Replication for a Distributed Multimedia System

Giwon On[1], Michael Zink[1], Michael Liepert[1], Carsten Griwodz[1], Jens B. Schmitt[1], Ralf Steinmetz[1,2]

[1]KOM-Industrial Process and System Communications
Darmstadt University of Technology
Merckstrasse 25
64283 Darmstadt, Germany
0049-6151-166162

[2]IPSI, German National Research Center for
Information Technology
Dolivostrasse 15
4293 Darmstadt, Germany
0049-6151-869869

Abstract

Replicating data and services at multiple networked computers increases the service availability of distributed systems. This paper presents the design and implementation architecture of a replication mechanism for a distributed multimedia system medianode which is currently developed as an infrastructure to share multimedia-enhanced teaching materials among lecture groups. With the replication mechanism, medianode provides enhanced access to presentation materials in both connected and disconnected operation modes. The main contribution of this paper is the identification of new replication requirements in distributed media systems and a multicast-based update propagation mechanism by which not only the update events are signaled, but also the updated data are exchanged between replication managers.

1. Introduction

Replication is the maintenance of on-line copies of data and other resources[2,5,6]. Replication of presentation materials and meta-data is an important key to providing high availability, fault tolerance and quality of service (QoS) in distributed multimedia systems. For example, when a user requires access (read/write) to a presentation material which comprises audio/video data and some resources which are not available in the local machine at this point of time, a local replication manager copies the required data from their original location and puts it into either one of the machines located nearby or the local machine without requiring any user interaction (user transparent). This function enhances the total performance of the distributed system, in this example, the presentation service system, by reducing the response delay that is often caused due to insufficient system resources at a given service time. Furthermore, because of the available replica in the local machine, the assurance that users can continue their presentation in a situation of network disconnection, is significantly higher than without replica.

The main contributions of this paper are (1) to identify the new replication requirements for distributed multimedia systems, and (2) to build a replication mechanism for distributed multimedia systems. To achieve these targets, we first study the characteristics of presentational media types which are handled in medianode system[1], and extract new replica units and granularities which have neither been considered nor supported in existing replication mechanisms. Furthermore, we give a survey on existing replication mechanisms and identify their features and limitations. By prototyping our proposed replication mechanism in medianode, we prove its principle feasibility and identify further research issues such as how to combine the concept of quality of service (QoS) with replication mechanisms.

The structure of the paper is as follows. In Section 2, we present our replication system model. After giving a short overview about medianode architecture, we define the scope of our replication mechanism in medianode and present the characteristics of presentational media types, for which we identify a need for new replica units and granularities. Section 3 presents the design and implementation implementation architecture of our replication model. We describe the proposed replication maintenance mechanism, e.g. how and when replicas are created and how the updates are signalled and transported. In Section 4, we give an overview of related work. The merits and limitations of existing replication mechanisms are discussed and a comparison of our approach with previous work is given. We conclude the paper with a summary of our work and an outlook towards possible future extensions of our replication mechanism.

2. Replication System Model

2.1. Architectural Overview of medianode

The medianode system architecture[1] is intended for de-centralized operation of a widely distributed system. Within this distributed system, each participating host is called a medianode and conceptually equal to all other par-

ticipating nodes, i.e. a medianode is not considered a client or a server. Client or server tasks are taken on by medianodes in the system depending on the their resources and software modules.

The central element of a medianode is called its core. The core performs two primary tasks: (a) it dynamically loads code which implements the medianode's operations and instantiates objects; (b) the core implements the routing of requests between medianode's components (called bows) that are instantiated in a medianode.

Each dynamically loaded module implements a child class of medianode's root class, the bow class. Some bows implement basic operations that are necessary for the start of a medianode; these are not loaded dynamically but statically linked to the medianode binary and well known to the core. The bow class has three abstract subclasses which structure the operations of medianode in general. These subclasses are called Access Bow, Storage Bow and Verifier Bow.

Objects of the class Access Bow implement the visible activity of a medianode: e.g. an HTTP access bow implements means of requesting content from the medianode via the HTTP protocol, a Telnet access bow allows a user to connect to a medianode using the telnet application for basic information and management tasks. Storage Bows implement the functionality of distributed file systems and distributed databases. In medianode, such storage bows are always capable of operating in disconnected operation modes, i.e. they implement all functionality locally, keep all relevant data locally, and are able to react to requests to unreachable data. Verifier Bows are intended to check the availability and accessibility of data and services that have been requested by access bows or storage bows.

2.2. Scope of our Replication System

By analysing the service requirements distributed multimedia systems for the example of medianode, we identified a number of issues that the design of our replication system need to address:

- **High availability:** The replication system in medianode should enable data/service access in both connected and disconnected operation modes. Users can keep multiple copies of their files on different medianodes that are distributed geographically accross several universities in the state of Hessen.
- **Consistency:** Concurrent updates and system failures can lead to replicas not being consistent any more, i.e. stale state. The replication system should offer mechanisms for both resolving conflicts and keeping consistency between multiple replicas and their updates.
- **Location and access transparency:** Users do not need to know where presentation resources are physically located and how these resources are accessed.
- **Cost efficient update transport:** Due to the limitation of system and network resources, the replication system should use multicast-based transport mechanism for exchanging updates to reduce resource utilization.
- **QoS support:** The specific characteristics of presentational data, especially of multimedia data should be supported by the proposed replication mechanism.

In medianode, we mainly focus on the replication service for accessing data in terms of 'inter-medianode', i.e. between medianodes, by providing replica maintenance in each medianode. Consequently, a replication manager can be implemented as one or a set of medianode's bow instances in each medianode. The replication managers communicate among each other to exchange update information through the whole medianodes. A replication service within a medianode, i.e., 'intra-medianode', is not considered for the first stage of our implementation. However, the replication concept in this paper is straightforwardly applicable to the replication service for intra-medianode scope.

2.3. Concept of Logically Centralized Database

For a technical realization of our proposed replication system, we use the concept of a so-called "logically centralized database (LCDB)" which especially enables the transparent access to presentation materials. Similar to the concept of location-independent identifiers in distributed database system[3], LCDB enables a mapping between logical and physical resources. So users do not need to know where presentation resources are located physically and how they are accessed. Requests from users, either for reading or writing any presentation materials, are first sent to the Access Bow of the local medianode that runs on the user's local machine. After successful check of the accessibility of the user and the availability of the requested resources, the corresponding storage bows send the target data to the users. Figure 1 illustrates the interface point, the bows building the LCDB and the interactions between the bows. Some additional remarks on LCDB are in order:

- According to the data types, all of the presentation contents and their meta-data are stored in corresponding storage bows.
- The 'front-end' of the storage bow API provides unique interface functions, independent of the data types: this is similar to the VFS (virtual file system) interface in UNIX systems.

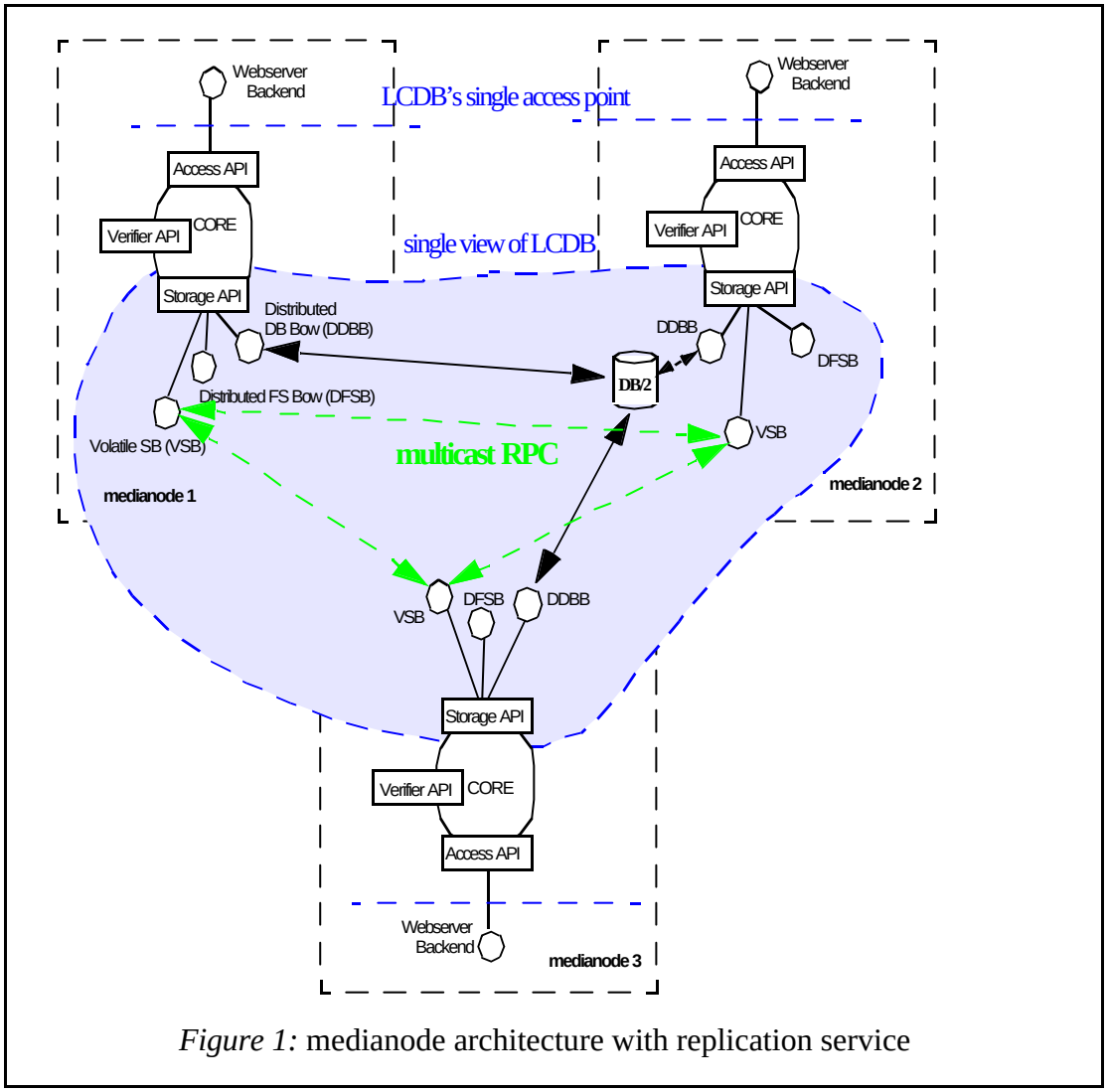

Figure 1: medianode architecture with replication service

- Replication has to be supported for most storage bows, although the number of replicas and the update frequency may differ between the individual bows.
- For the update propagation between replication managers, a multicast RPC (remote procedure call) communication mechanism is used.

2.4. Different Types of Presentation Data

Data organization comprises the storage of content data as well as meta information about this content data in a structured way. The typical data types which can be identified in medianode are the following:

- Presentation contents: this type of data comprises text, image, audio/video files and can be stored in file systems which should handle automatic data distribution and access, and also support the multimedia characteristics of this content type.
- Presentation description data, e.g. XML files.
- Meta-data of user, system, domain, and organization information. User's title, group, system platform, and university are examples for this meta-data category.
- Meta-data of system resource usage information such as memory usage, number of threads running within medianode process, number of loaded bows.
- Meta-data of user session and token information.

Table 1 shows an overview of these data types with their characteristics.

2.5. Classification of Target Replicas

As argued in subsection 2.2, the main goal of replication is to increase the high availability of medianode's services and to decrease the response time for accesses to data located on other medianodes. To meet this goal, data which

TABLE 1 Data categories and their characteristics in medianode

target data	availability requirement	consistency requirement	persistency	update frequency	data size	QoS playback	global interest
presentation description	high	middle	yes	low	small (middle)	not required	yes
organizational data	high	high	yes	low	small	not required	yes
file/data description	high	middle	yes	middle	small	not required	yes
multimedia resources	high	middle	yes	middle	large	required	yes
system resources	middle (low)	middle	no	high	small	not required	not strong
user session/ token	high	high	no	high	small	not required	no

is characterized by a high availability requirement (see Table 1) should be replicated among the running medianodes. We classify different types of target replicas according to their granularity (data size), requirement of QoS support, update frequency and whether their data type is 'persistent' or not ('volatile'). Indeed, there are three classes of replicas in medianode:

- Metareplicas (replicated metadata objects) that are persistent and of small size. An example would be a list medianodes (sites) which currently contain an up-to-date copy of a certain file. This list itself is replicated to increase its availability and improve performance. A metareplica is a replica of this list.
- Softreplicas which are non-persistent and of small size. This kind of replicas can be used for reducing the number of messages exchanged between the local and remote medianodes, and thereby reducing the total service response time. I.e., if a local medianode knows about the available local system resources, then the local replication manager can copy the desired data into the local storage bow, and the service that is requested from users which requires exactly the data can be processed in a shorter response time. Information about the available system resource, user session and the validity of user tokens are replicas of this type.
- Truereplicas which are persistent and of large size. Content files of any media type, which also may be parts of presentation files are Truereplicas. Truereplicas are the only replica type from the three types, to which the end users have access for direct manipulation (updating). On the other side, these are also the only replica type which requires the support of really high availability and QoS provision.

All replicas which are created and maintained by our replication system are an identical copy of original media. Replicas with errors (non-identical copy) are not allowed to be created. Furthermore, we do not support any replication service for function calls, and elementary data types.

3. Design and Implementation Architecture

3.1. The Replication Mechanism

Basically, our replication system does not assume a client-server replication model, because there are no fixed clients and servers in the medianode architecture; every medianode may be client or server depending on its current operations. Peer-to-peer model with the following features is used for our replication system:

(a) Every replica manager keeps track of a local file table including replica information.

(b) Information whether and how many replicas are created is contained in the every file table. I.e. each local replica manager keeps track of which remote replica managers (medianode) are caching which replicas.

(c) Any access to the local replica for reading is allowed, and guaranteed that the local cached replica is valid until notified otherwise.

(d) If any update happens, the corresponding replica manager sends a multicast-based update signal to the replica

managers which have the replica of the updated replica and therefore members of the multicast group.

(e) To prevent excessive usage of multicast addresses, the multicast IP addresses through which the replica managers communicate can be organized in small replica sub-groups. Examples for such sub-groups are file directories or a set of presentations about a same lecture topic.

3.2. Update Distribution & Transport Mechanism

The update distribution mechanisms in medianode differs between the three replica types and their managers. This is due to the fact that the three replica types have different levels of requirements on and characteristics of high availability, update frequency and consistency. Experience from [4] and [5] also shows that differentiating update distribution strategies makes sense for web and other distributed documents.

The medianode's replication system offers unique interface to the individual update signalling and transport protocols which are selectively and dynamically loaded and unloaded from the replica transport manager that is implemented as an instance of medianode's access bow. The possible update transport and signalling protocols are:

- RPC protocol [2] as a simple update distribution protocol. This mechanism is mainly used at the first step of our simple and fast implementation.
- A multicast based RPC communication mechanism. In this case, the updates are propagated via multicast other replica managers which are members of the multicast group. RPC2 [6,9] is used for the first implementation. RPC2 offers the transmission of large files, such as the updated AV content files or diff-files, by using the Side Effect Descriptor. But, the RPC2 with Side Effect Descriptor does not guarantee any reliable transport of updates.
- LC-RTP based reliable multicast protocol[10]: It is originally developed as an extension of RTP protocol to support the reliable video streaming within the medianode project. We adopt LC-RTP and check the usability of the protocol, depending on the degree of reliability required for the individual groups of replicas.

3.3. Approaches for Resolving Update Conflicts

The possible conflicts that could appear during the shared use of presentational data and files are either (a) update conflict when two or more replicas of an existing file are concurrently updated, (b) naming conflict when two (or more) different files are given concurrently the same name, and (c) update/delete conflict that occur when one replica of a file is updated while another is deleted. In most existing replication systems, the conflict resolving problem for update conflicts was treated as a minor problem. It was argued that most files do not get any conflicting updates, with the reason that only one person tends to update them[8]. Depending on the used replication model and policy, there are different approaches to resolving update conflicts, of which our replication system uses the following strategies [2,6, 11, 13, 15]:

- Swapping - to exchange the local peer's update with other peer's updates;
- Dominating - to ignore the updates of other peers and to keep the local tentative update as a final update;
- Merging - to integrate two or more updates and build one new update table;

3.4. Implementation Status

We have implemented a prototype of the proposed replication system model for Linux platform (Suse 7.0, Redhat 6.2). Implemented are the media (file) and its replica manager, update transport manager, replica service APIs which are Unix-like file operation functions such as open, create, read, write, close, and a Volatile storage bow which maintains user's session and token information. [15] gives a technically detailed description of our implementation.

4. Related Works

Several approaches to replication have already been proposed. The apporaches differ for distributed file systems than those for Internet-based distributed web servers and those for transaction-based distributed DBMS. Well-known replication systems in distributed file systems are Coda[6] and Roam[11] which keep the file service semantics of Unix. Therefore, they make easy to develop applications based on them. They are based on either client-server model or peer-to-peer model and use often optimistic replication which can hide the effects of network latencies. Their replication unit are mostly file system volumn which lead to a large size and relatively a low number of replicas.

There are some optimization works for these examples in terms of update protocol and replica unit. To keep the delay small and therefore maintain the sense of real-time interaction, it was desirable to use the unreliable transport protocol such as UDP. In the earlier phases, many approaches have used the unicast-based data exchanges by which the replication managers communicated with each other via 'one-to-one'. This has caused large delays and made the real-time interaction impossible. To overcome this problem, the multicast-based communication is used in some recent cases [8,9,12]. In the case Coda, the RPC2 protocol is used for multicast-based update exchange, which offers with Side Effect Descriptor the transmission of large files by using the Side Effect Descriptor.

For limiting the amount of storage used by a particular replica, Rumor and Roam developed the selective replication scheme[13]. A particular user who only needs a few of the files in the volume, the user can control which files to store in his local replica with selective replicaiton. A limitation or disadvantage of selective replication is the 'full backstoring' mechanism: if a particular replica stores a particular file in a volume, all directories in the path of that file in the replicated volume must also be stored.

JetFile[8] is a prototyped distributed file system which uses multicast communication and optimistic strategies for synchronization and distribution. The main merit of JetFile is its multicast-based callback mechanism by which the components of JetFile, such as file manager and versioning manager interact to exchange update information. However, the multicast callbacks in JetFile do not guarantee that they actually reach all of other replication peers, and the centralized versioning server which is responsible for serialization of all updates can lead to a overloaded system state. Furthermore, none of the existing replication systems does not support of the quality of service (QoS) characteristics of (file) data which they handle and replicate.

5. Summary and Future Work

In this paper, we presented a replication mechanism for distributed multimedia system medianode, and described the design and implementation architecture of the prototyped replication system. We first studied the characteristics of presentational media types which are handled in medianode, and extracted new replica units and granularities which have not been considered and not supported in existing replication mecanisms. We then built a replication mechanism for distributed multimedia systems based on the new requirements and the result of feature surveys.

We are currently in the process of implementing the versioning and storage/transport load leveling mechanisms, which are integrated with the replication manager. With the forthcoming implementation we will be able to build medianode as a highly available, scalable and cooperative, distributed media server for multimedia-enhanced teaching. The next working steps are to design other replication services which provide service implementations such as:

- Predictive replication: To increase access availability and to reduce latency. Similar approaches are Hoarding[14,16] and prefetched caching.
- QoS-aware replication for distributed multimedia systems, in which the decision whether a replica should be created from original file is made by checking the current usages of available system resources. An approach of combining replication, versioning and alternative media support is an good example for this replication model[17].

References

[1] The medianode project. (http://www.httc.de/medianode).

[2] G. Coulouris, J. Dollimore and T. Kindberg. *Distributed Systems*, 3rd Ed., Addison-Wesley, 2001.

[3] A. Eickler, A. Kemper and D. Kossman. Finding Data in the Neighborhood. In *Proc. of the 23rd VLDB Conference*, Athens, Greece, 1997.

[4] P. Triantafillou and D.J. Taylor. Multiclass Replicated Data Management: Exploiting Replication to Improve Effciency. In *IEEE Trans. on Parallel and Distributed Systems*, pages 121-138, Vol.5, No.2, Feb.1994.

[5] G. Pierre, I. Kuz, M. van Steen and A.S. Tanenbaum. Differentiated Strategies for Replicating Web documents, In *Proc. of 5th International Workshop on Web Caching and Content Delivery*, Lisbon, May 2000.

[6] M. Satyanarayanan, J.J. Kistler, P. Kumar, M.E. Okasaki, E.H. Siegel, and D.C. Steer. Coda: A Highly Available File System for a Distributed Workstation Environment. In *IEEE Transaction on Computers*, 39(4), April 1990.

[7] J. Yin, L. Alvisi, M. Dahlin and C. Lin. Volume Leases for Consistency in Large-Scale Systems. In *IEEE Transaction on Knowledge and Data Engineering*, 11(4), July1999.

[8] B. Groenvall, A. Westerlund and S. Pink. The Design of a Multicast-based Distributed File System. In *Proceedings of Third Symposium on Operating Systems Design and Implementation, (OSDI'99), New Orleans, Louisiana,* pages 251-264. February, 1999.

[9] M. Satyanarayanan and E.H. Siegel. Parallel Communication in a Large Distributed Environment. In *IEEE Trans. on Computers*, pages 328-348, Vol.39, No.3, March 1990.

[10] M. Zink, A. Jones, C. Girwodz and R. Steinmetz. LC-RTP (Loss Collection RTP): Reliability for Video Caching in the Internet. In *Proceedings of ICPADS'00: Workshop*, pages 281-286. IEEE, July 2000.

[11] D. Ratner, P. Reiher, and G. Popek. Roam: A Scalable Replication System for Mobile Computing. In *Workshop on Mobile Databases and Distributed Systems (MDDS)*, September 1999. (web site http://lever.cs.ucla.edu/project-members/reiher/available_papers.html)

[12] M. Mauve and V. Hilt. An Application Developer's Perspective on Reliable Multicast for Distributed Interactive Media. In *Computer Communication Review*, pages 28-38, 30(3), July 2000.

[13] D.H. Ratner. Selective Replication: Fine grain control of replicated files. *Master's thesis, UCLA*, USA, 1995.

[14] G.H Kuenning. Seer: Predictive File Hoarding for Disconnected Mobile Operation. *PhD. dissertation, UCLA-CSD-970015. UCLA*, USA, 1997.

[15] G. On and M. Liepert. Replication in medianode. *Technical Report TR-2000-03*, Darmstadt University of Technology, Germany, September 2000.

[16] C. Griwodz. Wide-Area True Video-on-Demand by a Decentralized Cache-based Distribution Infrastructure. *PhD. dissertation*, Darmstadt University of Technology, Germany, April 2000.

[17] J. Chung-I and M.A. Sirbu. *Distributed Network Storage with Quality-of-Service Guarantees*. web site http://www.ini.cmu.edu/~sirbu/pubs/99251/chuang.htm