Management and Delivery of Multimedia Traffic*

Panos K. Chrysanthis and Daniel Mossé
Department of Computer Science
University of Pittsburgh
Pittsburgh, PA 15260
{panos,mosse}@cs.pitt.edu

Abstract – We propose a scheme for the establishment of multimedia delivery channels in WAN environments that support the timing and reliability requirements of multimedia applications while allowing for better utilization of the network bandwidth. These virtual, end-to-end channels from source to destination consist of *multiple segments* that (a) have a resource reservation scheme that is able to issue guarantees of timeliness and reliability; (b) consider data processing and transformation requirements at specific, intermediate sites; (c) support *traffic splitting* to cope with limited bandwidth and reduce contention; and (d) offer alternative paths for fault tolerance and reliability.

I. Introduction

"It's 6 o'clock. Martha comes home tired from a heavy day at work. She is ready for watching the custom news on her Viewing Unit before watching a movie at 8pm. Similarly, Dave has finished all his work- and non-work-related activities for the day. He is ready to dive into Cyberspace and let the information highway lead the way to a more comprehensive news in certain subjects he is curious about. By 8pm, Dave is ready to place a request to watch a movie at 8:25."

A typical user of tomorrow's 6 o'clock news services will specify queries on particular subjects and headings, in his or her Viewing Unit. A typical request will attach to each subject a depth-level in terms of showing time (in parenthesis) and information (in brackets):

- Stories on criminality and its effects on American people (15) [expert reports]
- Specific stock (6) [quotations, commentaries]
- The weather forecast (2) [local, extended]
- Sports (10) [headlines]

The realization of such novel services and similar home applications such as netshopping and netmarketting will require multimedia system support to retrieve and present information in a manner that is both *timely* and *efficient*, as well as to handle the trade-offs between these often competing goals. For example, in the stock option query, the requested information must be *both* retrieved and presented within a tight time bound due to the temporally sensitive nature of the data items. In other cases, the time of retrieval may be separated from the time of presentation; retrieving information ahead of time can be an effective way of shifting system load from peak to off-peak periods.

With respect to efficiency, in massive information networks, there are often alternative ways to retrieve a piece of information. For example, the local weather forecast is available on various sites, in different formats, containing different amounts of detail and varying degree of consistency and accuracy. Moreover, there are typically multiple access paths to each site, and they may vary in reliability, cost, and speed.

The above example illustrates the wide range of timeliness and efficiency requirements that constitute the user-specified *Quality of Service* (QoS) that a system should be able to deal with. A user negotiates with the system the QoS parameters in terms of acceptable delays, minimum reliability, and maximum monetary cost.

For some applications, such as synchronized streams of audio and video channels, the timing constraints can be very stringent. Since this real-time requirement must warrant guaranteed and predictable behavior from the various components of the system, it is imperative that some form of resource allocation take place prior to the presentation of the multimedia applications [10, 22].

The resource allocation problem in distributed systems is computationally challenging, if the resources available are to be efficiently utilized. In a wide area network (WAN), resource allocation is even more difficult due to heterogeneity of sites and networks. The heterogeneity between sites executing a multimedia application in con-

*Supported in part by NSF RIA grants CCR-9308886 and IRI-9210588

0-7803-27560X/95/$4.00 © IEEE

junction with the diversity of data representations, further complicate the reliable delivery of multimedia data. It is often imperative for data filtering and transformation to be performed on the data at specific intermediate sites appropriate for such processing (e.g., some applications may need a proprietary decompression algorithm, or the processing power of a supercomputer). A resource reservation scheme that supports reliable delivery of multimedia data needs to take into consideration these disparities between sites and networks.

In this paper, we propose a scheme for the establishment of multimedia delivery channels in WAN environments that support the timing and reliability requirements of multimedia applications while allowing for better utilization of the network bandwidth. These virtual, end-to-end channels from source to destination consist of *multiple segments* that (a) have a resource reservation scheme that is able to issue guarantees of timeliness and reliability; (b) consider data processing and transformation requirements at specific, intermediate sites; (c) support *traffic splitting* to cope with limited bandwidth and reduce contention; and (d) offer alternative paths for fault tolerance and reliability. Another salient characteristic of the multi-segmented channel establishment is that it can be *tariff-based*. In other words, in addition to the timing, reliability and intermediate processing requirements, our scheme takes into account the monetary cost for the reliable delivery of multimedia traffic.

The rest of this paper is organized as follows: in the next section we discuss related work. In Section III, we describe the NETWORLD, that is, our system architecture. In Section IV, we describe the multi-segmented virtual channel scheme along with discussions on resource allocation, traffic splitting, and fault tolerance. We close the paper with conclusions and future work.

II. RELATED WORK

A. Network Support

Schemes based on *time division multiplexing* underutilize the network when the channel is idle [23]. In [6], the resources are granted on basis of fairness, according to the channels' requirements. Channels that use less than their fair share are able to get faster service. In order to support performance guarantees, in [5] an admission algorithm was added. The problem with these mechanisms is that they have high delays under higher loads and cannot control the jitter of packets, an important aspect of multimedia applications.

One of the first resource reservation schemes schemes was presented in [8]. The delay-jitter problem was further addressed in [22], and parameters were added to the QoS specification that deal with a channel's rate, burst, maximum end-to-end delay, jitter, and on-time reliability. This scheme is much too complex to be practical, and thus approximations have been developed that trade accuracy and guarantees for good resource utilization and efficiency.

In [9, 10] a versatile architecture, called $\mathcal{V}$–NET, was presented to address these concerns, while being flexible enough to accept different scheduling disciplines in different sites, different admission control policies, and different types of traffic sources. The architecture is based on an end-to-end communication channel ($\mathcal{V}$-channel) which represents an association between the application's QoS specifications and the network resources. The $\mathcal{V}$–NET provides a framework for flexible support of both real-time and non-real-time communication requirements.

B. Real-Time Scheduling

Many interesting results have been developed for *timeline-based* schedulers [4, 19, 18] and *rate-based* schedulers. The latter technique has been extensively studied for the case of periodic applications, including both the general case [14] and with more specific cases (priority inversion and other priority-based protocols [3, 17, 21]).

The algorithms that have considered the distributed scheduling of resources have often made limiting assumptions on the nature of tasks, or on the nature of the systems. Exceptions include distributed versions of the timeline-based algorithms [13], the generalized rate-based versions [20], and the *end-to-end* paradigm [7, 13, 12], where the deadline of the entire application is taken into account when scheduling, instead of considering only the deadlines of independent tasks. This turns out to be a complex problem which requires a comprehensive solution.

Most of the end-to-end scheduling algorithms schedule the entire path of computations and, upon accepting the channel/computation, issue a guarantee that the path is feasible from source to sink of the application. Similarly, our scheme establishes predictable multi-segmented channels by considering the entire application's QoS requirements as well as the individual task's timing and resource requirements.

III. SYSTEM ARCHITECTURE

In this section, we describe the NETWORLD, a system architecture we have developed for distributed multimedia systems. We assume that a set of processors and links in the WAN can fail at any instant of time and that another

set cannot fail before the system recovers from the first failure (reconfiguration latency). We also assume that there exists some fault detection mechanism that detects a site crash (e.g., fail-signal processors [16]) or errors of short duration. Both permanent and transient faults can be handled by our approach.

A. Basic Components

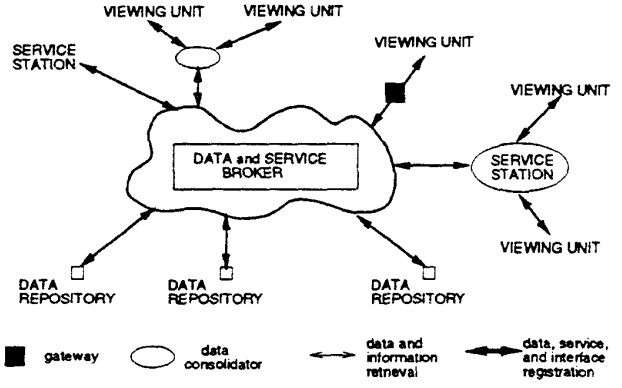

Figure 1: NETWORLD Architecture

The NETWORLD architecture (see Figure 1 and Figure 4) for the information access and retrieval in physically and logically distributed networks comprises:

Viewing Units (VUs), which are multimedia capable client workstations. These workstations should allow for different types of multimedia data to be displayed and perused, and should contain the appropriate software to translate the user request into requests for data with timing constraints. Furthermore, each VU also contains a *Remote Access Manager* that initiates the connection with the remote sites from which it will receive the requested data; it is also part of the allocation scheme, negotiating the QoS parameters and resource usage with the remote components of the NETWORLD.

Data and Service Broker (DSB) that serves both as global resource manager and as an information directory (i.e., performing meta-scheduling and metadata management). It provides a common interface between the various components of the system and exports services needed by a VU but not supported by the underlying system. In that respect, the functionality of the DSB resembles that of CORBA [11], with the difference that the DSB is customized for multimedia applications, with timing constraints and special multimedia data services.

Data Repositories (DRs) that incorporate real-time scheduling and data management functions to be able to satisfy requests in a timely fashion. Due to this characteristic, the DRs can be seen as managers of temporal data with timing constraints. That is, real-time tasks execute when a request for multimedia data arrives at the DR. Requests can be known in advance or not, as in pre-planned teleconferences or in dynamic request for current stock prices.

Service Stations (SSTs) that provide specialized data transformations (e.g., data compression and decompression). The resources needed to accomplish the data transformations must be reserved prior to accepting the task of performing the transformations (to provide guaranteed behavior). Furthermore, this must done without violating the timing constraints of other computations, and is achieved by using a real-time scheduling scheme.

Data Consolidators or *Distribution Centers* (DCs) that temporarily accumulate data to be transmitted to the VUs. These DCs function as front ends to the NETWORLD and at the same time as extended storage for the VUs. The DCs may consolidate multiple requests for the same multimedia data from different VUs into a single request. This simplifies the remote access to achieve better resource utilization, consequently minimizing the cost to the users. The requests from the VUs do not need to be satisfied simultaneously, but data can be sent to the VUs at different times and in accordance to the specified QoS. In that sense, the DCs function as "short-term" DRs.

IV. MULTI-SEGMENTED VIRTUAL CHANNELS

Our goal is to support reliable delivery of digital and multimedia data from several sources in a packet-switched network such as WANs. This can be achieved by establishing appropriate connections, i.e, *virtual channels*, between DRs and the VU executing the multimedia application, if there are sufficient resources to satisfy the user's QoS requirements. Our solution is comprehensive in that it integrates reliability aspects with reservation of processor, buffer, and network resources for distributed multimedia systems. It uses the *V*-channel [10] as the basic underlying abstraction for establishing virtual channels, combined with a more generalized approach for fault-tolerant resource allocation [13, 15] to establish reliable multimedia channels.

Recognizing that different resource allocation and scheduling schemes are employed at the network and op-

erating system levels, we approach the problem of predictable delivery of multimedia traffic in a divide-and-conquer manner. We divide the end-to-end virtual channel from a source to a destination into several *segments*, each consisting of communication links and store-and-forward switches (sites) with common characteristics. For example, a channel crossing multiple types of networks is divided into segments at the gateways between the networks so that each segment resides entirely within a network type. The establishment of the channel, thus, is subdivided into several segments and these segments are the $\mathcal{V}$-channels to be established *independently* from the system point of view[1]. Further, if transformation is needed on the data being transmitted, extra resources must be reserved at the service station (SST) where the transformation will be applied, say site i. Since the $\mathcal{V}-$NET only deals with resource reservations at the network level to yield guaranteed delays for data transfer, we add another layer of scheduling to reserve resources at the endpoints of each segment, in order to timely perform the needed data transformations. In the example, the channel is broken down into a segment *before* and a segment *after* site i, and timing and resource requirements of the two segments are determined at the network level independently from the resource requirements at the SSTs. Note that a channel is only established after it is determined that there exists a site that is able to perform the necessary data transformation (i.e., the software and hardware needed for the transformation is present in that site). The resource allocation is detailed in Section *IV.A*.

Multiple segments can be established between two sites along a channel, and these can serve multiple purposes:

- Better utilization of the limited bandwidth of the network. This can be done by *traffic splitting*: create several different outgoing divide the traffic among them. In other words, we name the multiple segments with the same channel identifier and split the traffic among the segments according to some heuristics (see Section IV.B). In this case, the $\mathcal{V}$-channel establishment procedure is used several times (one for each segment), depending on the traffic required and capacity of the network.
- Due to the traffic splitting scheme, the network contention can be decreased, consequently decreasing the number of lost packets and therefore increasing the reliability of the channel.
- Fault tolerance can be provided by establishing several alternative paths for each segment (see Section *IV.C*), in the same fashion of TMR or concurrent recovery blocks [2]. This will yield hard guarantees, including the case of the data corruption (e.g., transient failures in the network or store-and-forward sites) and data loss (e.g., permanent link or site failures, lack of buffers, etc).

  In addition, by using specific encoding algorithms that increase the reliability of the application [1], the multi-segment approach presents a solution for the partial message loss (some of the packets of a message may be lost, for example, due to network congestion during transient overloads). Such encodings introduce extra traffic on a channel due to the encoding scheme, but that traffic can be absorbed by the traffic splitting strategy.

The idea of multiple segments also allows multiple configurations, which in turn allow the system to apply dynamic algorithms to solve unpredictable situations. This is important when we consider the idea of alternative scenarios, or changes in the operational mode of the system. For example, if there is a single $\mathcal{V}$-channel segment established between two points in the network for normal operation, the system can establish a contingency segment that will be used only in case of emergency.

Also, if there are several segments, the guarantees can be given in a tariff-based, dynamic fashion for each segment, allowing for unusually variable or bursty traffic to be accommodated. For example, if the system receives a new user request, with a cost threshold[2] of C, the system may choose to cancel the guarantees of previously accepted jobs whose cost thresholds were $c \ll C$. The ratio c/C will depend on the characteristics of the system or applications, and the penalty assessed for breaking the contracts with the c-threshold user.

Finally, the resource allocation assumes a negotiation phase between an application and NETWORLD to establish the type of traffic, the encoding techniques, the redundant segments, among other factors. We describe the resource allocation next.

A. *Resource Allocation*

Resource allocation comprises more than simply allocating and scheduling resources at a specific site. Our ultimate goal is to guarantee that several streams of data will be able to be displayed in a reliable, timely, and synchronized fashion at a destination. In this context, we describe the resource allocation algorithm and the framework for the establishment of an entire channel.

[1] Note that from the application perspective, the channel being established is not segmented.

[2] The cost is expressed in terms of an abstract metric, which is a function of actual resources being used, such as CPU cycles and Kbytes of RAM as well as timing delays.

The multimedia application translates the user-specified QoS parameters into system-level QoS parameters. In addition, the system translates the user request into a *task precedence graph* (TG) [15], where the *nodes* represent tasks needed for data transformations and the *edges* depict communication between tasks. Assuming, for example, that the user requests two datatypes (a map and a video clip), Figure 2a shows how the system breaks the datatypes down into three data streams to be delivered and Figure 2b shows the TG depicting the system view of the tasks for retrieval and delivery of these three data streams. Note that the processing of the video stream and map have two alternative paths. That is, the TG is composed of both AND-paths and OR-paths (the dark arrows depict the AND-paths).

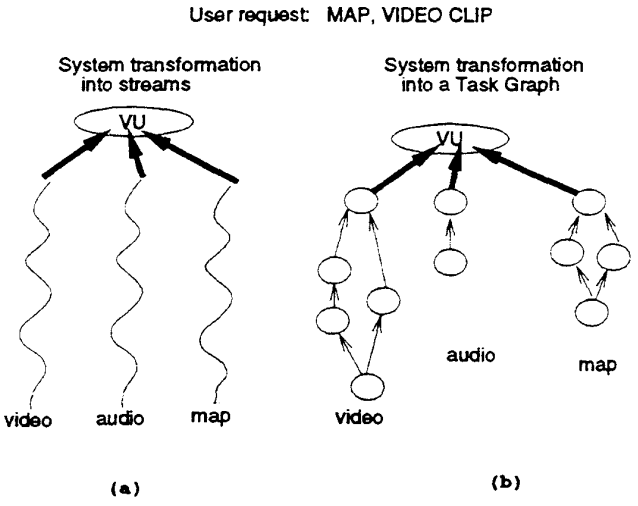

Figure 2: (a) system streams, (b) task graph

After the TG has been constructed, the *mapping* of the TG onto the NETWORLD is carried out, which constitutes the resource allocation phase itself. Specifically, *paths* in the TG are mapped to potential *routes* in the NETWORLD. A route is a set of communication links and intermediate sites between a source and a destination in the NETWORLD. These routes are subsequently segmented and, for each segment, a $\mathcal{V}$-channel is established. The $\mathcal{V}$-channel allocation algorithm takes as input the data requirements of the application (e.g., a video application may need to transfer 20Mbit/sec) and returns a delay and a cost value for the segment. If the delay/cost combination of the route exceeds the maximum values requested by the user, there are two possibilities: (a) discard that particular path of the TG and explore an alternative OR-path or an alternative route; (b) use traffic splitting, as described in the next section. In this section we elaborate further on the first possibility.

Resource Allocation for a TG::

1. Identify the sites in the system with capability for the transformations tasks in the TG.
2. Identify the possible routes from a source to a destination in the system, with the required capabilities for data transformations create an ordered list of candidate routes, *CandRoutes*.
3. Select the first route R from the list *CandRoutes* ($R \leftarrow$ *CandRoutes[1]*).
4. Segment the route R into a set of segments ($segs = \{\sigma_1, \ldots, \sigma_m\}$), according to the data transformations, the availability of resources in the sites of the system, and the user-specified reliability/fault tolerance parameters (see details below). Let $Xsites = \{S_1, \ldots, S_n\}$, $n = m + 1$, be the set of intermediate sites that will carry out data transformations.
5. For each segment σ_i in $segs$ establish a $\mathcal{V}$-channel v between the two sites (S_i and S_{i+1}) identified as endpoints of that segment. The underlying abstraction, the $\mathcal{V}$–NET, yields a guaranteed delay for channel v on segment σ_i ($\delta_{i,v}$) and a cost in terms of resources ($\kappa_{i,v}$), based on the sites' availability of CPU and buffers.
6. For each site $S_j \in Xsites$, determine the processing delays d_j and the costs μ_j for data transformation.
7. If the route R satisfies the QoS parameters:
   (i) the maximum user-specified delay, Δ_{max}

$$\Delta_{max} \geq \sum_{i=0}^{m} \delta_{i,v} + \sum_{j=0}^{n} d_j$$

   (ii) the user-specified cost threshold, C_{max}

$$C_{max} \geq \sum_{i=0}^{m} \kappa_{i,v} + \sum_{j=0}^{n} \mu_j$$

   then ACCEPT R, else select a new route ($R \leftarrow$ *CandRoutes[nextroute++]*) and go to Step 4.

The crucial part in the algorithm above is Step 4, which defines the segments to be constructed. Figure 3 shows an example of the mapping of the video clip from Figure 2b onto the systems resources. The store-and-forward sites are depicted by black circles and the sites $S_i \in Xsites$ are depicted by squares. The tasks of the TG are mapped *only* onto the squares, and the communication tasks are mapped onto the store-and-forward sites C. Note that the alternative paths for the video stream are mapped to two different routes on the NETWORLD, namely $S_4 \rightarrow S_5 \rightarrow S_3 \rightarrow S_6$ and $S_4 \rightarrow S_2 \rightarrow S_3 \rightarrow S_6$.

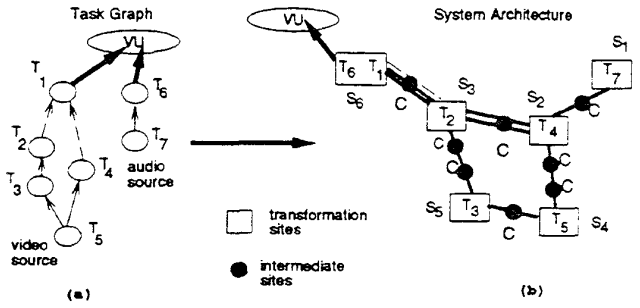

Figure 3: (a) video clip task graph, (b) TG mapped on system resources

Also, there are multiple segments between sites S_2 and S_3 (two independent segments) and sites S_3 and S_6 (two contingency segments depicted by a solid and a dashed line for the video and an independent segment for the audio).

B. *Traffic Splitting*

The traffic splitting scheme can be triggered at the start of a segment establishment (Step 5 of the resource allocation algorithm) or to remedy the situation of a rejected segment (Step 7 of the resource allocation algorithm). In the latter case, when an attempt by the allocation algorithm to establish a channel fails in a segment, the traffic splitting component of the allocation algorithm is invoked, with more concrete knowledge about the status of the network (or at least about the particular links that rejected the channel). On the other hand, if it is known *a priori* that the links do not have enough bandwidth or buffers to accept the volume of that channel, the traffic splitting component of the allocation algorithm is invoked at the start of the segment establishment.

Several options for splitting the traffic are outlined below. The "capacity" mentioned for each communication link includes the network capacity, which deals not only with the CPU cycles, but also with available buffers.

1. ***Even distribution***: split the data evenly on k outgoing links of a site (if total data is TD, data per link is TD/k). This can be easily achieved by sending packet i on outgoing link $i \bmod k$. The overhead to forward each packet to the correct link is no larger than forwarding each packet to a single link. Similar reasoning holds for the receiving end.
2. ***Proportional distribution***: send $TD \cdot P_l$ on each link l, where P_l the ratio of unused capacity on that link and the total unused capacity on all links. This exploits the unused capacity of each link "equally" and allows each link to have a certain left-over capacity for future requests.
3. *Maximum fit*: send as much data on each link as possible, filling each link in turn to capacity. This policy is similar to a first-fit algorithm for memory allocation, when several partitions are needed. Note that maximum fit will only work if there is enough knowledge about the network state to warrant such distribution.

It should be pointed out that although virtual channels may be split over segments (i.e., physical channels), the QoS parameters defined by the user still hold, and need to be obeyed. Therefore, it may be necessary to tune the traffic going through any of the segments created to fit the user-defined delay, cost, or reliability parameters. We intend to experiment with these metrics to show which of these have the best expected behavior with respect to channel admission when using traffic splitting.

C. *Fault Tolerance*

Our current scheme maps a permanent fault to a completely redundant path, and takes transient faults to a traffic splitting scheme. The allocation algorithm also updates the cost of the channel based on the resources consumed for the reliable channel (above the cost of the basic channel). If the network is homogeneous, the cost simply increases linearly with the number of extra links in the extra paths established for permanent faults (i.e., $cost' = cost \cdot redundancy\_factor$). If faults are only transient, and the traffic splitting is used, the increase is $cost' = cost \cdot split\_factor$, where $split\_factor$ is the increase in traffic caused by the encoding algorithm[3].

There can also be a combination of the two cases above, in which each segment uses data encoding and/or segment (TMR-type) redundancy. Such flexibility can yield large gains in reliability and timeliness of data delivery in cases of transient overloads. On the other hand, it is not clear whether this flexibility will be worthwhile in terms of efficiency of resource utilization, but it is a simple and clean algorithmic advantage in our approach.

With respect to specifying fault tolerance, there are two basic ways: through *implicit* and *explicit* specifications from the application. For implicit specification, the application specifies the cost to be added to the non-redundant channel, and the system decides where to establish additional segments used for fault tolerance between two sites. In the explicit way, the application determines the number of replicas and/or the number of segments to be established or the particular locations for the redundant components. The application may specify the number of

[3] Clearly, each encoding introduces a specific amount of redundancy. In [1] the extra data was shown to be in the order of 20% for MPEG traffic in the Mbone.

permanent or transient faults to be tolerated and the system maps these parameters to the actual data flow in each segment.

A combination of these can be implemented as a *hybrid* way to specify fault tolerance: the application specifies the level of reliability required and the maximum cost threshold; the system, in turn, applies the redundancy at any weak links or sites that it deems necessary. For example, consider a application that requests tolerance to one permanent fault and the probability of packet loss in the network is 15% within the transmission interval. In this case, the redundancy created by tolerating the permanent fault could be sufficient, if we could guarantee that the transient faults would only occur in one of the segments created. However, faults can occur randomly, and we need to include an encoding scheme that allows the loss of packets in either segment, thus introducing extra traffic in the network. If the system can accommodate a path in a segment, but cannot accommodate the second (redundant path) in another segment, the second path must be split between two segments, and the system creates a path straddling those two segments.

In Figure 3, the several segments representing the video stream ($\langle T_1, T_4, T_5 \rangle$ and $\langle T_1, T_2, T_3, T_5 \rangle$ mapped to $S_4 \rightarrow S_5 \rightarrow S_3 \rightarrow S_6$ and $S_4 \rightarrow S_2 \rightarrow S_3 \rightarrow S_6$, respectively) are examples of fault tolerance.

D. Example

To recap and illustrate the concepts discussed above, in this section we revisit the motivating application in the introduction, focusing on the *video-on-demand* part. This example also illustrates the aspect of negotiation of the requested QoS parameter between the VUs and the NETWORLD.

Recall that Martha (on VU_1) requested a movie for an 8pm viewing, and Dave requested (on VU_2) a movie for a 8:25pm viewing. Since the movie will be cached for the 8pm viewing and can be held until 8:10pm, the DC (Data Consolidator) may suggest to VU_2 to view the movie no later than 8:10 at a significant reduction in cost, because it takes advantage of the locally cached data of the 8pm request.

In addition to the stream/segment perspective, we show in Figure 4 how a request is processed and how the different components of the NETWORLD process the multimedia data. The retrieval, transformation, and delivery transactions are also shown.

In Figure 4, we show an execution path followed when multimedia data is requested by the two VUs (*request phase*): (0, not shown) Each VU receives a request from a user and transforms this request into QoS parameters;

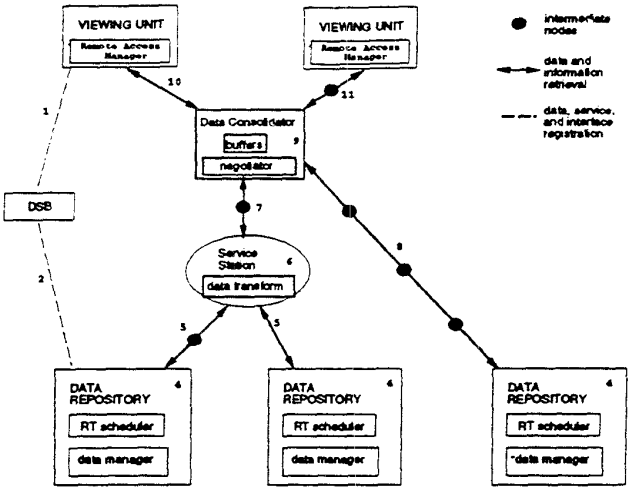

Figure 4: Execution of request phase and operational phase of multimedia traffic

(1) the VU passes the request to the DSB, either directly or indirectly via the DC; (2) the DSB contacts the DRs, with the appropriate route from the DRs to the VUs requesting the data; (3, not shown) the DSB contacts the SSTs and the DC, informing them that data transformation (e.g., decompression) and timing modifications are being applied (e.g., send the movie to VU_2 at 8:10pm).

Resource allocation phase: When the DSB gets a positive response from the SSTs and DC, the resources are allocated and the channels setup by the multi-segmented virtual channels. Note that the DC's negotiator is involved in the decision of advancing VU_2's viewing from 8:25pm to 8:10pm.

During the *operational phase*: (4) the DRs retrieve the necessary data in a timely (coordinated by the real-time scheduler) and consistent (coordinated by the local data manager) fashion; (5) data is sent to the SST, possibly through intermediate sites; (6) the SST carries out the data transformation; (7) the uncompressed data is forwarded from the SST to the DC; (8) alternatively, data is sent directly from the DR to the DC, if no intermediate transformation is necessary; (9) the data is held in the DC's buffers; (10) data is passed on to the VU_1, according to the QoS parameters of that request; (11) data is passed on to the second VU_2, after the timing modifications.

V. CONCLUSION AND FUTURE WORK

We have presented an architecture for distributed multimedia systems, and a scheduling/allocation scheme that achieves synchronization and timely delivery of these different streams in a guaranteed fashion. This scheme is

based on *multi-segmented virtual channels* and provides the flexibility to accept channels that cannot be accepted in single paths. The scheme also provides fault tolerance, allowing for users to specify the reliability of the channel both in terms of absolute number of failures and in percentage of packet losses.

Since our system allows different media streams from different sources, it is not unreasonable to think about the multimedia presentation being done at multiple recipient workstations (e.g., teleconferences). We can use the multi-segmented approach to extend the multicast paradigm, since we can depict sections of the tree by segments. That is, we will partition the multicast tree into branches, and these branches will be assigned segment numbers. This is one of the main areas of future work in our approach.

REFERENCES

[1] A. Albanese, J. Blomer, J. Edmonds, M. Luby, and M. Sudan. Prioritized Encoding Transmission. In 35^{th} *Symp. on Foundations of Comp. Sci.*, pages 604–612, 1994.

[2] F Belli and P Jedrzejowicz. An Approach to the Reliability Optimization of Software with Redundancy. *Transactions on Software Engineering*, 17(3):310–312, 1991.

[3] M. Chen and K. Lin. Dynamic Priority Ceilings: A Concurrency Control Protocol for Real-Time Systems. *Journal of Real-Time Systems*, 2(4):325–346, 1990.

[4] S. Cheng, J. A. Stankovic, and K. Ramamritham. Dynamic Scheduling of Groups of Tasks with Precedence Constraints in Distributed Hard Real-Time Systems. In *Proc. IEEE Real-Time System Symposium*, pages 175–180, 1986.

[5] D. C. Clark, S. Shenker, and L. Zhang. Supporting Real-Time Applications in an Integrated Services Packet Network: Architecture and Mechanism. In *SIGCOMM '92*, pages 14–26, 1992.

[6] A. Demers, S. Keshav, and S. Shenker. Analysis and Simulation of a Fair Queueing Algorithm. In *ACM SIGCOMM '89*, pages 3–12, 1989.

[7] M. DiNatale and J. A. Stankovic. Dynamic End-to-End Guarantees in Distributed Real-Time Systems. In *Real-Time systems Symposium*, pages 216–227, 1994.

[8] D. Ferrari and D. Verma. A Scheme for Real-Time Channel Establishment in Wide-Area Networks. *IEEE Journal on Selected Areas in Communications*, 8(3):368–379, 1990.

[9] B. Field and T. Znati. α-channel: A Network Framework to Support Appliction Real-Time Performamce Guarantees. *IEEE Journal on Selected Areas in Communications*, 11(3):1317–1329, October 1993.

[10] B. Field, T. Znati, and D. Mossé. $\mathcal{V}$–NET: A Framework for a Versatile Network Architecture to Support Real-Time Communication Performance Guarantees. In *InfoComm*, Apr 1995.

[11] OMG ORBTF (Object Request Broker Task Force). Common Object Request Broker Architecture. Technical report, Object Management Group, 1992.

[12] R. Gerber, S. Hong, and M. Saksena. Guaranteeing End-to-End Timing Constraints by Calibrating Intermediate Processes. In *Real-Time systems Symposium*, pages 192–203, 1994.

[13] S.-T. Levi, D. Mossé, and A. K. Agrawala. Resource Allocation under Fault-tolerance Constraints. In *IEEE Real-Time Systems Symposium*, Huntsville, AL, 1988.

[14] C. Liu and J. Layland. Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment. *Journal of the Association for Computing Machinery*, 20(1):46–61, 1973.

[15] D. Mossé, S. Noh, B. Trihn, and A. K. Agrawala. Multiple Resource Allocation for Multiprocessor Distributed Real-Time Systems. In *Workshop on Parallel and Distributed Real-Time Systems (PDRTS), IEEE IPPS'93*, Newport Beach, CA, 1993.

[16] S. K. Oh and G. MacEwen. Toward Fault-tolerant Adaptive Real-Time Distributed Systems. External Technical Report 92-325, Department of Computing and Information Science, Queen's University, Kingston, Ontario, Canada, 1992.

[17] R. Rajkumar. *Synchronization in Real-Time Systems: A Priority Inheritance Approach.* Kluwer Academic Publishers, 1991.

[18] K. Ramamritham. Allocation and Scheduling of Complex Periodic Tasks. In *Proceedings* 10^{th} *International Conference on Distributed Computing Systems*, pages 108–115, 1990.

[19] K. Ramamritham, J. A. Stankovik, and W. Zhao. Distributed scheduling of tasks with deadlines and resource requirements. *IEEE Trans. on Comput.*, 38(8):1110–1123, 1989.

[20] L. Sha, R. Rajkumar, and S.S̃. Sathaye. Generalized Rate-Monotonic Scheduling Theory: A Framework for Developing Real-Time Systems. *Proceedings of the IEEE*, 82(1):68–82, 1994.

[21] H. Tokuda, C. W. Mercer, Y. Ishikawa, and T. E. Marchok. Priority inversions in real-time communication. In *Proc. IEEE Real-Time Syst. Symp.*, pages 348–359, 1989.

[22] D. Verma, H. Zhang, and D. Ferrari. Delay Jitter Control for Real-Time Communication in a Packet Switching Network. In *Proceedings of TriComm '91*, pages 35–43, 1991.

[23] L. Zhang. VirtualClock: A New Traffic Control Algorithm for Packet Switching Networks. In *SIGCOMM '90*, pages 19–29, 1990.