Peer-to-Peer Architectures for Scalable, Efficient and Reliable Media Services

Vana Kalogeraki
Univ. of California, Riverside
Riverside, CA 92521
vana@cs.ucr.edu

Alex Delis
The Univ. of Athens
Athens, 15771, Greece
ad@di.uoa.gr

Dimitrios Gunopulos
Univ. of California, Riverside
Riverside, CA 92521
dg@cs.ucr.edu

Abstract

In this paper, we propose and study the behavior of a number of peer-to-peer (P2P)-based distributed computing systems in order to offer efficient and reliable media services over a large-scale heterogeneous network of computing nodes. Our proposed middleware architectures exploit features including availability of high-performance links to networks, usage of exclusive and partial indexing in peers, making nodes "aware" of the content of their own vicinity, replication of objects and caching of popular items, as well as full connectivity among servers if feasible. Through detailed simulation and experimentation, we investigate the behavior of the suggested P2P architectures for video provision and examine the trade-offs involved. We show that under realistic assumptions, the proposed architectures are resilient to multiple peer-failures, provide timeliness guarantees and are scalable with respect to dropped requests when the number of messages in the network increases.

1. Introduction

The ever improving network infrastructure in combination with the emerging peer-to-peer (P2P) framework offer new opportunities for distributed organization of computing systems. Thus far, most of the work in the area has concentrated in the exchange/sharing of "small" objects including MP3 music files, images, and audio. It is our hypothesis that with certain restrictions on the key distribution nodes of an P2P infrastructure and the necessary provisions, we will be able to offer diversified and dependable video services on ad-hoc P2P networks. Prior work in furnishing video over a computer network has exclusively focused in the creation of video-on-demand (VOD) systems [3, 7, 24, 18, 31, 10]. Although there have been a number of proposals, research prototypes, and some VOD products, it is evident that initial investment required for commercial use is steep. Such systems are also restricted by the number of concurrent accesses that they allow as well as load balancing issues that ensue when the demand for video streams is skewed [18, 24, 9].

In this paper, we build upon the approach of ad-hoc P2P networks of resources and propose new architectures that can efficiently support video-related services. The range of such services is wide and includes storage and management of movies, video-clips, and documentaries. In the context of a P2P infrastructure realization, two issues need to be considered:

- One should provide fast connections to the end user. However, this is well within reach as more individuals choose T1-level (or higher) connections both for their businesses and homes. In addition, cable and other specialized modems (ADSL, HDSL, etc.) do provide for asymmetric connections with impressive downstream rates (around 4-10Mb/sec) while maintaining significant upstream capabilities (close to 0.5Mb/sec). The corresponding figures for network transmission for a two hour MPEG-II movie are 8Mb/sec network bandwidth and 7.2 Gigabytes space per movie.

- The size of a reasonable population of movies can definitely increase the disk space requirements into the Petabyte area. The accommodation of such volumes calls for collaborative computing that can be carried out only in a distributed setting.

1.1. Assumptions and Problem Statement

We assume that movies and/or video-clips are maintained by a network of computing sites. The latter are termed "servers" and they are the computing nodes responsible for storage as well as retrieval of the multimedia elements in discussion. Via the existing networking infrastructure, servers stream requested clips and movies to user-sites for viewing and/or processing. We assume that all sites are

0-7695-1926-1/03/$17.00 (C) 2003 IEEE
IEEE COMPUTER SOCIETY

connected via a multi-hop network. However, the key provision is that (some of the) peers may be connected via a low-latency and high-bandwidth networking option capable of effective shipment and handling of high data volumes; for instance, the network could function even at the OC3 level [26]. Customers interact with the infrastructure via (thin) clients that allow for the search and display of the clips obtained from the network. Fig. 1 depicts the

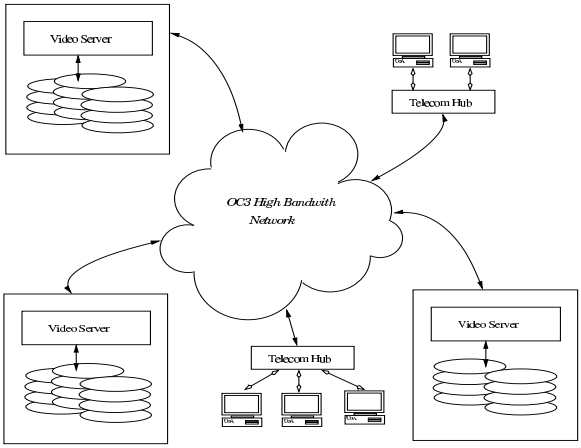

Figure 1. Environment for P2P Video Services

typical peer-to-peer computing environment in which we envision the placement of such P2P video-services. The video-segments are of considerable volume –at least 0.5 Gigabytes– and are organized by storage managers. The latter operate on the top of multiple disks resident within the servers' chassis. Segments and movies are all entitled, feature a number of keywords pertinent to their content, date of creation, names of producers, owner, distributor, and cast, as well as a summary of their contents and terms and conditions for the video's usage.

In the above operating environment and while observing quality of service (QoS) requirements for the delivery of multi-media data, a number of issues have to be addressed:

1. Architectural organization of video servers and distributed indexing mechanisms that allow for efficient retrieval of multimedia data.

2. Routing of queries in the network so that "flooding" of messages is avoided and compliance with QoS requirements for the delivery of data.

3. Assuring reliability of a P2P network for video services in light of server failures. Would replication of multimedia objects be allowed, guarantees for the response time of client requests should be established. In addition, ensuring that recovery from multiple site failure is considered critical.

4. Dealing with the dynamic aspects of the systems such as arrival/departure of a server node, load-balancing in light of skewed accesses, publishing/withdrawal of video-segments by users/servers and on-line recreation of indexes.

The main contributions of this work are that we:

- propose scalable, dynamic and reliable P2P architectures for delivering video services in ad-hoc environments.

- suggest efficient mechanisms for object replication and load distribution in the P2P architectures. These mechanisms are instrumental for ensuring reliable operation and improving the availability and performance of the services.

- provide an extensive experimental study that establish the performance, scalability, reliability, and flexibility of the investigated P2P architectures for video-services.

1.2. Previous Work

A number of mechanisms have been proposed recently for providing large-scale storage ([21, 29, 11]), efficient search and retrieval ([30, 19, 2]) and organizing the peer-to-peer networks ([27]). The Jxta and Hailstorm initiatives intend to offer P2P architectures that follow the fully distributed and server-based approach respectively [23, 33, 12]. A classification of the users who make use of the free P2P services is provided in [5, 4]. In [28], a mapping of the Gnutella network is discussed and statistics about the path-lengths traversed by user requests are experimentally derived.

Khazana [8] uses shared "regions" of memory space among a number of LAN-based sites in a seamless manner in order to assist the delivery of multimedia. In [35], video staging is used as a tool to retrieve only parts of a video-stream from a server located across a wide area backbone network. The design of a tightly connected distributed system for the provision of Video-on-Demand and its evaluation is discussed in [15]. Techniques that improve the throughput of long-term storage subsystems for multimedia with the use of staggered striping, disk scheduling, and multi-user editing servers are presented in [7, 6, 14, 20]. In [25], the design of a fault-tolerant VOD system that is able to overcome the failure of disk failure is described. The use of segmentized multimedia objects in proxy servers in advocated in [13] to guarantee quality of service requirement for video distribution. A middleware layer that achieves performance differentiation and soft QoS guarantees is proposed in [1]. The use of forward error correcting codes in streaming data is used as the basis for Digital Fountain's solution

IEEE COMPUTER SOCIETY

to media delivery to multiple clients [17]. The functionality of BeeHive, a novel resource manager that handles transactions with quality of service requirements over the Internet, is discussed in [32].

In [22], the design of system modules for multi-resolution media delivery to clients with a variety of network connections is outlined. Possibly the work more closely related to ours is [34]; however, the latter mostly deals with file sharing options in the design of P2P applications. In contrast, our work is the first effort to the best of our knowledge that designates architectural choices for the development of P2P video-services with guarantees for reliability and outlines the dynamic behavior and reconfiguration of the peers as needed.

2. Architectures for P2P Video-Services

In this section, we present alternative configurations that can help create the required underpinning infrastructure for the realization of reliable video-services.

2.1. Single/Multiple Index site(s)–Multiple Servers (SIMS/MIMS)

In this architecture, a number of servers form the basis for managing the storage and retrieval of video objects. Movies and clips resident in these nodes are allowed to be viewed by users who check into the system. These objects can be shipped across the network to requesting users and other sites and be shared among servers should the latter be necessary. Peers have a maximum capacity for video objects and each stored object maintains frequency accesses and timestamps for the last time they were accessed.

The central feature of the architecture, depicted in Fig. 2, is that a node undertakes the exclusive role of being the *indexing* site. The indexing node is responsible for (1) brokering of connections between the users and the data servers and (2) finding and efficiently alerting users about the location of sought video-objects. This is done through a set of "global" indexes maintained for the stored objects in the network. Each stored object is represented through a triplet that includes the object-ID (associated with search terms in the *Multimedia Indexing* module), the IP address (in which the object is resident) and a port number (through which the object can be fetched).

We assume that a requesting node maintains an open connection with the its peer-server throughout the entire time of the interaction regarding a video object. If the indexing node indicates "local" retrieval from the peer-server, the objects can be `downloaded` immediately. Otherwise, the object will have to be either routed to the peer (from the node that manages a copy of the object) or the client has to

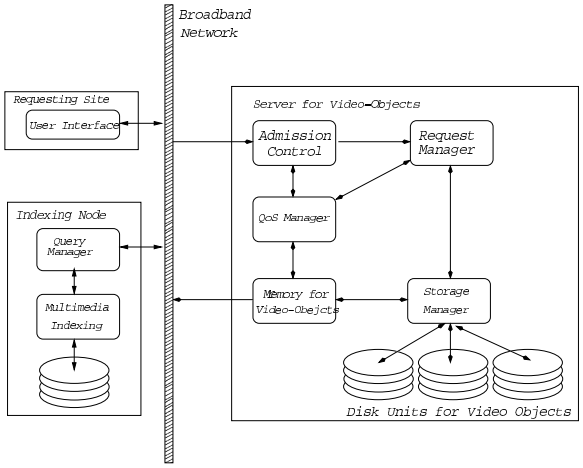

Figure 2. SIMS/MIMS Architecture

establish a new connection with the peer bearing the object. These two options can be summarized as follows:

1. *First Cache–Then Deliver:* The item is first cached to the server that the requesting site is attached to and then it is distributed. The condition that enables such a copy is that an object has become popular. The latter is quantified by constraints that indicate that a object has received $k\%$ of the most recent Δ requests. The *Indexing Node* has to also be alerted to this effect. Users download objects directly from their corresponding serving peers.

2. *Forward Object*: The data server managing the item streams the object via the network directly to the requesting site. In this case, users download copies from peers that manage object copies.

If more than two servers can furnish the sought data objects, simple heuristic policies can be applied depending on the least loaded peer, proximity of the user to peer (as this is manifested by the number of hops needed in the network), as well as on-going traffic at the segments of the network. This information could be easily provided to the *Indexing* site with piggybacked messages.

To ensure reliable operation of the network in light of node failure and/or time outs, we propose a simple yet effective replication policy that calls for the mirroring of an object in the network to at least another additional node. The selection of the new node can be done randomly or based on load-balancing heuristics. In conjunction with the possible caching and subsequently floating copies (due to First Cache–Then Deliver operation above), the P2Psystem will be able to recover from more than two site failure. As peers reach their capacity, there is obviously need for housekeeping by doing garbage collection. Objects can be elim-

IEEE COMPUTER SOCIETY

inated in a least recently requested object fashion and provided that at least two copies exist in the network. The latter can be determined by issuing a query message to the indexing node and receiving more than two object hooks.

2.2. Multiple Independent Indexed Servers (MIIS)

In this architecture, peers maintain their own multimedia data. In addition, each peer features *local* access structures that help identify and retrieve objects of interest. Servers are also aware of the contents of all their (sibling) peer-servers in the P2Pnetwork. These are peers that are a few (usually 2 or 3) hops away in the network. To this end, peers maintain *partial* indexes for the video and/or clips held by sibling nodes. The rationale behind such partial indexes is to offer distributed "hooks" in the computing vicinity. Fig. 3 depicts this architecture term Multiple Independent Indexing Servers (MIIS). Servers, as in the SIMS/MIMS configurations, maintain open connections with all their server peers as the assumption is that their networking substrate displays low-latency and high bandwidth characteristics.

Periodically, servers propagate updates to their server peers about newly arrived movies/clips so that a consistent global view is achieved in the network. A low-entropy protocol (that may function after a sufficient inactive period between any two sites has elapsed) can be used to propagate appropriate changes of meta-data to other peer nodes in the P2Pinfrastructure.

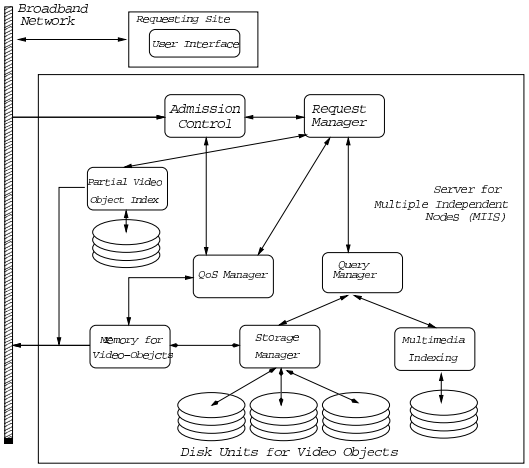

Figure 3. MIIS Architecture

A user can connect to either its closest/local server or a node of her choice (by issuing `ping` and receiving `pong` messages). By contacting a peer-server, users initiate object requests by using the payload of `query` messages. When a server receives a request, it searches through its local repository. If the object is found locally, it is presented to the requesting site (through a `query_reply` message). Otherwise, the peer uses its partial index access objects stored in a sibling (peer):

- If the object is found to be available in a sibling, the handling server can satisfy the pending request either re-directing the user to the remote server (that has the object) (*forward* option), or by obtaining the movie/clip locally due to popularity (*cache first-then deliver* option).
- Otherwise, the handling server simply forwards the `query` message to all its peers for further processing. The latter will ultimately satisfy the request as it triggers the searching of all servers in the P2Pnetwork.

Whenever object caching takes place, the peer refreshes its local index accordingly and notifies a queue. This is done so that siblings in a forthcoming opportunity are notified about the change and change the contents of their own *local* indexes. We want to avoid hard consistency of data items among servers and consequently, we do not adhere to any such protocol. Clearly, not all objects are available in all sites but should the cache first-then deliver option be used, it is guaranteed that at least the most popular video/clips will be present (due to caching) in multiple locations offering reduced object access latency and increased reliability.

2.3. Fragmented And Multiple Servers (FAMS)

This scheme, follows the fundamental design choices of the Gnutella network. Subsequently, there is neither centralized data server not global indexing sites for the multimedia objects. Each peer essentially is a video-service provider and maintains a local set of movie and/or video clips files along with their meta-data. The key features of this architecture is that no upper limit in the number of servers exists and the servers are not fully connected. Fig. 4 depicts the organization of a server. It is worth noting that nodes could be connected even with low-bandwidth links and the topology of the resulting grid could be random. In addition, they are no explicit quality of service modules on the peers. By simply restricting their connections to clients to a handful or so, peers ensure timely delivery of multimedia objects.

Each node in the P2Pnetwork maintains a *Peer-List* which keeps track of the IP-addresses of its own peer sites. The latter make up the sites with which a node is/was connected in the (recent) past and thus, it is likely to connect again (or remain connected) in the near future. The number of connections the node maintains is typically limited by the peer's resources and the type of network links that exist between the peer and the rest of the network. It is worthwhile pointing out that nodes are connected in an ad-hoc manner and no peer has a global view of the system "as is" at any

IEEE COMPUTER SOCIETY

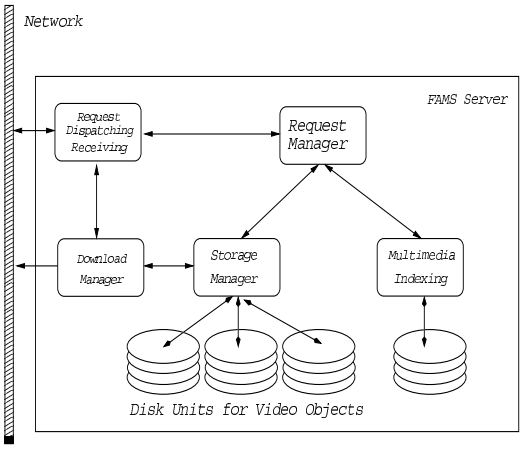

Figure 4. FAMS Architecture

particular moment [16]. This creates both problems and opportunities; the topology of the network is not known and can change dynamically over time. Every node only knows about its first-line peers. On the other hand, in light of site failures, the overall system can still function (almost certainly with longer response times).

In order to make our FAMS model more pragmatic, while providing video services, we impose the following two assumptions:

- We disallow `downloading` of multimedia objects through low bandwidth connections that may appear in *Peer-List*s if a faster connection is available. Although a connection to a peer may exist, it might be not viable in order to sustain the presumed quality of service requirements.
- We assume that any video segment is available from at least two peers in line with the reliability rule suggested in the SIMS/MIIS architectures. This policy permits "new releases" to have at least one replica randomly created in some node in the P2Pnetwork. The `downloading` option allows for further caching of objects and propagation of their corresponding meta-data.

To search for a object in the network, a node sends a `query` message to its peers including a "constraint" that essentially articulates the search operation. Typically this constraint is a set of keywords (meta-data) attempting to outline what is being sought. A peer receiving a `query` message evaluates the constraint locally against the meta-data in its own local storage manager. If the evaluation is successful, the peer will generate a `reply` message back to the originating node which includes the object(s) corresponding to the constraint. Otherwise, a new `query` is initiated from the peer in discussion to the nodes in its own *Peer List*. If the segment is available in the network, this recursive operation guarantees ultimate retrieval. However, in practice we limit the depth of the recursion to avoid flooding the network with messages. User sites can `download` the object(s) of their choice directly from the network peers and commence viewing whenever is deemed feasible.

As there is no specialized site for indexing and/or management of meta-data, deployment of new nodes in the network is straightforward. Also, when new items are publicized by a single node, the only task that needs to happen is that at least another copy is floated. This will at least guarantee successful operation in a single node failure.

3. Experimental Evaluation

In this section, we discuss our experimental evaluation of the proposed peer-to-peer architectures for video-service provision. By employing a number of different workloads, we have estimated performance indicators and carried out sensitivity analyses. In this context, our main goals were to:

- investigate the "average" behavior of the suggested configurations in the presence of uniform and skewed requests.
- examine the reliability features of each architecture and the effect of the proposed replication/caching policy as well as to experimentally gauge the levels for continuous operation despite failure of multiple nodes.
- carry a competitive scalability analysis and quantify the number of requests unable to be serviced by every architecture.

In order to carry out our experimental objectives, we developed detailed queuing network models for all architectures (discussed in the following subsection) and based on those models we created four extensive simulation packages. The software was developed in C++ and the size of the packages ranges from 2k-2.5k lines of source code; the packages run on the Linux RedHat7.1 distribution. The key parameters used across all simulation packages along with their values are outlined in Table 1.

We ran experiments with 10,000 requests for continuous objects. In all configurations, the requests "arrive" sequentially. Each time interval, a random user selects a movie and submits the request. We used two distributions to model the movie selection. In the first case, the distribution of the requests is uniform. In the second, one tenth of the multimedia objects are popular movies and represent half of the requests. The other half of the requests is uniformly distributed to the rest of the objects. The popular objects are randomly distributed to the servers. Also, we assumed that all the multimedia objects have the same size, and take the same time to download. We varied this download duration of the multimedia objects between 100 and 1,000 time units in different experiments. Each movie/clip download, keeps

IEEE COMPUTER SOCIETY

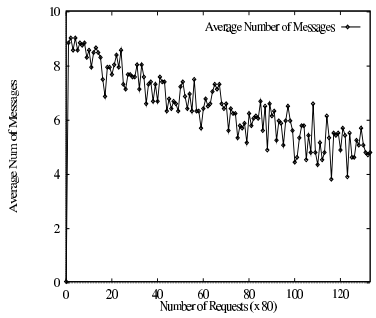

Figure 5. MIIS Architecture: Average number of messages per request to find an object.

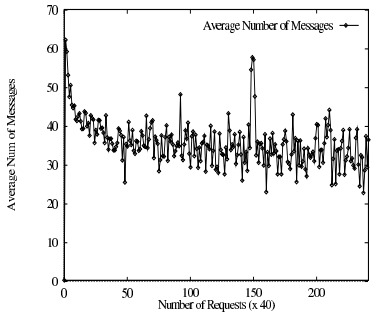

Figure 6. FAMS Architecture: Average number of messages per user request to find an object.

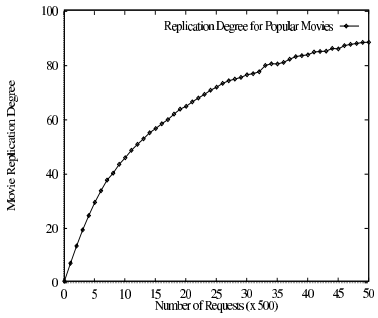

Figure 7. MIIS Architecture: Average Replication Degree for Popular Movies.

Parameter	SIMS/MIMS	MIIS	FAMS
NumPeers	10	10	100
NumObjects	1000	1000	1000
ConnServer	10-20	10-20	2-10
FracObjects	200	200	20
RepliDegree	2	2	2 (init)
NoIndSits	1	N/A	N/A
VicinityObjs	N/A	finite num	finite num
NetworkType	Star	fully connected	random
RandNetDgr	N/A	10	10
RespTime	100	100	100

Table 1. Key Simulation Parameters

one server connection busy. In all architectures, we assume that there is an upper limit in the number of users that can be simultaneously served by a single site. This number is in the range of 10 to 20 connections for the SIMS/MIMS and MIIS architectures and smaller (in the range of 2 to 10 connections) in the FAMS architecture.

4. Description of the Experiments

4.1. Search Performance

In the first set of experiments, we evaluated the efficiency of each of the architectures. This was done by measuring the average number of messages that have to be exchanged between users and servers for each request before downloading commences. Here, we do not consider the time it takes for a server to search its index. For datasets in the range of $10^4 - 10^6$ movies/clips, we expect the logarithmic search time to be reasonably short compared to the communication time among servers.

In the SIMS/MIMS architecture, the number of messages remains small. Upon login, a user sends a `ping` message to connect to one of the well-known servers in the network. The server accepts the connection by replying with a `pong` message. If the movie/clip is available, the download process can begin. Otherwise, the user queries the *indexing* node for movies/clips in the network. Therefore, for each user request, only `two` or `four` messages are needed to find a movie. If the user request is denied by the *Admission Control* manager and the user re-issues his request, the latter is viewed as an entirely new request.

Figure 5 shows that the average number of messages per user request in the MIIS architecture. To search for a movie, the user sends a `query` request to its local server. If the peer has the movie, or has its location in the local index, then it replies with a `query_reply` message that contains the location of the movie for download. On the other hand, if the server has no information about the location of the movie, it has to initiate a search in the network. Since we assume that the between server connections are always on, the server can broadcast the `query` to all other servers. The figure also shows that the average number of messages decreases with the number of user requests over time. As more users request movies from the server, the server learns about the location of the movies and therefore re-directs the users to those servers directly. This reduces the number of the messages in the network.

The average number of messages per user request is much higher in the FAMS architecture (Figure 6) compared to the other architectures. The reason is that in this architecture there are no dedicated servers where the users find movies/clips. Also, since there is a larger number of servers in the network, each peer has only a partial index of the movies of its "vicinity". When the server has zero-knowledge of the location of a multimedia object, it initiates

IEEE COMPUTER SOCIETY

a Breadth-First-Search in the network. As a result, the number of messages propagated in the network is large. Similar to the MIIS architecture, the performance of the FAMS architecture improves as the server "learns" about the location of the movies in the network. However, the improvement is small because the server learns about only a small proportion of the movies that are available in its peers (compared to the MIIS architecture, and for the same replication degree of the movies).

4.2. Replication Algorithm

In the second set of experiments, we evaluated the performance of our replication algorithm. Our goal was to investigate the replication degree of the popular and the non-popular movies as the number of requests increases. We ran experiments in the context of the MIIS architecture only. The reason was, that, in the SIMS/MIMS architecture the algorithm cannot be directly applied as we assume that servers are not directly connected with each other. On the other hand, the connections in the FAMS architecture form a graph with relatively low degree, and servers are not connected to most of the other servers directly. To do the replication efficiently, we would have to open direct connections between servers, and change the topology of the network dynamically.

Figure 7 depicts the average Replication Degree of the popular movies in the SIMS/MIMS architecture as a function of the user requests. To verify our results, we chose to run experiments in a larger network with 100 dedicated movie servers, 10,000 movies/clips and 10,000 peers. At first, the replication degree for all movies is 2. If a server peer receives more than three requests for a movie not locally available, it caches the multimedia object. The latter is an indicator that the clip/movie in discussion is a popular one. As Figure 7 shows, the replication degree of the popular movies increases quickly. Eventually, most of the servers cache a copy of the popular movies. As a result, the number of messages needed to find a movie decreases continuously. Figure 8 shows the average Replication Degree for the non-popular movies. Our experimental results indicate that only a few peers cache locally a copy of the non-popular movies. The reason is that these movies are requested less frequently and a maximum of four replicas seems to be sufficient to satisfy the user requests.

4.3. Reliability

To evaluate the reliability of the different architectures, we measured the number of movies that are no longer available in the system as servers fail. Figure 9 displays the number of movies lost in the SIMS/MIMS architecture when 30% of the servers fail over various replication degrees for

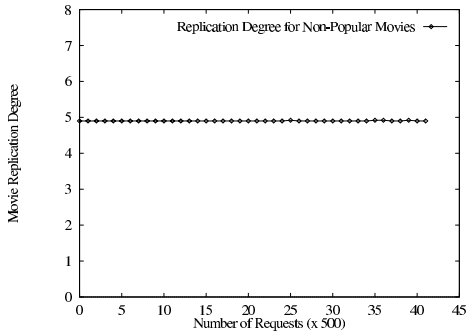

Figure 8. MIIS Architecture: Average Replication Degree for Non-Popular Movies.

the movies/clips. The number of objects no longer available increases in this case due to the higher probability that all object copies are located in the "faulty" peers. With replication degree 2 and 30% of the servers failing, 8% of the movies can be lost. Our experiments indicate that when additional servers maintain object copies, only a few movies are eventually lost.

In the MIIS architecture, the number of movies lost are initially the same compared to SIMS/MIMS for the same replication degree and same number of faulty servers, as shown in Figure 10. Here too the number of movies lost for 30% faulty Servers and replication degree 2 increases considerably compared to the number of movies lost when the replication degree is larger, or the number of faulty servers smaller. However, in the MIIS architecture, as the operation of the system progresses, more copies of the popular movies are cached into the servers, therefore the number of movies lost decreases with the user requests. The improvement is evident, as a comparison of Figures 9 and 10 shows, but it is slow because mainly only popular movies are replicated.

Finally, we run a set of experiments to quantify the number of objects lost in the FAMS architecture when Servers fail over different replication degrees. Figure 11 shows that only very few movies/clips are lost when 10% of the servers fail respectively. Since there is a large number of servers where the objects are uniformly distributed among, each server has a small number of objects, so the probability that ten servers with the same movie will fail simultaneously is small. Clearly, FAMS is the most reliable architecture regarding the number of server failures.

4.4. Scalability

In the fourth set of experiments, we evaluated the scalability of the architectures by measuring the number of user requests that every architectures rejects when the maximum

IEEE COMPUTER SOCIETY

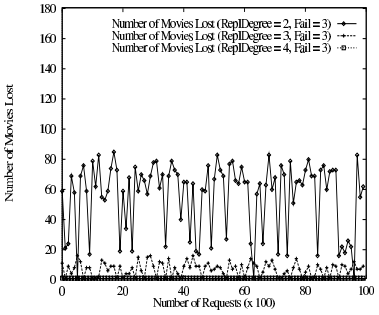

Figure 9. SIMS/MIMS Architecture: Number of movies lost when 30% of the Servers fail at movie replication degrees 2, 3, and 4.

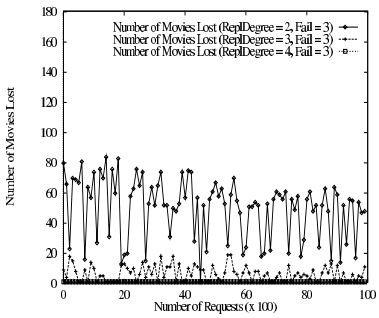

Figure 10. MIIS Architecture: Number of Objects lost when 30% of the Servers fail at initial replication degrees 2, 3 and 4.

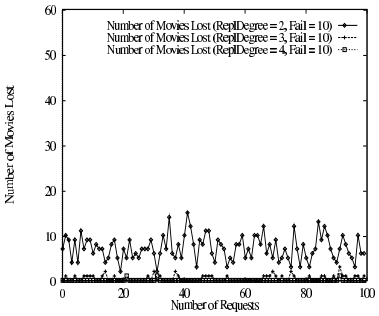

Figure 11. FAMS Architecture: Number of movies lost when 10% of the Servers fail at movie replication degrees 2, 3 and 4.

number of connections allowed is exceeded.

Figure 12 shows the number of rejected requests for the SIMS/MIMS architecture. The server can facilitate 10 to 20 concurrent open connections. Our results indicate that the architecture can service 87% of the requests (approximately 1,300 rejections) when the maximum number of connections per server is 10. Also, the results show that as the maximum number of connections for the server increases to 15 and more, almost no requests are rejected.

The MIIS architecture demonstrates better scalability performance compared to SIMS/MIMS (Figure 13). The architecture can service 88% of the requests (1,200 rejections) when the maximum number of open connections per server is 10, but 95% when this number raises to 12 and almost 100% for 20. The main reason for that is that it allows the popular movies to replicate, so a server that is busy uses its index to redirect a user request to another server that can service the request (if that server is not busy itself).

In Figure 14, we show the number of failed requests in FAMS. Since the servers in this architecture are likely to be less powerful, we assume 2, 5, 7 or 10 maximum number of simultaneously open connections per server. Allowing only 2 connections per server results to 2,000 dropped requests. However, increasing the limit (of network connections) quickly reduces the number of dropped requests, and in fact even allowing 5 connections per server results in 0.25% requests rejected.

5. Conclusions

In this paper, we have investigated a number of peer-to-peer architectures that offer scalable, reliable and timeliness support to media services over a network of computing nodes. These architectures are:

- Single/Multiple Index site(s)–Multiple Servers (SIMS/MIMS): peers function independently but the indexing of network objects happens either via accessing a single node (SIMS) or multiple dedicated for this purpose nodes (MIMS).
- Multiple Independent Indexed Servers (MIIS): peers feature managers that locally store movies/objects as well as *partial* pointers/hooks for the objects available in their network vicinity.
- Fragmented and Multiple Servers (FAMS): fully distributed and independent peers function as data storing servers that connect in an ad-hoc manner to form a random network.

We have outlined the operational protocols for the above frameworks follow and qualitatively compared them. We evaluate the various trade-offs that such systems present via extensive simulation experiments and under a number of diverse settings. We have sought to quantify issues related to search performance for multimedia objects, the behavior of the replication algorithm utilized, reliability, scalability and timeliness. In particular, we have evaluated:

1. the efficiency with which each of the architectures can reply to a given user request: for each architecture we count how many messages on average have to be exchanged between servers before the movie download can begin. It is evident that the SIMS/MIMS architecture needs the smallest number of messages to start downloading. Our experimental results indicate that on average the MIIS architecture achieves very similar performance. The reason is that as the number of

IEEE COMPUTER SOCIETY

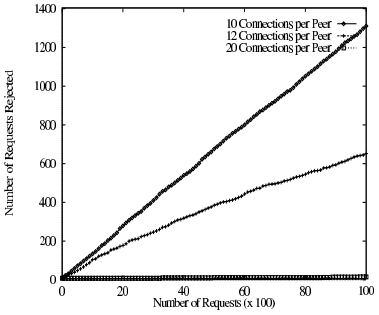

Figure 12. SIMS/MIMS Architecture: Cumulative Number of Requests Rejected for Maximum Number of Open Connections 10, 12 and 20.

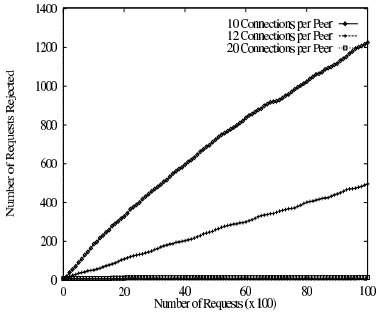

Figure 13. MIIS Architecture: Cumulative Number of Requests Rejected for Maximum Number of Open Connections 10, 12 and 20.

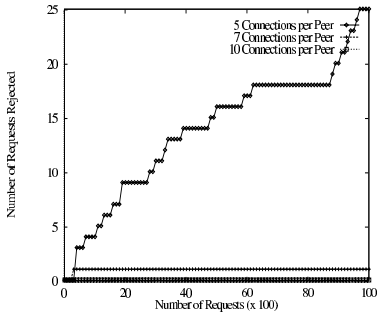

Figure 14. FAMS Architecture: Cumulative Number of Requests Rejected for Maximum Number of Open Connections 5, 7, and 10.

user requests increases, each server adds the locations of more movies in its local index, and it tries to download the most popular movies. With limited connections per server and no centralized indexing, the number of messages in the FAMS architecture can increase dramatically. The messages are propagated over many hops from one server to another until the multimedia objects are found.

2. the performance of the movie replication algorithm we proposed in the context of the MIIS architecture: we show how the replication degree of the popular and the non-popular movies changes as the number of requests increases. Our experiments show that the replication degree of the popular movies increases quickly. As a result, the cache hit rates for the popular movies continuously increase. In addition, the number of messages needed to find a movie/clip decreases continuously.

3. the reliability of the different architectures by measuring the number of movies no longer available in the system, as servers fail: our experimental results show that the FAMS architecture is the most reliable one because different server peers have a variety of movies. Also, our results show that as the system executes, the reliability of the MIIS architecture improves. This is due to the fact that more copies of popular objects are cached into servers, therefore the number of movies/clips lost decreases with the user requests.

4. the scalability of the architectures by measuring the number of user requests rejected by each of the architectures: in the SIMS/MIMS architecture, the indexing server can become a bottleneck as the number of user requests increases. However, the server peers do guarantee QoS once their requests are accepted. In the FAMS architecture the number of rejected requests is very small, but these may be slow connections, therefore it is difficult to guarantee QoS support.

It is worth mentioning that FAMS is likely the most inexpensive option as no particular computational features are required of the participating sites. It is our belief however that schemes similar to SIMS/MIMS and MIIS will benefit the area of multimedia delivery the most as the close collaboration of some dedicated storage peers helps in the timely and efficient delivery of multimedia data. In our experiments, all the distributed architectures (SIMS/MIMS, MIIS, and FAMS) have shown impressive reliability and scalability furnished by only a small degree of replication.

Acknowledgments: A. Delis was on a leave from Polytechnic University and was partially supported by NSF under grant IIS-9733642 and D. Gunopulos was supported by grants IIS-9984729 and ITR-0220148.

References

[1] T. Abdelzaher and K. Shin. QoS Provisioning with qContracts in Web and Multimedia Servers. In *Proceedings of the 20th IEEE Real-Time Systems Symposium*, Phoenix, Arizona, December 1999.

[2] K. Aberer, M. Punceva, M. Hauswirth, and R. Schmidt. Improving Data Access in P2P Systems. *IEEE Internet Computing*, 6(1):58–67, January/February 2002.

[3] S. Adali, K. Candan, S.-S. Chen, K. Erol, and V. Subrahmanian. Advanced Video Information Systems. *ACM Multimedia Systems Journal*, 4(4):172–186, 1996.

IEEE COMPUTER SOCIETY

[4] L. Adamic, R. Lukose, A. Puniyani, and B. Huberman. Search in Power-Law Networks. Technical report, Xerox Parc Research Center, http://www.parc.xerox.com/istl/groups/iea, Palo Alto, CA, 2000.

[5] E. Adar and B. Huberman. Free Riding on Gnutella. Technical report, Xerox Parc Research Center, http://www.parc.xerox.com/istl/groups/iea/papers/plsearch, Palo Alto, CA, 2000.

[6] W. Aref, I. Kamel, and S. Ghandeharizadeh. Disk Scheduling in Video Editing Systems. *IEEE Transactions on Knowledge and Data Engineering*, 13(6):933–950, 2001.

[7] S. Berson, S. Ghandeharizadeh, R. R. Muntz, and X. Ju. Staggered Striping in Multimedia Information Systems. In *Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data, Minneapolis, Minnesota, May 24-27, 1994*, pages 79–90. ACM Press, 1994.

[8] J. Carter, A. Ranganathan, and S. Susarla. Khazana: An Infrastructure for Building Distributed Services. In *Proceedings of the 18th IEEE International Conference on Distributed Computing Systems*, Amsterdam, The Netherlands, May 1998.

[9] S. Carter, J. Paris, S. Mohan, and D. Long. A Dynamic Heuristic Broadcasting Protocol for Video-On-Demand. In *Proceedings of the 21st IEEE International Conference on Distributed Computing Systems*, Phoenix, CA, May 2001.

[10] M. Chen, D. Kandlur, and P. Yu. Storage and Retrieval Methods to Support Fully Interactive Playout in a Disk-Array-Based Video Server. *ACM Multimedia Systems Journal*, 3(3):126–135, July 1995.

[11] G. Chockler, D. Dolev, R. Friedman, and R. Vitenberg. Implementing a Caching Service for Distributed CORBA Objects. In *Proceedings of the IFIP/ACM International Conference on Distributed System Platforms and Open Distributed Processing (Middleware 2000)*, Hudson River Valley, NY, April 2000.

[12] M. Corporation. Hailstorm Software Architecture. http://www.microsoft.com/net/hailstorm.asp.

[13] H. Fahmi, M. Latif, S. Sedigh-Ali, A. Gafoor, P. Liu, and L. Hsu. Proxy Servers for Scalable Interactive Video Support. *IEEE Computer*, 34(9):54–60, September 2001.

[14] S. Ghandeharizadeh and R. Muntz. Design and Implementation of Scalable Continuous Media Servers. *Parallel Computing, Special Issues on Applications, Parallel Data Servers and Applications*, 24(1):91–122, January 1998.

[15] L. Golubchik, R. Muntz, C.-F. Chou, and S. Berson. Design of Fault-Tolerant Large-Scale VOD Servers With Emphasis on High-Performance and Low-Cost. *IEEE Transactions on Parallel and Distributed Systems*, 12(4):363–386, 2001.

[16] Z. Haas and S. Tabrizi. On Some Challenges and Design Choices in Ad-Hoc Communications. In *Proceedings of IEEE MILCOM*, Bedford, MA, October 1998.

[17] G. Horn, P. Knudsgaard, S. Lassen, M. Luby, and J. Rasmussen. A Scalable and Reliable Paradigm for Media on Demand. *IEEE Computer*, 34(9):40–45, September 2001.

[18] J. Hsieh, M. Lin, J. Liu, D.-C. Du, and T. Ruwart. Performance of a Mass Storage System for Video-On-Demand. In *Proceedings of the IEEE INFOCOM Conference*, Boston, MA, 1995.

[19] V. Kalogeraki, D. Gunopulos, and D. Zeinalipour-Yazti. Distributed Information Retrieval in Peer-to-Peer Networks. In *Proceedings of the Eleventh International Conference on Information and Knowledge Management*, November 2002.

[20] S. H. Kim and S. Ghandeharizadeh. Design of Multi-user Editing Servers for Continuous Media. In *Proceedings of the 8th Workshop on Research Issues in Database Engineering (RIDE'98)*, February 1998.

[21] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao. OceanStore: An Architecture for Global-Scale Persistent Storage. In *Proceedings of ASPLOS*, Cambridge, MA, 2000.

[22] R. Lienhart, M. Holliman, Y.-K. Chen, I. Kozintsev, and M. Yeung. Improving Media Services on P2P Networks. *IEEE Internet Computing*, 6(1):73–77, January/February 2002.

[23] S. Microsystems. Jxta. http://www.jxta.org.

[24] B. Özden, A. Biliris, R. Rastogi, and A. Silberschatz. A Disk-Based Storage Architecture for Movie On Demand Servers. *Information Systems*, 20(6):465–482, 1995.

[25] B. Özden, R. Rastogi, P. Shenoy, and A. Silberschatz. Fault-tolerant Architectures for Continuous Media Servers. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, Montreal, Quebec, Canada, June 4-6, 1996*, pages 79–90. ACM Press, 1996.

[26] C. Partridge. *Gigabit Networking*. Addison-Wesley, 1993.

[27] M. Ramanathan, V. Kalogeraki, and J. Pruyne. Finding good peers in peer-to-peer networks. In *Proceedings of the International Parallel and Distributed Computing Symposium*, April 2002.

[28] M. Ripeanu, A. Iamnitchi, and I. Foster. Mapping the Gnutella Network. *IEEE Internet Computing*, 6(1):50–57, January/February 2002.

[29] A. Rowstron and P. Druschel. Storage Management and Caching in PAST, a Large-scale Persistent Peer-To-Peer Storage Utility. In *Proceedings of the 18th SOSP*, Toronto, Canada, 2001.

[30] I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. In *Proceedings of ACM SIGCOMM Conference*, San Diego, CA, August 2001.

[31] M. Vernick, C. Venkatramani, and T. Chiueh. Adventures in Building the Stony Brook Video Server. In *Proceedings of the Forth ACM International Conference on Multimedia*, Boston, MA, November 1996.

[32] A. Victor, J. Stankovic, and S. H. Son. QoS Support for Real-Time Databases. In *IEEE Workshop on QoS Support for Real-Time Internet Applications*, Vancouver, BC, June 1999.

[33] S. Waterhouse, D. Doolin, G. Kan, and Y. Faybishenko. Distributed Search in P2P Networks. *IEEE Internet Computing*, 6(1):68–72, January/February 2002.

[34] B. Yang and H. Garcia-Molina. Comparing Hybrid Peer-to-Peer Systems. In *Proceedings of the 27th VLDB Conference*, Rome, Italy, September 2001.

[35] Z. Zhang, Y. Wang, D. Du, and D. Shu. Video Staging: a Proxy-server-based Approach to End-to-End Video Delivery over Wide-area Networks. *IEEE/ACM Transactions on Networking*, 8(4):419–442, August 2000.

IEEE COMPUTER SOCIETY