layer (See Tannenbaum) has been relegated to a secondary position in the OSI hierarchy. In a multimedia environment, we show that the session functionality, refined and expanded, provides the essential integrating capability for conversationality.

The remaining communications services, at OSI layer 4 and below, become, at best, delimiting factors in the communications environment. In this paper we show that there are certain underlying performance factors of the lower four layers, that when combined control the overall end to end performance as viewed from the users perspective. As a major point in this paper, we argue that the standard approach to communications system design, from the physical layer and up is the wrong way to proceed for multimedia. Specifically, in a multimedia environment, one must, perforce of user acceptance, design the system from the top layers and down.

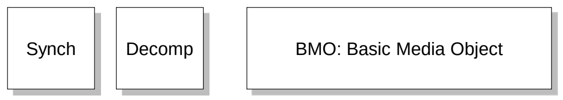
II. Multimedia Data Objects

In a more standard computer communications environment, the data objects have significant structure and they are frequently integrated into a system wide data base management system that ensures the overall integrity of the data structures. In a multimedia environment, the data elements are more complex, taking the form of video, voice, text, images and may be real time in nature or can be gathered from a stored environment. More importantly, the separate data objects may combined into more complex forms so that the users may want to create new objects by concatenating several simpler objects into a complex whole. Thus we can conceive of a set of three objects composed of an image, a voice annotation and a pointer motion annotating the voice annotation. The combination of all three of these can also be viewed as a single identifiable multimedia object.

Before commencing on the issues of communications, it is necessary to understand the data objects that are to be communicated. We can consider a multimedia data object to be composed of several related multimedia data objects which are a voice segment, an image and a pointer movement (e.g. mouse movement). As we have just described, these can be combined into a more complex object. We call the initial objects Simple Multimedia Objects (SMOs) and the combination of several a Compound Multimedia Object (CMO). In general a multimedia communications process involves one or multiple SMOs and possibly several CMOs.

The SMO contains two headers that are to be defined and a long data sting. The data string we call a Basic Multimedia Object (BMO). There may be two types of BMOs. The first type we call a segmented BMO or SG:BMO. It has a definite length in data bits and may result from either a stored data record or from a generated record that has a natural data length such as a single image screen or text record. We show the SMO in Figure 21..

Figure 2.1 SMO Structure



The second type of BMO is a streamed BMO, ST:BMO. This BMO has an a priori undetermined duration. Thus it may be a real time voice or video segment.

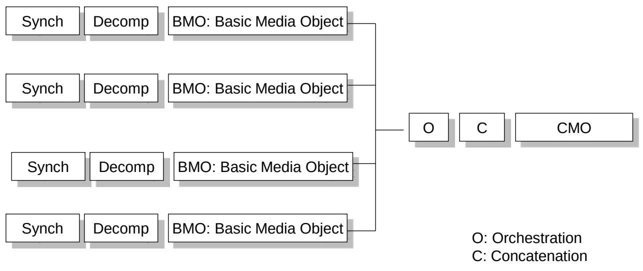
A simple multimedia object, SMO, is a BMO with two additional fields; a Synchronization field (Synch) and a Decomposition field (Decomp). Figure 2.1 depicts the SMO structure in detail. The Synch field details the inherent internal timing information relative to the BMO. For example it may contain the information on the sample rate, the sample density and the other internal temporal structure of the object. It will be a useful field in the overall end to end timing in the network.

The second field is called the Decomp field and it is used to characterize the logical and spatial structure of the data object. Thus it may contain the information on a text object as to where the paragraphs, sentences, or words are, or in an image object, where the parts of the image are located in the data field.

These fields are part of an overall architecture requirement finds it necessary to provide an "out-of-band" signaling scheme for the identification of object structure. The object structure is abstracted from the object itself and becomes an input element to the overall communications environment. Other schemes use in-band signaling which imbeds the signal information with the object in the data stream. This is generally an unacceptable approach for this type of environment.

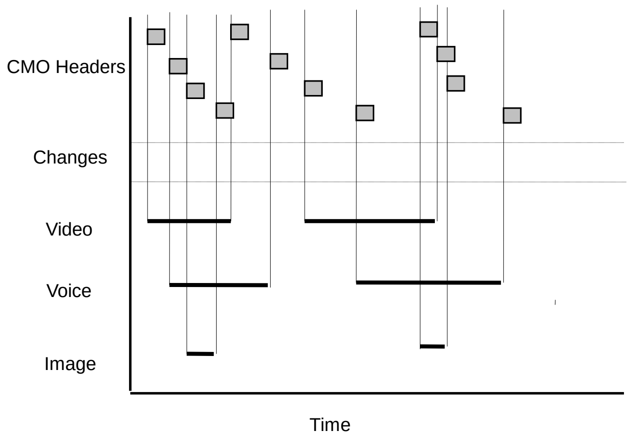
When we combine these objects together we can create a compound multimedia object. This is shown in Figure 2.2. A CMO has two headers, the Orchestration header and the Concatenation header. The Orchestration header describes the temporal relationship between the SMOs and ensures that they are not only individually synchronized but also they are jointly orchestrated. The orchestration concept has also been introduced by Nicolaou. In this paper we further extend the orchestration function beyond that of Nicolaou. The concatenation function provides a description of the logical and spatial relationships amongst the SMOs.

Figure 2.2 CMO Structure



These concepts have been further developed in McGarty[2] and there we have provided more detailed structure to the multimedia data objects. We can now add dynamics to this process and we show this in Figure 2.3. In this Figure we show first the real time display of video, voice, image, pointer and text. In the Figure we depict the time that these object are involved in the system dynamics. We then also plot the times that the CMO, the concatenation of all simultaneous objects, change in this system. In Figure 2.5 we depict 21 change element. Then we also show the CMO headers that are flowing in the network at each change interval. It is this dynamic process of data elements that must be controlled by the session layer to be discussed in the next session.

Figure 2.3 Temporal Interaction of CMOs



We can also expand the concept of a CMO as a data construct that is created and managed by multiple users at multiple locations. In this construct we have demonstrated that N users can create a CMO by entering multiple SMOs into the overall CMO structure.

The objectives of the communications system are thus focused on meeting the interaction between users who are communicating with CMOs. Specifically we must be able to perform the following tasks:

Allow any user to create an SMO and a CMO.

Allow any user or set of users to share, store, or modify a CMO.

Ensure that the user to user communications preserves the temporal, logical and spatial relationships between all CMOs at all users at all times.

Provide an environment to define, manage and monitor the overall activity.

Provide for an environment to monitor, manage and restore all services in the event of system failures or degradation.

We shall see in the next section that the session layer service address all of these requirements.

III. Session Layer Functions

The OSI layered communications architecture has evolved to manage and support the distributed communications environment across error prone communications channels. It is presented in detail in either Tannenbaum or Stallings. A great deal of effort has been spent on developing and implementing protocols to support these channel requirements. Layer 7 provides for the applications interface and generally support such applications as file, mail and directory. The requirements of a multimedia environment are best met by focusing on layer

5, the session layer whose overall function is to ensure the end to end integrity of the applications that are being supported.

Some authors (See Couloris and Dollimore or Mullender) indicate that the session function is merely to support virtual connections between pairs of processes. Mullender specifically deals with the session function in the context of the inter-process communications (IPC). In the context of the multimedia object requirements of the previous section, we can further extend the concept of the session service to provide for IPC functionality at the applications layer and specifically with regards to multimedia applications and their imbedded objects.

The services provided by the session layer fall into four categories:

Dialog Management: This function provides all of the users with the ability to control, on a local basis as well as global basis, the overall interaction in the session. Specifically, dialog management determines the protocol of who talks when and how this control of talking is passed from one user to another.

Activity Management: An activity can be defined as the totality of sequences of events that may be within a session or may encompass several sessions. From the applications perspective, the application can define a sequence of events called an activity and the session service will ensure that it will monitor and report back if the activity is completed or if it has been aborted that such is the fact.

For example, in a medical application, we can define an activity called "diagnosis" and it may consist of a multiple set of session between several consulting physicians. We define a beginning of the activity when the patient arrives for the first visit and the end when the primary physician writes the diagnosis. The session service will be responsible for ensuring that all patients have a "diagnosis".

Synchronization: We have seen that at the heart of a multimedia system is a multimedia data object. Each of the objects has its own synchronization or timing requirements and more importantly, a compound object has the orchestration requirement. The session service of synchronization must then ensure that the end to end timing between users and objects is maintained throughout.

Event Management: The monitoring of performance, isolation of problems, and restoration of service is a key element of the session service. Full end to end network management requires not only the management of transport and sub network, but requires that across all seven OSI layers, that overall end to and management be maintained (See McGarty and Ball).

Here we have shown the session entity which is effectively a session service server. The entity is accesses from above by a Session_Service Access Point (S_SAP). The session entities communicate through a Protocol Data Unit (PDU) that is passed along from location to location. Logically the session server sits atop the transport server at each location.

The servers are conceptually at a level above the transport level. We typically view the transport servers as communicating distributed processes that are locally resident in each of the transmitting entities. This then begs the question as to where does one place the session servers. Are they local and fully distributed, can they be centralized, and if so what is their relationship to the Transport servers. Before answering these questions, let us first review how the session services are accessed and how they are communicated.

Session services are accessed by the higher layer protocols by invoking session service primitives. These primitives can invoke a dialog function such as Token_Give. The application may make the call to the S_SAP and this request may be answered. There are typically four steps in such a request, and these are listed in Stallings who shows that the requests are made of the session server by entity one and are responded to by entity two. The model does no however say where the session server is nor even if it is a single centralized server, a shared distributed server, or a fully distributed server per entity design. We shall discuss some of the advantages of these architectural advantages as we develop the synchronization service.

IV. Dialogue Management

Dialog management concerns the control of the end user session interaction. Specifically, who has permission to speak and when, who can pass the control and how is that implemented. In this section we shall describe the environment for the dialogue management function and develop several possible options for implementing this function.

Dialog management requires that each of the virtual users have a token or have access to a baton in order to seize control of the session. In the course of a typical session, the two virtual users fist establish the initial sub session that becomes the first part of the session. The addition or binding of other virtual users through sub sessions to the session allows for the growth of the session. The baton or token may be a visible entity that is handed from one to the other or it may be hidden in the construct of the applications.

Consider the session level service called dialogue. The service can be implemented in four possible schemes. These schemes are:

(1) **Hierarchical:** In this scheme there is a single leader to the session and the leader starts as the creator of the session. The baton to control the

session can be passed upon request from one user to another, while full control remains with the session leader. The session leader may relinquish control to another user upon request and only after the leader decides to do so. The leader passes the baton from users to user based upon a first come first serve basis. It is assumed that each users may issue a request to receive the baton, and that any requests that clash in time are rejected and the user must retransmit. There transmit protocol uses a random delay to reduce the probability of repeated clashing. The leader always acknowledges the receipt of the request as well as a measure of the delay expected until the baton is passed.

(2) **Round Robin:** In this scheme, the baton is passed sequentially from one user to another. Each user may hold the baton for up to Tbat sec and then must pass the baton. When the baton is held, this user controls the dialogue in the session.

(3) **Priority:** In this case, all of the users have a priority level defined as $P_k(t)$, where k is the user number and t is the time. We let the priority be;

$$P_k(t) = R_k(t) + T_k(t) + D_k(t)$$

Here R is the rank of the k the user, T is the time since the last transmission and D is the data in the buffer. We assume that some appropriate normalization has occurred with this measure.

Every T_{check} seconds, each users, in sequence sends out a small pulse to all other users, on a broadcast basis, and tells them their current priority. Each user calculates the difference between theirs and all the others. User k calculates a threshold number, TR_k, which is;

$$TR_k = \max |P_k(T)-P_j(T)|$$

If $TR_k > 0$, then user k transmits its packets for T_{send} seconds.

(4) **Random Access**: Each user has a control buffer that indicates who has control of the session, namely who has the baton. The session is broken up into segment T_{sess} in length, with T_{req} seconds being relegated to a baton ownership selection period and T_{sess}-T_{req} being left for the session operation. During T_{req} , all of the users transmit a request packet that is captured by all of the other users buffers. T_{req} is broken into two parts, T_{send} and T_{eval}. These requests are broadcast in T_{send}.

Now after the sent messages are received, one of two things can happen, the message is received or it collides with another message and is garbled. If the message is garbled, the buffer is not loaded and is left empty. If it is filled, then each buffer during T_{eval} sequentially broadcasts its contents and all of the users listen to the broadcast and record the counts, N_k where N_k is the number of votes for user k in that call period.

The choice of baton control is then;

Choose user k if $N_k = \max_j | N_j |$

else restart T_{req} again.

For each of the protocols we describe the advantages and disadvantages of each in Table 4.3.

Table 4.3 Dialog Protocol Comparison

Protocol	Advantage	Disadvantage
Hierarchical	Single Point of Control of the Session.	Lacks capability to have open discussion.
Priority	Establishes who is in charge by allocation.	Requires a scheme to give priority that may be open to compromise.
Round Robin	Everyone gets to talk. Egalitarian approach.	May be excessively time consuming.
Random	Strongest player wins.	May not permit dissent.

V. Activity Management

Activity management looks at the session as an ongoing activity that users may come and go to. This services provides an ability to easily add, delete and terminate the entire session.

An activity in the terms of the session is a total bounded event that can be compartmentalized in such a way that other events may be locked in suspension until that event is complete. Activity management is in the session layer a function similar to transaction management in a transaction processing system. It allows for the definition of demarcation points that permit suspension of activities in other areas until the activity managed transaction is complete. Activity management can also be developed to manage a set of events that can be combined into a single compound event.

There are several characteristics that are part of activity management:

Activity Definition: This allows for the defining of an activity as composed of several dialogue. It allows for the defining of the activity as a key element of a single session or even to expand over several sessions.

Activity definition is the process of informing the session server of the beginning and end parts of an activity and in providing the session server with an identifiable name for the activity.

Activity Integrity Management: Activities are integral elements of action that cannot be segmented. The activity management system must ensure that once an activity is defined and initiated, hat no other activity that could interfere with the existing one is allowed to function.

Activity Isolation: The ability to provide integrity is one part of managing the activity. Another is the ability to isolate the activity from all others in the session. An activity must be uniquely separable from all other activities, and this separation in terms of all of its elements must be maintained throughout its process.

Activity Destruction: All activities must be destroyed at some point. This is a standard characterization.

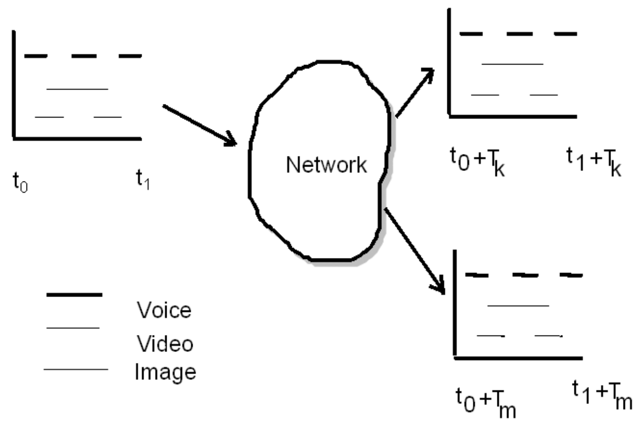
There are several sets of activities that are definable in a multimedia environment. These are as follows: Compound Multimedia Object Transfer, Sub-Session, Management, Dialog Control

The algorithms to perform the activity management functions are developable consistent with the OSI standards. There are no significant special development necessary.

VI. Synchronization Management

Synchronization is a session service that ensures that the overall temporal, spatial and logical structure of multimedia objects are retained. Consider the example shown in Figure 6.1. In this case we have a source generating a set of Voice (VO), video (VI), and Image (IM) data objects that are part of a session. These objects are simple objects that combined together form a compound multimedia object. The object is part of an overall application process that is communicating with other processes at other locations. These locations are now to receive this compound object as show with the internal timing retained intact and the absolute offset timing as shown for each of the other two users.

Figure 6.1 Synchronization



In this example, the synchronization function provided by the session server to the applications processes at the separate locations is to ensure both the relative and absolute timing of the objects. The location of the functionality can be centralized or distributed. Let us first see what the overall timing problem is. Consider a simple SMO synchronization problem. The network than transmits the packets and they arrive either in order or out of order at the second point. The session server must then ensure that there is a mechanism for the proper reordering of the packets at the receiving end of the transmission.

Let us consider what can happen in this simple example.

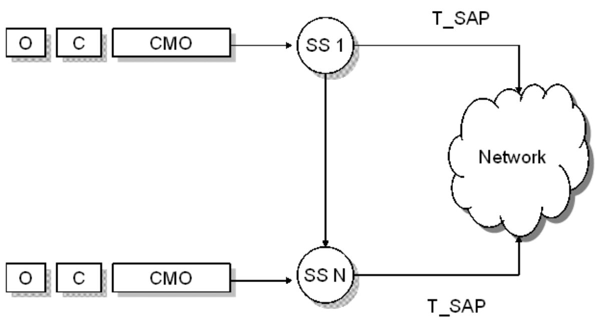
First, if the BMO of the SMO is very lengthy, then as we packetize the message, we must reassemble it in sequence for presentation. Let us assume that the BMO is an image of 100 Mbits. Then let us assume that the packet network has a packet delay that will be low if there is no traffic and grows as traffic increases. Now let us assume that the network provides 500 bit packets transmitting at 50 Mbps.

Second, let us note that there are 200,000 packets necessary to transmit the data. Each packet takes 10 microseconds to transmit. If we assume that there is a load delay of 5 microseconds per packet, then the total transmit time goes from 2 to 3 seconds.

We can also do the same with a compound object. In this case, we take the CMO and note that it is composed of SMOs. The SMOs must then be time interleaved over the transmission path to ensure their relative timing. It is the function of the session service to do this. The application merely passes the CMO and its header information as a request to the session server to ensure the relative timing is maintained.

The architecture for the session synchronization problem is shown if Figure 6.2. Here we have a CMO entering the network, knowing that the session server at Server 1 must not only do the appropriate interleaving but it must also communicate with the other servers (in this case K and N) to ensure that de-interleaving is accomplished. We show the session servers communicating with the network through the T_SAP and that in turn takes care of the packetizing. However, we also show that the session server, 1 and N, communicate in an out of band fashion, using some inter process communications (IPC) scheme, to ensure that the relative actions are all synchronized amongst each other.

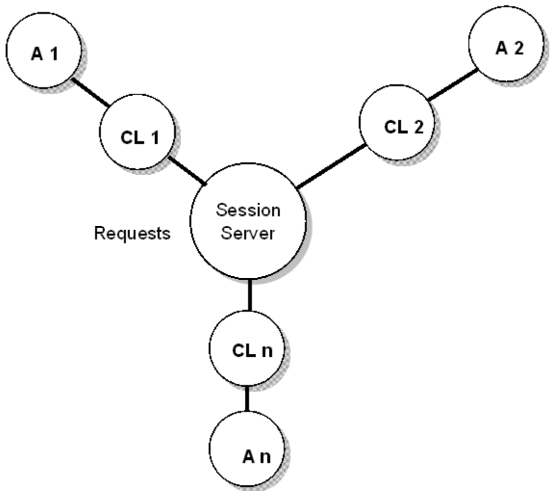
Figure 6.2 Synchronization Architecture



We can now envision how the architecture for this can be accomplished. There are two schemes:

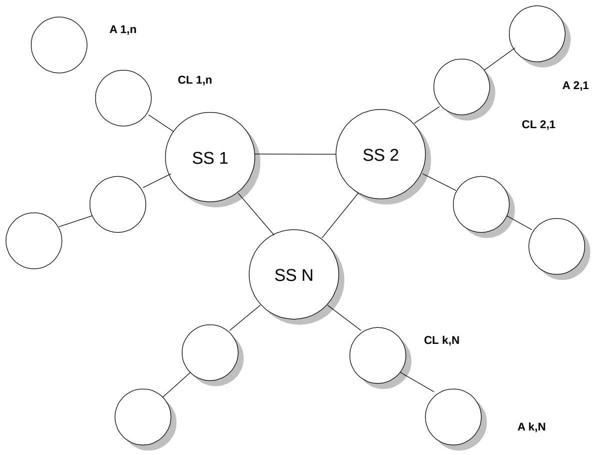
Centralized: Figure 6.3 depicts the centralized synch scheme for the session service. It assumes that each application (A) has a local client (CL). The application communicates with the local client (CL) to request the session service. The session server is centrally located and communicates with the application locally by means of a client at each location. This is a fully configured client server architecture and can employ many existing techniques for distributed processing (See Mullender or Couloris et al).

Figure 6.3 Centralized Architecture



Distributed: In contrast to the centralized scheme, we can envision a fully distributed session server architecture as shown in Figure 6.4. In this case we have a set of applications, and cluster several applications per session server. We again user local clients to communicate between the session server and the applications. The clients then provide local clusters of communications and the session servers allow for faster response and better cost efficiency. However, we have introduced a demand for a fully distributed environment for the session managers to work in a distributed operating system environment. As a further extreme, we could eliminate the clients altogether by attaching a session server per applications and allow for the distributed processing on a full scale.

Figure 6.4 Distributed Architecture

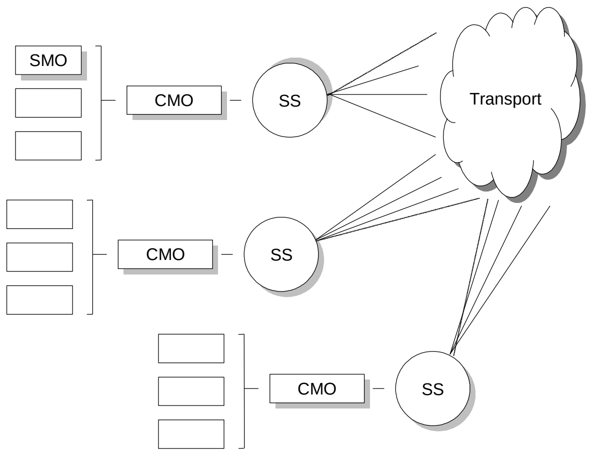


The major functions of the session server in its synch mode are:

1. To bind together simple objects into compound objects as requested by the application.
2. To provide intra object synchronization to ensure that all timing within each object is met.
3. To orchestrate amongst objects to provide inter object timing.
4. To minimize delay, slippage, between simple objects.
5. To minimize delay, latency, between different users.

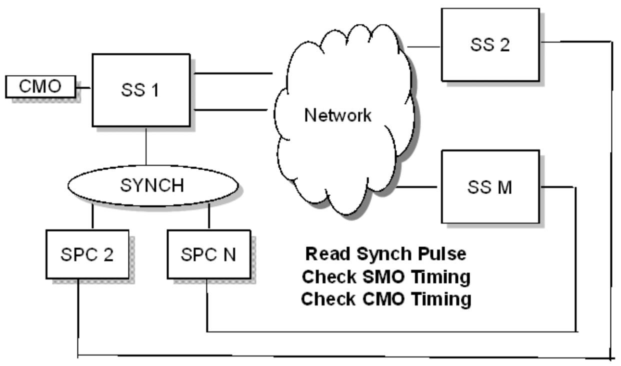
To effect these requirements, we have developed and implemented a scheme that is based on a paradigm of the phased locked loop found in communications (See McGarty and Treves, McGarty). We show this configuration in Figure 6.5. Here we have a distributed session server architecture receiving a CMO from an application. The session server passes the message over several paths to multiple users. On a reverse path, each server passes information on the relative and absolute timing of the CMO as it is received using the session services primitives found in the OSI model. Generally for segmented BMOs this is a simple problem but with streamed BMOs this becomes a real time synchronization problem.

Figure 6.5 Synchronization Architecture



The specific implementation is shown in Figure 6.6. Here we show M session servers and at the sending server we do the pacing of the packets to the T_SAP and allow for the interleaving of the SMOs. Based on the commands from the feedback system we provide delay adjustment, through caching and resetting priorities to the T_SAP for quality of service adjustments for the lower layer protocols.

Figure 6.6 Detailed Synchronization Implementation



At the receiving session servers, the synch pulses are read by the server, the SMO timing errors are read, knowing the synch header, and an error message is generated. We also do the same for the inter object CMO timing error.

The information is sent back in an out of band fashion to the source session server which in turn controls the synch control pulses for the source session server.

We can provide further detail on the synchronization scheme as follows:

A CMO is generated by the applications program. This may be a totally new CMO or a result of a new SMO addition or deletion.

The Source Session Server (SSS) transmits the header of the CMO to the Receiver Session Servers (RSSs). They then respond with an acknowledgment and in turn set up their internal timing and sequencing tables for local control. They also use the CMO header to adjust their local clock for network timing references.

The SSS commences to interleave, sequence and pace the SMOs of the CMO down to the T_SAP for transport across the network. At this point, the Transport protocol must have certain requirements of either increasing bandwidth (e.g. local data rate requests and also controlling sequence order. This interaction between the SSS and the T_SAP will define what additional capabilities we will need at the Session layer.

At indicated instances, the SSS inserts local synch pulses in the interleaved CMO. The synch pulses are to be used as local timing reference point for global coordination.

The RSSs read the local synch pulses and relates them to both the SMO and the CMO and obtain offsets from the global system clock that has been updated in the RSS. It then send back the offset of the synch pulses on a periodic basis. The offset is a vector that is given by:

$$E(k,j) = [e(k,j,1),.....,e(k,j,n),e(k,j,M)]$$

where E(k,j) is the offset vector of RSS j at time instant k. The internal values of the vector are the offsets of each of the SMO elements and the last entry is the offset of the CMO.
The SSS uses the set of E(k,j) for j=1,..,N RSSs to calculate an overall error signal to control the SSS. There are two major control features. If the average error is low then the SSS can reduce the insertion of synch pulses and the lower the processing load. If the errors are large, then more synch pulses are inserted to obtain finer loop control. The second element is control over the lower layers. We use the magnitude of the delay offsets to send messages to the T_SAP to change the quality of service parameters for the system.

We have developed several performance models for these protocols and the architecture that has been developed to implement them.

VII. EVENT MANAGEMENT

Event Management deals with the overall end to end management of the session. It is more typically viewed as a higher level network management tool for multimedia communications. In the current sate this service is merely a reporting mechanism. Although ISO has expanded the network management functionality of the seven layers, most of the functionality is still that of event reporting. In this section we discuss how that can be expended for the multimedia environment.

Event management at the session layer provides for the in band signaling of the performance of the various elements along the route in the session path as well as reporting on the status of the session server and the session clients. We note that each entity in the session path, which is limited to all involved clients and all involved servers provide in band information on the status of the session. In particular the in band elements report on the following:

Queue size at each client and server. The queue size can be determined on an element by element basis.

Element transit and waiting time. For each element involved in a session, the time it takes to transit the entire block as well as the time that the block has been in transit can be provided.

Session synchronization errors can be reported in this data slot. These errors can be compared to lower level errors and thus can be used as part of the overall network management schema.

The structure of the event management system has been effectively demonstrated. It is represented as a header imbedded in the transit of every data block. We can generate specific event management blocks that are also event driven and not data transit driven. These are generated by direct transmission of such blocks as overhead devoid of data content.

VIII. Conclusions

What we have shown in this paper is that the session layer functions are key to supporting the overall needs of a multimedia communications environment. We have also developed algorithmic approaches for dialog and synchronization services and have shown that these services depend upon the lower layers for support. Specifically, we have shown that if the underlying communications network is jittery in the packet transport provided, the resulting delays associated with the synchronization process can be significant.

Architecturally, we have raised several issues as to how best to provide the session service, specifically where to place and how to communicate with a session server. The session services require considerable entity to entity communications and this may require a distributed environment of session servers all functioning in a fully distributed mode. In the network applications developed to date (See McGarty and Treves), the session server has been centralized and has allowed for communications in a distributed fashion on a UNIX environment using sockets (Berkeley 4.3). However, in future implementations, the session server will be architects in a more distributed fashion.

IX. Acknowledgment

The author would like to thank Muriel Medard who assisted him in the preparation of the material herein and also assisted him in the teaching of Multimedia Communications at MIT in the Fall of 1989 during which this works was accomplished.

X. References

[1] Adiha, M., N.B. Quang, Historical Multimedia Databases, Conf on VLDB, Kyoto, 1986.

[2] Bradley, Alan, Optical Storage for Computers, Wiley (New York) 1989.

[3] Burns, Alan, Andy, Wellings, Real Time Systems and Their Programming languages, Addison Wesley (Reading, MA), 1989.

[4] Chang, N.S., K.S. Fu, Picture Query Languages for Pictorial Data Base Systems, IEEE Computer, Nov 1981.

[5] Chang, S., T. Kunii, Pictorial Data Base Systems, IEEE Computer, Nov 1981, pp 13-19.

[6] Christodoulakis, et al, The Multimedia Object Presentation Manager of MINOS, ACM, 1986, pp 295-310.

[7] Christodoulakis, S., et al, Design and Performance Considerations for an Optical Disk Based Multimedia Object Server, IEEE Computer, Dec 1989, pp 45-56.

[8] Elmarsi, Ramez, Shamkant Navathe, Fundamentals of Database Systems, Benjamam (Redwood City, CA) 1989.

[9] Hanson, Owen, Design of Computer Data Files, Computer Science Press (Rockville, MD) 1988.

[10] Kim, W., H. Chou, Versions of Schema for Object Oriented Databases, Conf VLDB, 1988.

[11] Krishnamurthy, E.V., Parallel Processing, Addison Wesley (Reading, MA), 1989.

[12] Kunii, T. L., Visual Database Systems, North Holland (Amsterdam), 1989.

[13] Little, T.D.C., A. Ghafoor, Synchronization and Storage Models for Multimedia Objects, IEEE Journal on Sel Areas in Comm, April, 1990, pp. 413-427.

[14] Loomis, Mary, Data Management and File Structures, Prentice Hall (Englewood Cliffs) 1898.

[15] Maier, David, The Theory of Relational Databases, Computer Science Press (Rockville, MD) 1983.

[16] McGarty, T.P., L.B. Ball, Integrated Network Management Systems, IEEE NOMS Conf, New Orleans, Nov. 1987.

[17] McGarty, T.P., Multimedia Communications, Wiley, to be published.

[18] McGarty, T.P., Multimedia Data Base Systems, Presented at Syracuse University, April 30, 1990.

[19] McGarty, T.P., S.T. Treves, Multimedia Communications Applications in Health Care Services, SCAMC Conf, Washington, DC, Nov, 1990.

[20] McGarty, T.P., Session Management in Multimedia Communications, Presented at MIT, May 2, 1990.

[21] McGarty, T.P., Stochastic Systems and State Estimation, Wiley (New York), 1974.

[22] McGarty, T.P., Understanding Multimedia Communications, Presented at MIT, February, 1990.

[23] Mee, C. Dennis, Eric D. Daniel, Magnetic Recording, McGraw Hill (New York) 1988.

[24] Nicolau, Cosmos, An Architecture for Real Time Multimedia Communications Systems, IEEE Journal on Sel Areas in Comm, April, 1990, pp. 391-400.
[25] Parsaye, Kamran, et al, Intelligent Databases, Wiley (New York) 1989.
[26] Pizano, A. et al, Specification of Spatial Integrity Constraints in Pictorial Databases, IEEE Computer, Dec 1989, pp 59-71.
[27] Steinmetz, Ralf, Synchronization Properties in Multimedia Systems, IEEE Journal on Sel Areas in Comm, April, 1990, pp 401-412.
[28] Teorey, Toby, James Fry, Design of Database Structures, Prentice Hall (Englewood Cliffs, NJ) 1982.
[29] Terry, D., D. Swinehart, Managing Stored Voice in the Etherphone System, ACM Trans Cptr Sys, Vol 6, No 1, Feb 1988, pp3-27.
[30] Tsichritzis, D. et al, A Multimedia Office Filing System, Proc VLDB 1983.
[31] Ullman, Jeffrey, Database and Knowledge Base Systems, Computer Science Press (Rockville, MD) 1988.
[32] Woelk, D, W. Kim, Multimedia Information Management, VLDB Conf 1987.
[33] Woelk, D. et al, An Object Oriented Approach to Multimedia Databases, ACM, 1986, pp 311-325.

Terrence P. McGarty (M’63) is Managing Partner of The Telmarc Group and is also Lecturer at MIT, Cambridge, MA. McGarty holds a PhD from MIT in Electrical Engineering and Computer Science. He has been on the faculties of MIT, Columbia University, George Washington University, and Polytechnic Institute. He also has extensive business experience in telecommunications*.