Resource Lock Commit Protocol (RLCP) for Multimedia Object Retrieval *

K. Selçuk Candan, Eenjun Hwang, B. Prabhakaran, and V.S. Subrahmanian

Department of Computer Science,
Institute for Advanced Computer Studies
Institute for Systems Research,
University of Maryland,
College Park, Maryland 20742.
Email: {candan,hwang,prabha,vs}@cs.umd.edu.

Abstract

Many multimedia presentation applications involve retrieval of objects from more than one collaborating server. Presentations of objects from different collaborating servers might be interdependent. For instance, we can consider distributed video servers where blocks of movies are distributed over a set of servers. Here, blocks of a movie from different video servers have to be retrieved and presented *continuously* without any *gaps* in the presentation. Such applications first need an estimate of the available network resources to each of the collaborating server in order to identify a schedule for retrieving the objects composing the presentation. A collaborating server can suggest modifications of the retrieval schedule depending on its load. These modifications can potentially affect the retrieval schedule for other collaborating applications. Hence, a sequence of negotiations have to be carried out with the collaborating servers in order to commit for a retrieval schedule of the objects composing the multimedia presentation. In this paper, we propose an application sub-layer protocol, *Resource Lock Commit Protocol (RLCP)*, for handling the negotiation and commitment of the resources required for a collaborative multimedia presentation application.

KEYWORDS: Multimedia object retrieval, multimedia presentation, resource reservation, QoS negotiation.

*This research was supported by the Army Research Office under grants DAAH-04-95-10174 and DAAH-04-96-10297,, by the Air Force Office of Scientific Research under grant F49620-93-1-0065, by ARPA/Rome Labs contract Nr. F30602-93-C-0241 (Order Nr. A716), NSF Young Investigator award IRI-93-57756, and the Army Research Laboratory under Cooperative Agreement DAAL01-96-2-0002 Federated Laboratory ATIRP Consortium.

1 Introduction

Multimedia presentation applications might involve retrieval of objects from more than one collaborating server. As an example, one can consider a distributed multimedia document presentation where objects composing the document are distributed on more than one server [7]. We can also consider a distributed video presentation where blocks of movies are distributed on more than one video server [12]. In such applications, presentation of objects from different servers can be inter-dependent. For instance, in the case of a distributed multimedia document presentation, an object $O1$ from a collaborating server $CS1$ might have to be presented at the *same* time with another object $O2$ from another server $CS2$. Here, presentation time instants for objects $O1$ and $O2$ are specified in a relative manner. If $O2$ is available for presentation only at time t_{o2}, then $O1$ will also have to be presented at t_{o2}. In the case of a distributed video presentation application, blocks of a movie from different video servers have to be presented *continuously* without any *gaps* in the presentation. Hence, the schedule for presentation of movie blocks has to be determined in such a way that the blocks are available continuously. These applications have a *flexible* presentation schedule, i.e., presentation time instants of objects are chosen *relative* to that of other objects composing the presentation. Also, objects (or blocks of a movie) composing the multimedia presentation may be replicated. In this case, a presentation application has a choice of the server from which the replicated objects can be retrieved. This choice depends on the available network resources as well as the load on the collaborating server.

Hence, such collaborative multimedia presentation applications have to determine object retrieval schedules taking into consideration the dependencies among object presentations. A presentation application determines a retrieval schedule for objects from each of the collaborating servers as follows :

- Estimate the required network resources in terms of the Quality of Service (QoS) parameters such as throughput, delay, and delay jitter. The required resources can be specified as a range of values (with maximum and minimum desirable values).
- Estimate the available network resources to each of the collaborating servers.
- Based on the available network resources, time of object presentation, duration of presentation, available buffer space (locally to the presentation application), determine the time at which an object is to be retrieved from a collaborating server.
- A collaborating server determines the required system resources (such as disk bandwidth, buffer space, and CPU processing time) for retrieving the specified objects. If the required resources are available, then the server commits for the requested retrieval. If the collaborating server is not able to commit for the requested object retrieval, then the presentation application can :
  - try another server where the object is replicated.
  - delay the retrieval of the object presentation. (We are not considering the obvious option of dropping the object from presentation as of now).

  In either case, there can be a modification of the required network resources as well as commitments from other collaborating servers.

Resource Lock Commit Protocol (RLCP) : In this paper, we propose RLCP as an application sub-layer that handles network and system resource negotiations in a phased manner. RLCP is oriented towards collaborating applications with flexible presentation schedules. RLCP uses advance resource reservation features offered by the network service provider. RLCP incorporates object retrieval scheduling algorithms for this purpose.

One of the main features of RLCP is that it handles negotiation of network resources and the system resources (such as disk bandwidth, buffer space, and CPU processing time) required at the collaborating server in subsequent phases. The reason is that object retrieval schedules can be identified only when available network resources are known. A server handling object retrieval can determine the required system resources (such as disk bandwidth, buffer space, and CPU processing time) only when objects to be retrieved are known. This is due to the possibility that multimedia objects can be striped over different disks on a server. Also, in some instances, objects might have to be realized through some format conversions. Hence, in the first phase, RLCP identifies available network resources. In the second phase, RLCP identifies the objects that are to be retrieved from different collaborating servers and negotiates the system resources required for object retrieval. This two-phase negotiation of required network and system resources have to be carried out with all the collaborating servers. RLCP can finalize the entire negotiation process only after reaching an agreement with all the collaborating servers. If there is any disagreement with one or more collaborating servers, RLCP might have to modify the required resources from other collaborating servers as well. Hence, RLCP executes a third phase of resource commitment or modification.

Organization of the paper : In the following section, we discuss the features of the collaborating applications towards which RLCP is oriented. In section 3, we emphasize the need for addressing the dependencies among object presentations before reserving the required resources. In section 4 and 5, we describe the Resource Lock Commit Protocol and its features. In section 6, we present an example application that uses RLCP. We compare the features offered by RLCP with those of other application sub-layer protocols in section 7.

2 Distributed Multimedia Presentation Applications

In this section, we consider two applications : (i) presentation of a distributed multimedia document and (ii) distributed video presentation. Both these applications carry out multimedia presentations based on objects retrieved from multiple servers.

2.1 Distributed Multimedia Document Presentation

Let us consider presentation of a multimedia document in which the objects composing the document are distributed over a set of collaborating servers $CS1$ to $CS4$, as shown in Figure1. The presentation involves objects $O1$ to $O4$ with presentation times and durations as shown in Figure 2(b). We assume that $O2$ can be realized (through some format conversions) from objects $O2'$ or $O2''$. However, this realization involves additional CPU processing time. Let us assume that the presentation of objects have to be done according to the following constraints :

- Object presentations can be started anywhere within a specified interval. For example, $O1$ can be presented at any point in the interval $t_{o1} - \delta t, t_{o1} + \delta t$.
- Objects $O1$ and $O2$ have to be presented simultaneously, i.e., t_{o1} and t_{o2} should be the same. In a similar manner, objects $O3$ and $O4$ have to be presented simultaneously.
- Objects $O3$ and $O4$ have to be presented *immediately* after the completion of the presentation of objects $O1$ and $O2$.

The application running in system S, carrying out the presentation, has to download the objects from the appropriate collaborating servers. For this purpose, the application has to create a retrieval schedule that

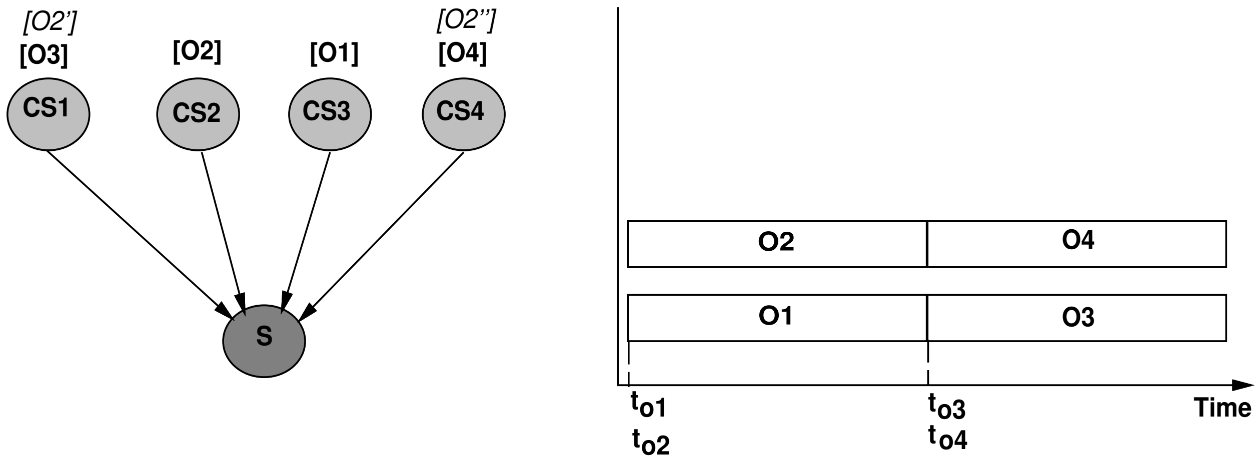

Figure 1: **Distributed Multimedia Document Presentation**

identifies the time instants at which objects are to be retrieved from the collaborating servers. This retrieval schedule has to be agreed upon by the applications running at the collaborating servers as well. If any of the collaborating servers does not agree to the schedule, then the application in system S might have to modify the retrieval schedule. For instance, let us assume that $CS2$ defers delivery of object $O2$ by Δt. Then retrieval of $O1$ might have to be deferred by a similar time interval (assuming that system S is not able to buffer the object for the time period). Alternatively, $O2$ can be realized from $O2'$ or $O2''$. If $CS1$ or $CS4$ can support the conversion of object into the required format of $O2$, then S can request one of these collaborating servers to download the object. Otherwise, presentation of objects $O1$ and $O2$ have to be deferred. Since the retrieval of objects $O3$ and $O4$ depends on the presentation of $O1$ and $O2$, their retrieval might also be deferred. Hence, there is a dependency in the retrieval schedules of the objects composing the multimedia document.

2.2 Distributed Video Presentation

We can consider a distributed video presentation application where movies may not be stored entirely on a particular server [12]. Blocks of the same movie may be replicated and stored on many Video-on-Demand (VoD) servers on the network. A VoD server caters to a set of customers at any point in time. In the case of the requested movie being not available in the server, the server tries to get blocks of the movie from other VoD servers.

Figure 2(a) shows a server S serving a customer $c1$. Let us assume that the requested movie has 25 blocks distributed over this server (S) and other VoD servers ($CS1$ to $CS4$). Blocks $b16$ to $b20$ are assumed to be replicated (as shown in italics). The server S has to download blocks $b6$ to $b25$ from servers, in order to serve the customer's request. Before delivering the requested movie to the customer $C1$, the server S has to create a retrieval schedule for the blocks $b6$ to $b25$. This retrieval schedule has to ensure that there is no *gap* in the movie presentation for the customer.

Blocks $b1 - b6$ are available local to the server S and hence can be shipped to the customer directly. The server S has to get a commitment from other VoD servers for downloading the missing blocks as follows : $b6 - b10$ from $CS1$, $b11 - b15$ from $CS3$, $b16 - 20$ from $CS2$ and $b21 - b25$ from $CS4$. For downloading the blocks from the VoD servers, the server S has to specify the time at which the blocks are needed by S. In case, a VoD server is not able to commit for the download at the requested time, the server S can either try another VoD server or request the same VoD server for another commitment time. In Figure 2(b), let us

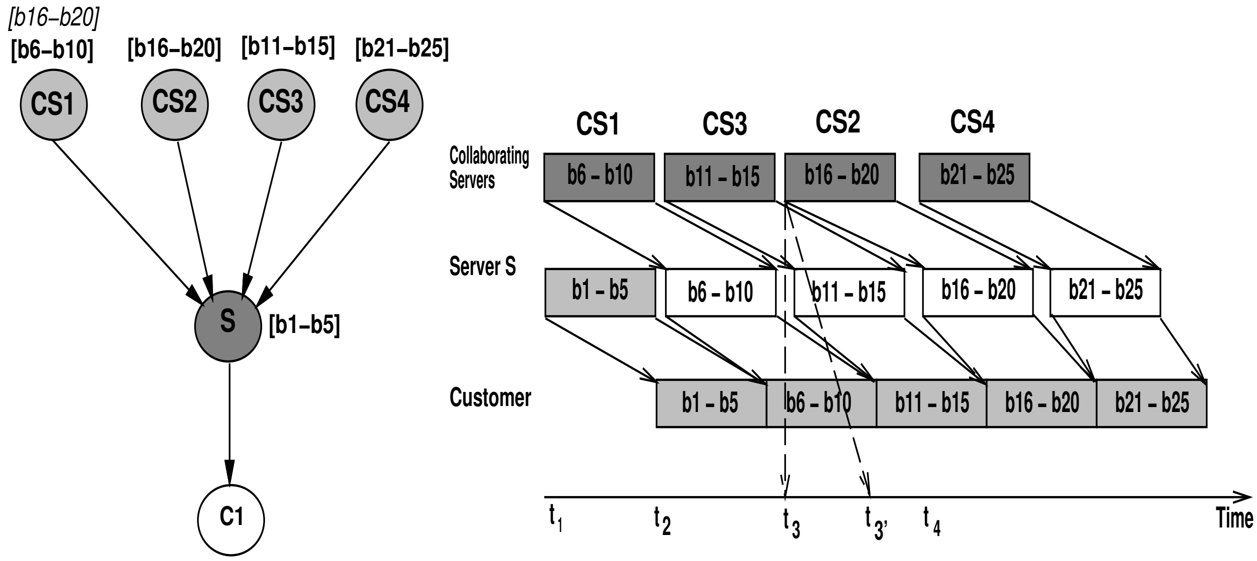

Figure 2: Example 2 : Distributed Video Presentation

assume that the VoD server $CS2$ is not able to commit at the requested time t_3. Instead, it is able to commit for the blocks $b16 - b20$ at time $t_{3'}$. The server S has two options :

- Request server $CS1$ to provide blocks $b16 - b20$. If $CS1$ is able to commit for the blocks at t_3, then S can proceed with the same retrieval schedule.
- Otherwise, S has to identify the minimum delay involved in downloading the blocks $b16 - b20$ from either $CS1$ or $CS2$. Assuming that $t_{3'}$ is the earliest time at which the blocks are available, then S has to delay the movie presentation by the time difference δt $(= t_{3'} - t_3)$. In this case, S might have to reschedule requests for the previous blocks as follows : $b6 - b10$ at $t_1 + \delta t$, $b11 - b15$ at $t2 + \delta t$ and $b21 - b25$ at $t4 + \delta t$.

In this example also, there is a dependency among the objects (blocks of a movie) that are to be retrieved.

3 Need for Addressing Dependencies Among Resources

The example application scenarios discussed above, show dependencies among network and system resources as follows.

- Retrieval schedule for the objects composing a multimedia presentation depends on the available network resources to each of the collaborating servers.
- Also, there may be a dependency among the objects that are to be retrieved from the different collaborating servers. Hence, a retrieval schedule for a multimedia presentation needs agreement from more than one participating collaborating servers. In the case where, one or more collaborating servers are not able to commit for the requested schedule, the retrieval schedule might have to be modified. This modification of the retrieval schedule might mean modification of the agreements with other collaborating servers.

- Objects composing a multimedia presentation might have to be processed by collaborating servers, say for format conversions. Objects might be striped over different disks in the server. Hence, until the retrieval schedule is *finalized* by the presentation application (running in S in Figure 1 and 2), the collaborating servers will not be able to identify the resources (e.g., disk bandwidth, buffer space, and processing time) required for delivering the multimedia objects.
- Replication of objects composing a multimedia presentation provides the application a *choice* of the collaborating server(s) from which the replicated objects can be retrieved. Hence, in the case of a collaborating server denying an object retrieval request, the presentation application can try another source for the same object.

Hence, the retrieval schedule to be determined by the presentation application depends on the following factors :

- Object presentation specifications such as time instants and duration of presentation.
- Available network QoS such as throughput, delay, and delay jitter.
- Local system (where the presentation application is being executed) resources such as available buffer space, disk bandwidth, and processing time.
- Available system resources such as buffer space, disk bandwidth, and processing time at the collaborating servers.

4 Resource Lock Commit Protocol

In this section, we propose a three-phase application sub-layer protocol called *Resource Lock Commit Protocol (RLCP)* for creating retrieval schedules for collaborating applications (such as distributed document presentation and distributed video presentation) involving dependent resources. Such applications are assumed to have a flexible presentation schedules , i.e., presentation time instants of objects are chosen *relative* to that of other objects composing the presentation. RLCP assumes advance resource reservation features to be offered by the network service provider. RLCP is a receiver initiated protocol (as opposed to a sender initiated one). RLCP proceeds in the following phases.

- **Network resource negotiation phase :** Here, RLCP identifies and *locks* network resources to each of the collaborating servers.
- **System resource negotiation phase :** In this phase, RLCP identifies schedule for retrieving objects composing the multimedia presentation based on the available network resources. Using this retrieval schedule, RLCP negotiates the required system resources (i.e., disk bandwidth, buffer space, and CPU processing time) from each of the collaborating servers.
- **Commitment or Modification Phase :** As discussed before, agreement from all the collaborating servers is required for finalizing the retrieval schedule. If all the collaborating servers agree to the object retrieval requests, RLCP can commit to the use of the requested network and system resources. If any of the collaborating servers disagree, RLCP might have to modify the requested resources to (some or all of) the servers. This commitment or modification of the requested resources (both system and network) is carried out in the third phase.

We assume the notion of *lock* and *commit* of a resource (such as network throughput or disk bandwidth) as follows. *Locking* of a resource makes a *temporary* reservation of the required resource whereas *committing*

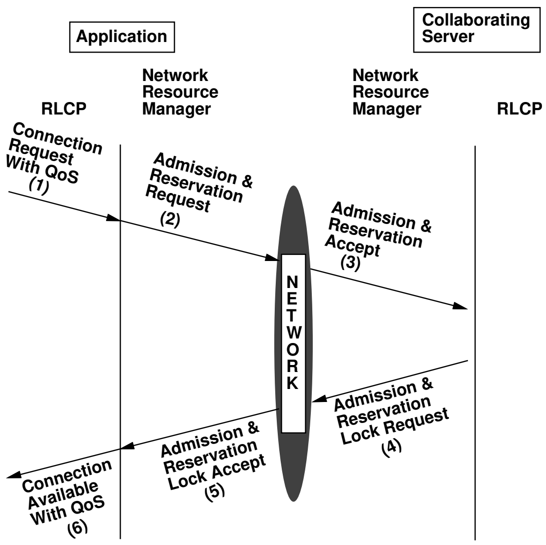

Figure 3: **Phase 1 of RLCP** *(in "accept" mode)*

for a resource can be considered as an agreement to use the resource at the requested time (and for the requested duration). Locked resources can be released after a time out period, if no commitment is made by the requesting application. *Modification* of the requested resources can be in terms of the quantity or the required time.

4.1 Different Phases of RLCP

Now, we discuss the three different phases in which RLCP proceeds to negotiate network and server resources for creating a multimedia presentation.

Network resource negotiation phase : In the first phase, RLCP requests the network service provider to identify the available network resources for communication with each of the collaborating servers. RLCP can make an initial estimate of the required network resources using some heuristic algorithms such as the ones discussed in [8]. The network resources are also locked (temporarily reserved) for communication with the collaborating servers. The message exchange (with "accept" of connection request) in the first phase is shown in Figure 3. The application issues a request to the network service provider for estimating the available QoS to a collaborating server (event 1). The network service provider makes an *admission and reservation request* to the network (event 2). If the requested network resources are available, the network issues an *admission and reservation accept* to the peer network service provider at the collaborating server (event 3).

The peer network service provider issues an *admission and reservation lock request* to the network for locking the available resources temporarily (event 4). The network locks the resources for the connection and issues an *admission and reservation lock accept* to the initiating network service provider (event 5). The initiating network service provider then informs the application that a connection to the collaborating server is available with the locked QoS (event 6). If the requested network resources are not available, then the request will be rejected by the network (not shown in Figure 3). The interactions discussed here are for advance resource reservation.

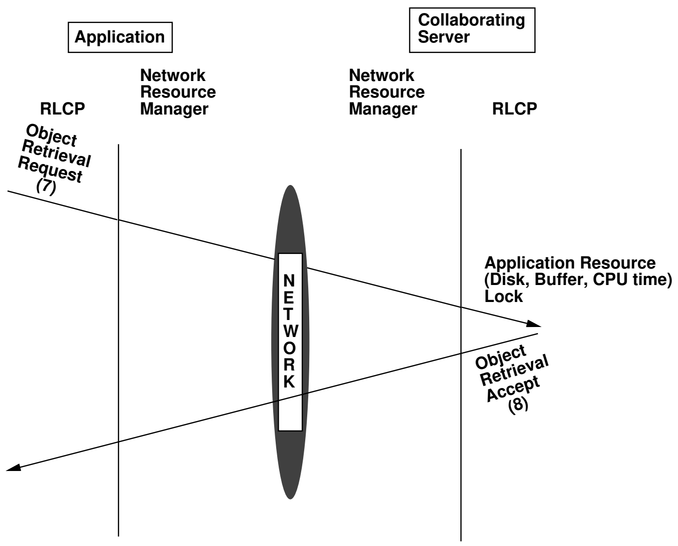

Figure 4: **Phase 2 of RLCP***(in "accept" mode)*

System resource negotiation phase : In the second phase, the initiating RLCP computes a retrieval schedule based on the available network resources. This retrieval schedule identifies : (i) the collaborating server from which each object is to be retrieved, and (ii) the time at which each object is to be retrieved. Algorithms for generating retrieval schedules for different multimedia presentation applications are discussed in [2, 8, 7, 12]. These algorithms can be appropriately incorporated in the RLCP for computing retrieval schedules based on the available network resources. The initiating RLCP then issues an *object retrieval request* to the collaborating server (event 7, in Figure 4). This request identifies the objects that are to be retrieved from the collaborating server as well as the times at which the objects are required. The collaborating server identifies the system resources (such as disk bandwidth, buffer, and processing time) required for honoring the requested schedule. If the requested resources are available, they are locked for use by the presentation application. Then, the collaborating server issues an *object retrieval accept* to the initiating application (event 8). If the requested resources are not available due to prior commitment, then the server issues an *object retrieval reject*. (If the requested resources are locked but not committed, then the server can indicate it to the requesting application so that it can retry at a later point in time. One can also think in terms of giving a *priority* to this request by considering it as the next candidate for the requested resources if the locked resources are released).

Commitment or modification phase : If all the collaborating servers agree to provide the requested resources for objects retrieval, then RLCP can *commit* to use the locked resources. If any of the collaborating servers modify the resource request (by rejecting or by suggesting changes according to the available resources), then RLCP might have to modify the requested resources to some or all of the collaborating servers. This modification is decided using retrieval scheduling algorithms [7]. As discussed earlier, the modification can be one of the following.

- Request another collaborating server, if object(s) is(are) replicated. This implies an increase in the resource requirements (at network and application level) for the new collaborating server.
- Request the object(s) at a modified time, when the load on the collaborating server is lower. This implies that resource commitments for other collaborating servers might have to delayed appropriately.

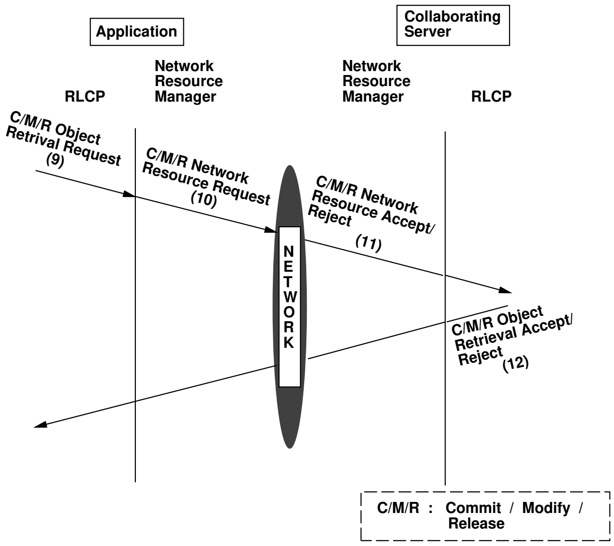

Figure 5: **Phase 3 of RLCP**

For handling the above modifications, RLCP provides a third modification phase as shown in Figure 5. In this phase, the presentation application checks the commitment offered by the collaborating servers. If all the servers commit to the requested schedule, then RLCP sends resource commit request to all the collaborating servers (events 9-12). If any of the servers is not able to commit for the requested schedule, then the presentation application initiates a *modify object retrieval request* (event 9). RLCP also computes the modifications required in the allocated network resources. The modifications can be in terms of (i) the quantity of the required resources, (ii) time at which the resources are required, or (iii) both. These modifications go as *modify network resources request* to the network service provider (event 10). If the network is able to accept the modifications, then it issues *modify network resources accept* to the peer network service provider (event 11). If the collaborating server is able to honor the request, it issues a *modify object retrieval accept* (event 12). If any of locked resources (at network or application level) are not required in the modified retrieval schedule, the presentation application can release them explicitly (events 9-12).

As can be imagined, phase 3 might have to be executed a few times before the desired retrieval schedule is finalized. The number of iterations for finalizing the retrieval schedule (and hence the number of executions of phase 3) is very much dependent on the type of multimedia presentation application (i.e., the number of objects composing the presentation, the number of collaborating servers, and the number of servers on which objects are replicated). It also depends on the type of retrieval scheduling algorithms used. Appendix I carries a brief discussion on identifying the number of iterations for finalizing the retrieval schedule of an example multimedia presentation using algorithms discussed in [7].

Handling Retrieval of Replicated Objects : Retrieval of replicated objects can be handled in different ways :

- RLCP can make an estimate of the available network resources to *all* the servers and choose the one that suits the presentation schedule (and the availability of local system resources, such buffer space). This approach has the advantage from the application point of view in that it can select the best source for an object. However, it leads to an increase in traffic at the network level.

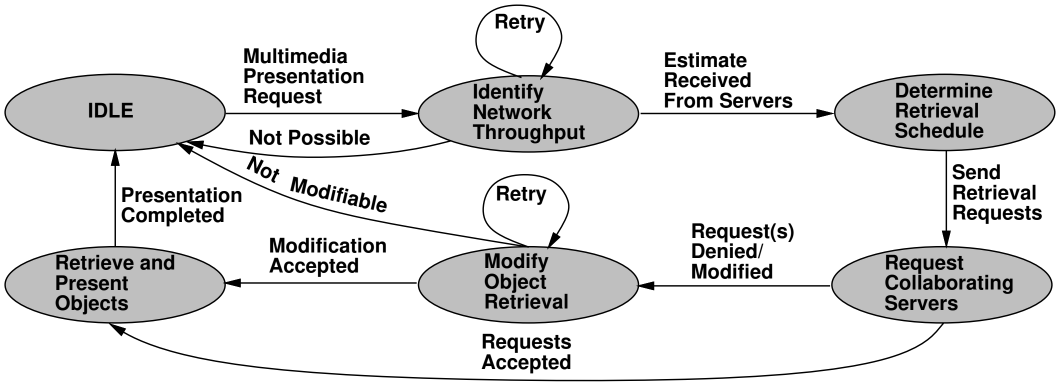

Figure 6: **RLCP Initiator Finite State Machine**

- Alternatively, RLCP can make an estimate to *one* of the servers (chosen using some heuristics, say, load distribution on the collaborating servers). This approach minimizes the network traffic but the chosen object source may not be the best one. (RLCP has to try other servers one by one, if the chosen server is not able to handle the request)

Depending on the implementation, application can request the RLCP to exercise one of the available options for retrieving replicated objects.

4.2 RLCP and its Architectural Position

Figure 6 shows the finite state machine (FSM) associated with the RLCP that initiates a multimedia presentation. When a multimedia presentation is executed, RLCP makes an initial estimate of the required network resources. Then, the RLCP initiator moves to the state of identifying network throughput. After getting the QoS available to the required collaborating servers, RLCP determines the retrieval schedule and sends out object retrieval requests. If all the collaborating servers agree for the retrieval requests, then the multimedia presentation can be initiated (as denoted by the state *Retrieve and Present Objects*). Otherwise, RLCP goes through a phase of modifying the object retrieval schedule (as denoted by the state *Modify Object Retrieval*). The FSM associated with RLCP of the collaborating server is shown in Figure 7. If the object retrieval request can be honored, the collaborating RLCP moves to the state *Resources Locked.* When a request for modifying an object retrieval schedule is received, the collaborating RLCP stays in the same state after sending a message to the initiator regarding the acceptance or rejection of the request. When the initiator commits to the retrieval schedule, the collaborator moves to the state *Object Retrieval Commit.* After the retrieval of objects is completed, the collaborating RLCP moves to the *Idle* state.

RLCP is architecturally located at the application sub-layer, as shown in Figure 8. This implies that RLCP can use any resource reservation protocol that provides advance resource reservation capabilities. Only modification that will be necessary is that during the connection request time the resources need not be allocated, only locked.

5 RLCP Services

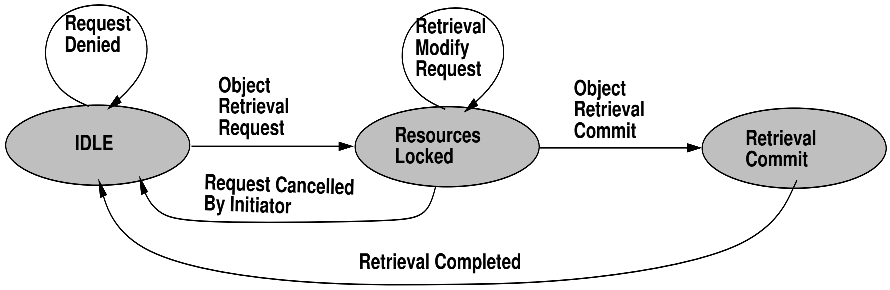

Figure 7: **RLCP Collaborator Finite State Machine**

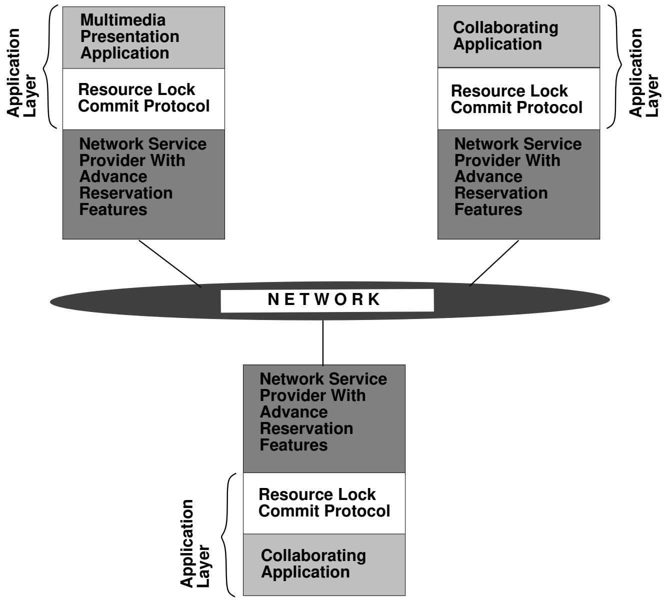

Figure 8: **Architectural Position of RLCP**

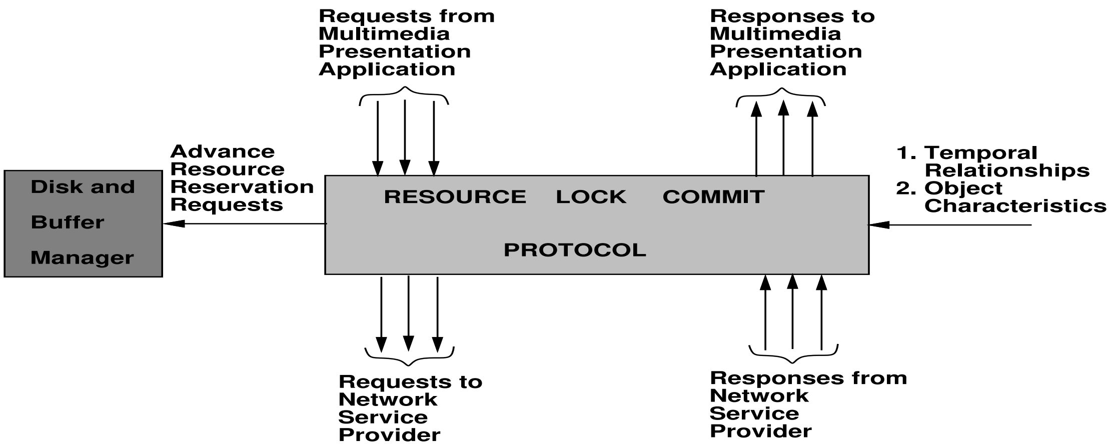

Figure 9: **Services Offered By RLCP**

	Service	Parameters
Requests	R_InitiatePresentation	Presentation Configuration
	R_PresentationDelay	Accept/Reject
Responses	R_RejectPresentation	Reason
	R_DelayStart	Time

Table 1: RLCP Services to Multimedia Presentation Applications

	Service	Parameters
Requests	R_SendObject	ObjectId, TimeRequired
	R_ModifyObjectSendTime	ObjectId, TimeRequired
Responses	R_SendObjectResponse	Accept, Reject, Delay
	R_ModifySendTimeResponse	Accept, Reject, Delay

Table 2: Services For Interacting With Peer RLCP

As shown in Figure 9, RLCP takes as input the temporal relationships and characteristics of the objects (such as their sizes) composing the multimedia presentation that is to be initiated. It interacts with the network service provider, disk, and buffer manager for reserving the required resources.

Interaction With a Multimedia Presentation Application : The interface provided by the RLCP to a multimedia presentation application is in terms of a set of requests and a set of responses, as summarized in Table 1. Here, an application can initiate a multimedia presentation using *R_InitiatePresentation* request. RLCP uses one of the retrieval schedule generation algorithms discussed in [2, 8, 7, 12], in order to identify the object retrieval times and the collaborating servers from which objects are to be retrieved. If the start of the presentation has to be delayed or if the presentation cannot be initiated, RLCP responds with *R_DelayStart* or *R_RejectPresentation*. Application can make use of the request *R_PresentationDelay* to convey its acceptance or rejection of the delay in initiating the presentation. Other services can be incorporated into RLCP for handling user interactions during the multimedia presentation and for handling dynamic network behavior, similar to the services offered in [11].

Interaction With Peer RLCP : For carrying out a multimedia presentation, RLCP interacts with peer collaborating RLCPs. The requests made to collaborating RLCPs and their responses are summarized in Table 2. Initiating RLCP makes requests for object retrievals from the collaborating RLCP using the services *R_SendObject* and *R_ModifyObjectSendTime*. The collaborating RLCP responds with *R_SendObjectResponse* and *R_ModifySendTimeResponse*. The collaborating RLCP interacts with disk and buffer managers in order to reserve the bandwidth and buffer space required for the requested object retrieval.

Interaction With Network Service Provider : Interaction of RLCP with the network service provider is in terms of :

- Requests for network connections with required QoS at the specified for a given duration.
- Modification of the requested time and duration of a network connection.

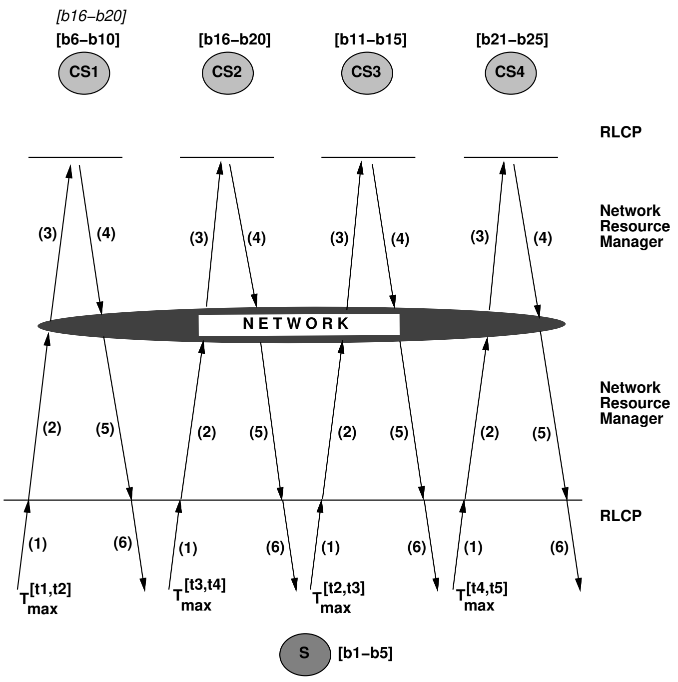

Figure 10: **Requesting Network Resources to Collaborating Servers**

- Confirmation for using the requested resources from the network service provider.

6 Using Resource Lock Commit Protocol : An Example

Let us consider the distributed video presentation example discussed in Section 2.2. Server S has to identify a retrieval schedule for presenting a movie at time t_0 for a customer $c1$. It has to download the movie blocks $b6 - b25$ from collaborating VoD servers $CS1, CS2, CS3$ and $CS4$. As shown in Figure 10, in the first phase, server S requests the network service provider to give an estimate of the available network resources. Based on the available network resources, S creates an *initial* retrieval schedule. (In this example, we assume that RLCP contacts servers with replicated objects one after another). Let us assume that the initial schedule is as follows :

- Blocks $b1 - b5$ are available locally. Movie presentation starts at time t_0. Based on the initial start time and the playing time of the previous blocks, the retrieval schedule for other blocks are determined as follows.
- Blocks $b6 - b10$ to be delivered by $CS1$ at time t_1.
- Blocks $b11 - b15$ to be delivered by $CS3$ at time t_2.
- Blocks $b16 - b20$ to be delivered by $CS2$ at time t_3.
- Blocks $b21 - b25$ to be delivered by $CS4$ at time t_4.

The server S identifies and requests for network resources to the collaborating servers, as shown in Figure 10. Then, S makes object retrieval requests to the collaborating servers based on the above initial schedule,

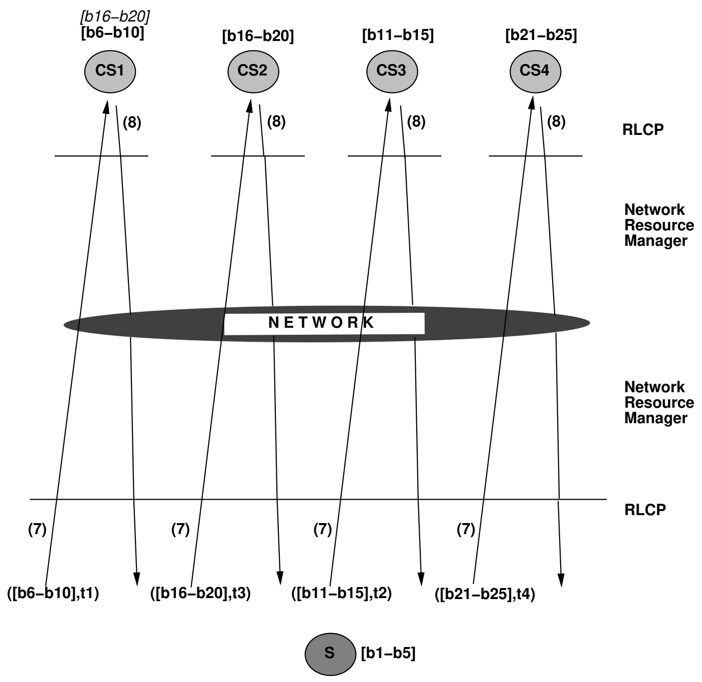

Figure 11: **Making Object Retrieval Requests**

as shown in Figure 11. If all the collaborating servers are able to allocate the required resources for object retrieval, then they issue object retrieval commits to S. The locked network resources for the video presentation are also allocated, as shown in Figure 11.

As discussed in Section 2.2, let us assume that $CS2$ is not able to commit for the blocks $b16 - b20$ at time t_3. Instead, it is able to commit for the blocks at $t_{3'}$. Then, S can try to check whether $CS1$ can provide the blocks at t_3. If $CS1$ can commit for the blocks at t_3, then S can finalize the retrieval schedule for the movie presentation. Otherwise, S has to delay the movie presentation by δt $(= t_{3'} - t_3)$. This implies that commitments for other movie blocks also have to be delayed by δt. Hence, S executes phase 3 of RLCP, as shown in Figure 12. Here, request for movie blocks are modified as follows : $b6 - b10$ at $t_1 + \delta t$, $b11 - b15$ at $t_2 + \delta t$, $b16 - b20$ at $t_{3'}$ and $b21 - b25$ at $t_4 + \delta t$. Modifications required for the network resources are also computed and appropriate requests are issued. Figure 12 shows acceptance of the request for modification of the requested resources. Hence, S can finalize the retrieval schedule as the desired one.

7 Comparison With Existing Protocols

Handling of negotiation of required resources for multimedia applications have mostly been dealt at the transport protocol level [1]. Requirements at the application level makes resource negotiation a more complex task. Few protocols have been suggested for handling resource negotiation and object retrievals in distributed multimedia applications :

- Application and Network Synchronization Protocol (ASP and NSP) [2].
- Multimedia Application Protocol (MMAP) [11].

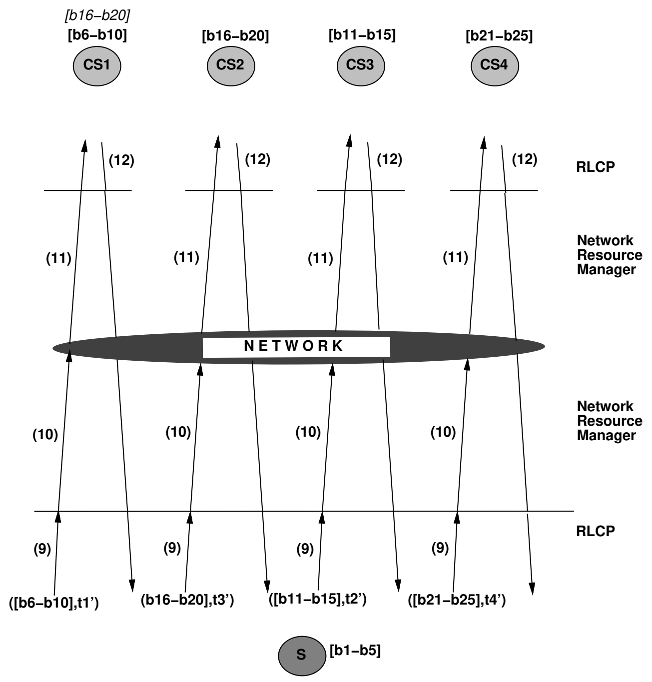

Figure 12: **Modifying the Requested Resources**

- Negotiation and Resource Reservation Protocol (NRP) [13].

Application and Network Synchronization Protocol : In [2], a set of protocols - one at application service level (*Application Synchronization Protocol (ASP)*)and another at the network service level (*Network Synchronization Protocol (NSP)*) - have been proposed for providing synchronization of objects as a network service. Application Synchronization Protocol (ASP) is proposed for providing specific services pertaining to the retrieval of complex multimedia objects from multiple sources across a network for playout at a single site. This protocol has an associated Network Synchronization Protocol (NSP) which provides a general network interface and functionality for many applications. These protocols are aimed at synchronizing object presentations rather than handling resource negotiations for multimedia presentation applications.

Multimedia Application Protocol (MMAP) : The main emphasis of MMAP is that computing the QoS requirements, negotiating it with the network service provider and handling the dynamic behavior of the user and the network service provider can be provided as a generic protocol service interface [11]. The services provided by MMAP addresses these issues and hence MMAP services can be used by an orchestrated presentation application. However, MMAP does not handle dependencies among object presentations and does not address the collaborative nature of multimedia presentations discussed in this paper.

Negotiation and Resource Reservation Protocol (NRP) : has been proposed in [13] for describing application level QoS with an end-to-end view spanning the whole distributed application. NRP carries out negotiation based on client specified QoS value ranges and given resource availabilities on endsystems and communication links. NRP helps in translating Application-QoS specified in terms of media specific characteristics (such as video frame size : 480 * 360) into network QoS. It proceeds in three phases : (i)

Mapping Application-QoS to network level QoS, (ii) Fixing QoS values for each communication link and (iii) Relaxing resource reservation according to final QoS values. NRP helps in matching device capabilities (such as video frame sizes) to available network resources. In our approach, we propose RLCP to help in finalizing retrieval schedules for objects from more than one collaborating server. The negotiation of the schedule takes into consideration the dependencies among object presentations. Hence, the purpose of RLCP is different from that of NRP.

8 Summary

Negotiating and reserving the resources required by a multimedia application at the network as well as application level is a complex task. This task is very much application dependent. In this paper, we consider an interesting class of multimedia application : presentation of media objects distributed in a set of collaborating servers. A key requirement of these applications is the creation of a retrieval schedule that will help in identifying the objects to be downloaded from each collaborating server at the desired time. This retrieval schedule depends on the availability of resources at the network and the application level. Hence, determining a retrieval schedule implies negotiation and reservation of resources at network and application level.

We propose RLCP as an application sub-layer protocol for handling creation of retrieval schedules for distributed multimedia presentation applications. RLCP helps in resource reservation at network and application level. RLCP proceeds in a phased manner, by estimating the available network resources first. It then computes an initial retrieval schedule for each of the collaborating server based on the available network QoS. This retrieval schedule is propagated to the collaborating server for validation in the second phase. In this phase, the required system resources (such as disk bandwidth, buffer space, and CPU processing time) are also identified and locked. If the schedule is accepted by all the collaborating servers, RLCP finalizes the schedule. Otherwise, it executes a sequence of resource modification phase till the retrieval schedule can be finalized. RLCP can use services offered by any network protocol supporting advance reservation capabilities such as Tenet suite of protocols [4] or ST2+ [10].

References

[1] D. Ferrari (1990) *Client Requirements For Real-Time Communication Services*, IEEE Communication Magazine, Vol. 28, No. 11, Nov. '90, pp. 65-72.

[2] T.D.C. Little and A. Ghafoor (1991) *Multimedia Synchronization Protocols for Broadband Integrated services*, IEEE J. on Selected Areas of Communications, vol. 9, no. 9, Dec. 1991, pp. 1368-1382.

[3] L. Zhang, S. Deering, D. Estrin, S. Shenker, D. Zappala, "RSVP: A New Resource ReSerVation Protocol", IEEE Network, September 1993, pp.8-18.

[4] D. Ferrari, *et al*, "Network Support for Multimedia : A Discussion of Tenet Approach", Computer Networks and ISDN Systems, Special issue on Multimedia Networking, 1994.

[5] K. Nahrstedt and Ralf Steinmetz, "Resource Management in Networked Multimedia Systems", IEEE Computer, Vol. 28, No. 4, April 1995.

[6] S.V. Raghavan, B. Prabhakaran and Satish K. Tripathi "Synchronization Representation and Traffic Source Modeling in Orchestrated Presentation", IEEE Journal on Selected Areas in Communication, special issue on Multimedia Synchronization, Vol. 14, No. 1, January 1996.

[7] K. S. Candan, B. Prabhakaran, and V. S. Subrahmanian, "CHIMP : A Framework for Supporting Multimedia Document Authoring and Presentation", Proceedings of ACM Multimedia '96 Conference, Boston, November 1996.

[8] S.V. Raghavan, B. Prabhakaran and Satish K. Tripathi *Quality of Service Considerations For Distributed, Orchestrated Multimedia Presentation*, Proceedings of High Performance Networking 94 (HPN'94), Paris, France, July 1994, pp. 217-238. Also available as Technical Report: CS-TR-3167, UMIACS-TR-93-113, University of Maryland, College Park, Computer Science Technical Report Series, October 1993.

[9] K. Nahrstedt and J. Smith, "The QoS Broker", IEEE Multimedia, Vol. 2, No. 1, pp. 53-67, Spring 1995.

[10] L. Delgrossi and L. Berger, "Internet Stream Protocol Version ST2+", RFC 1819, August 1995.

[11] S.V. Raghavan, B. Prabhakaran and Satish K. Tripathi, "Handling QoS Negotiations in Distributed Orchestrated Presentation", to be published in Journal of High Speed Networking.

[12] E. Hwang, B. Prabhakaran, and V.S. Subrahmanian, "Presentation Planning for Distributed Video Systems", CS TR 3723, Umiacs TR 96-91, University of Maryland, College Park, Computer Science Technical Report Series, December 1996.

[13] G. Dermler, W. Fiederer. I. Burth, and K. Rothermel, "A Negotiation and Resource Reservation Protocol (NRP) for Configurable Multimedia Applications", Proceedings of IEEE Multimedia '96 Conference, Japan, pp. 113-116.

Appendix I

As discussed earlier in the paper, the number of iterations for finalizing the retrieval schedule (and hence the number of executions of phase 3) is very much dependent on the type of multimedia presentation application (i.e., the number of objects composing the presentation, the number of collaborating servers, and the number of servers on which objects are replicated). It also depends on the type of retrieval scheduling algorithms used. In this section, we provide an example where an estimate can be made regarding the worst case number of modification requests made by Phase 3 of RLCP. Here, we consider the retrieval scheduling algorithm discussed in [7].

Let a multimedia document D contain two objects $o1$ and $o2$ with presentation schedules as shown in Figure 13. In this figure, st and et denote the start and the end times of the object presentations respectively. Let $req(o)$ denote the identified retrieval schedule for an object in the second phase of RLCP. During each interval shown in Figure 13, the following conditions must hold.

- The total size of the objects stored at the local buffers must be less than the local buffer size.
- The required throughput for each object must fit into the throughput identified in the first phase of the RLCP protocol for that object.

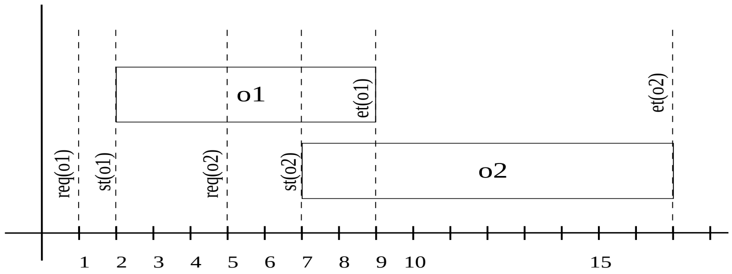

Figure 13: Schedule validation process

The following algorithm uses feedbacks to make the necessary modifications in either the presentation or the retrieval schedule, as appropriate[7] :

Algorithm:

1. Sort the events (the vertical dashed-lines in the figure), and identify the number of intervals (num_{int}).
2. Set the borders of the intervals as **unmarked**. (When a border is **marked**, the event that corresponds to the border can not be changed).
3. $Satisfied = True$;
4. $\forall_{FirstInterval \leq i \leq LastInterval}(FailNo[i] = 0)$;
5. $ThisInterval = LastInterval$;
6. **while** $Satisfied$ and ($ThisInterval \geq FirstInterval$) **do**
   (a) Check if the interval $ThisInterval$ is *valid*
   (b) **while** $ThisInterval$ is not *valid* **do**
      i. $FailNo[ThisInterval] = FailNo[ThisInterval] + 1$;
      ii. **if** $FailNo[ThisInterval] > MaxFails$ **then** return empty schedule
      iii. Create k alternative feedbacks, each modifying either the presentation schedule or the retrieval schedule (here we do not discuss how the feedback is generated). Note that the values that are already **marked** must be kept constant in feedbacks.
      iv. Send all feedbacks to the *schedule modifier*.
      v. **while** *schedule modifier* returns no schedule and ($ThisInterval < LastInterval$) **do**

A. $ThisInterval = ThisInterval + 1$;

B. $FailNo[ThisInterval] = FailNo[ThisInterval] + 1$;

C. **if** $FailNo[ThisInterval] > MaxFails$ **then** return empty schedule

D. Set the borders of the interval $ThisInterval$ as **notmarked**

E. Create k alternative feedbacks. Note that the values that are already **marked** must be kept constant in feedbacks.

F. Send feedbacks to the *schedule modifier*.

vi. Check if the interval $ThisInterval$ is *valid*

(c) **if** interval $ThisInterval$ is *valid* **then**

i. Set the borders of the interval as **marked**

ii. $ThisInterval = ThisInterval - 1$

(d) **else**

i. $Satisfied = False$

7. **if** $Satisfied$ **then** return the schedules

8. **else** return empty schedule

This algorithm starts from the last interval and proceeds towards the first interval. It generates feedbacks whenever either the buffer or the throughput conditions is not satisfied, and modifies the presentation and retrieval schedules accodingly. The details of the algorithm can be found in [7]. Here, we will not go into the details of the algorithm, but we note that each interval in the schedule gets modified at most $MaxFails$ times, as the algorithm proceeds (steps 6(b) : (ii) and (iii)).

In general, let a document D contain n objects $(o_1 \ldots o_n)$. Then, the presentation and retrieval schedules of the above algorithm would have at most $3 \times n$ intervals. Hence, the above algorithm would issue $3 \times MaxFails \times n$ feedback creation requests before it halts. Each feedback creation causes k alternative feedbacks. If $rep(o_i)$ is the number of servers in which an object o_i is replicated, then each alternative feedback causes at most $\sum_{1 \leq i \leq n} rep(o_i)$ modification messages to the servers. Since the retrieval scheduling algorithm generates atmost $3 \times MaxFails \times k \times n$ feedbacks, the number of modification requests generated by RLCP in the third phase will be atmost

$$3 \times MaxFails \times k \times n \times \sum_{1 \leq i \leq n} rep(o_i).$$

Note that, if $rep(o_i)$ is bounded by some constant, i.e. it is in $O(1)$, for all objects $(o_1 \ldots o_n)$ in the document, then the number of modification requests is in $O(n^2)$.