Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2015) 23: 1081 – 1097

doi:10.3906/elk-1305-85

Load management in a distributed multimedia streaming environment using a fault-tolerant hierarchical system

Hadi Işık AYBAY[1], **Mohammad Ahmed SHAH**[2,*]
[1]Department of Computer Engineering, Eastern Mediterranean University, Gazimağusa, Northern Cyprus
[2]Department of Computer Engineering, İstanbul Aydın University, İstanbul, Turkey

Received: 10.05.2013 • **Accepted/Published Online:** 15.07.2013 • **Printed:** 10.06.2015

Abstract: In contrast to text-only forms of communications, multimedia uses a combination of audiovisual means alongside textual modes of communication. Streaming multimedia is such multimedia that is constantly delivered by a provider of the multimedia to a client. In streaming multimedia the streamed content is continually presented to and received by the end user. Distributed multimedia systems (DMSs) deliver multimedia content to end-users by means of distributed multimedia databases and distributed information servers. These DMSs are designed to deliver multimedia content over a network. A fault-tolerant redundant hierarchy (Red-HI)-based load management policy for a distributed multimedia streaming system is proposed in this paper. The main objectives of this policy are to provide fault tolerance while increasing the flexibility offered in the network by improving network resource utilization. Another advantage of this policy is that it provides popularity-zone centric dynamic multimedia object placement. The load management and fault tolerance are achieved using Red-HI for the multimedia servers coupled with parent-only query flooding policy. We show through simulations that our scheme outperforms the traditional pure hierarchy employed in most distributed multimedia systems.

Key words: End-to-end performance, quality of service, load balancing, fault tolerance, redundant hierarchy, distributed multimedia streaming, dynamic object placement, video-on-demand

1. Introduction

Increases in network bandwidth, power available to computers, and high storage capacity along with new services and applications have made it possible for networks to deliver not only simple text but also graphics and sophisticated multimedia content [1–3]. The advance from the textual to the audio/video arena poses many challenges in terms of managing the transmission of high-quality and high-speed data, searching and locating required files in the pool of millions of files stored, and transmitting these files to potentially millions of concurrent users while maintaining a scalable and reliable system of delivery [4–7].

These challenges have incited the interest of researchers in building better and faster systems that would satisfy the growing needs of multimedia consumers [8–10]. From this interest, distributed multimedia systems (DMSs) have emerged. A DMS is defined as an integrated communication, computing, and information system that enables the processing, management, delivery, and presentation of synchronized multimedia information with quality-of-service guarantees [11]. Typically these DMSs consist of a hierarchical configuration of video servers and network switches.

*Correspondence: ahmed.shah@gmail.com

The list of services that may be offered while employing a DMS includes but is not limited to video-on-demand (VoD), digital libraries, and home shopping. Of the many services, interactive VoD has received the attention of many researchers because of the popularity of this service [1–20]. It is expected that a large-scale VoD system would serve thousands if not millions of clients. It is not practical to have a centralized server facility serve all the users, due to tremendous networking requirements; at the same time, it is not possible to have many replicated facilities to respond locally to individual user populations. To strike a balance between availability and cost-effectiveness, hierarchical configurations of multimedia servers and network switches are employed [12]. Typically, a neighborhood server will handle requests from a particular user population and the server on the next level will handle requests from several user populations if they cannot be handled at their local servers. This hierarchical configuration provides scalability in addition to providing cost efficiency. Scalability is provided by using clusters of storage nodes with disk arrays. Concurrent access to multiple requests and parallel access to multiple disks for the same request in the disk array provide better throughput [12].

The proposed system aims to realize optimized streaming of multimedia objects from a server containing the multimedia item, belonging to a set of geographically distributed servers, to the clients making requests for these objects. For this purpose, a set of interconnected, hierarchical multimedia servers is used, where each server is capable of both streaming and storing multimedia objects. The target is to optimize the bandwidth used for streaming, as well as the load on the servers that belong to this distributed environment. Another objective of this study is to dynamically distribute multimedia objects in the system based on their demand. This dynamic distribution should make it possible for the clients to find a requested object with minimum streaming packet hop from server to server.

In the proposed streaming system, the streaming is realized in two phases. In the first phase, the object to be streamed is located; this step is called 'Object Location'. In the second phase, the located item is streamed continuously from the node containing the object to the client that requested the multimedia object, going through all intermediate servers (if any). This step is called 'Object Delivery'.

The rest of this paper is organized as follows. In Section 2, related work is listed and in Section 3 the system architecture is detailed. In Section 4, the life cycle of a request is discussed. Section 5 contains the policy for dynamic object placement and Section 6 gives the details of the fault tolerance included in this system. Section 7 presents the performance evaluation. Finally, a conclusion is provided in Section 8.

2. Related work

Distributed multimedia content management has received wide attention [21–27]. Almeida et al. [21] considered minimizing the delivery cost in a fixed topology using optimization of content replication as well as routing. Boufkhad et al. [22] investigated the maximum number of videos that can be served by a set of peers. Zhou et al. [23] focused on throughput and how to maximize it on video servers while minimizing the load imbalance in the system. Tan and Massoulie [26] presented a method for optimal content placement in P2P networks. Their goal was to maximize the utilization of peer uplink bandwidth resources. Applegate et al. [27] looked at content placement in a mixed integer program for a system that considers link bandwidth and disk space constraints.

In network resource utilization, Borst et al. [28] proposed a link bandwidth utilization method using a tree structure with limited depth. Valancius et al. [29] proposed an LP-based heuristic to calculate the number of video copies placed at customer home gateways. Zhou and Xu [30] presented a way to optimize video replications on a cluster of VoD servers.

Topology building, being a central design criterion in distributed streaming systems, has been covered in

various research works [31–33]. Zhang et al. solved the problem of optimal P2P streaming under node degree constraints [34].

3. System architecture

A layered approach is used for placing the multimedia servers in the network because with geographically distributed servers, the connections as well as the communications between the servers (nodes) can be handled with relative ease using a layered approach. Furthermore, resources such as bandwidth and memory can be better allocated based on the functionality of the layer that a multimedia server belongs to, as each layer has a predefined functionality in the system.

We assume three categories of layers, namely entry, intermediate, and root level layers. There are single entry level and root level layers in the system, whereas there may be many intermediate level layers. A simple schematic representation of the system is depicted in Figure 1. Redundant hierarchy (Red-HI) [13] is used to connect the servers in each layer to their parents in the adjacent layer.

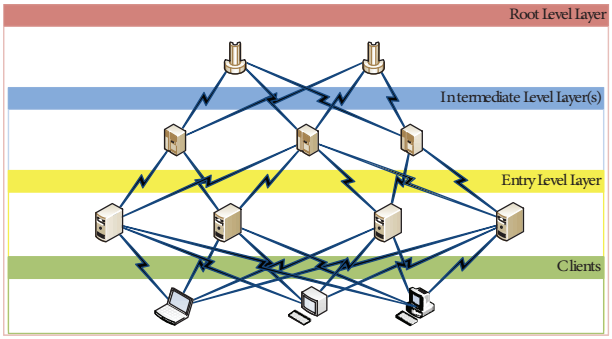

Figure 1. Network architecture of the streaming servers in the system.

3.1. Assumptions

It is assumed that the following are provided for the policy to work:

- All nodes have exactly two parent nodes based on Red-HI [13].
- Both the parent nodes belong to the same immediate parent layer of the node.
- Root nodes have no parent nodes.
- Each node can store multimedia data.
- Each node is a multimedia server in itself with multimedia processing capabilities.
- Only entry level servers, also known as head-end servers, communicate with the clients.

3.2. Entry level layer

This layer is responsible for all direct communication between the clients and the system. The servers in this layer act as head-ends to the system and receive all requests from the clients. Once the requested object is located, these head-end servers deliver the requested stream from the system of servers to the requesting client.

As shown in Table 1, it is assumed that the head-end servers have relatively low storage capacity but a high bandwidth capacity. The storage capacity of these servers is low because they only need to keep the current streaming object and a few (popular) previously streamed objects in their memory. The bandwidth capacity allocated to these servers is high as they are responsible for streaming to a possibly large set of clients. The servers in this layer have designated parent servers in the adjacent intermediate layer. Each server in this layer has exactly two parent servers in the adjacent intermediate layer that can help it in realizing the streaming of a requested multimedia object.

Table 1. Resource distribution in the system.

Layer	Capacity	Bandwidth
Root level layer	High	Low
Intermediate level layer	Medium	Medium
Entry level layer	Low	High

3.3. Intermediate level layer

The servers in this layer act as intermediaries to the system. When a request is received from a lower level server and the requested object is not available in the memory of intermediate level servers, it forwards the request to the layer above and waits for the response from that higher level layer. Once the requested object is located, the corresponding intermediate level server informs the lower level layer about the location and resources required along with the current load on the servers involved in the path of the requested object.

Once the streaming commences, a server of this layer, involved in streaming of a multimedia object, forwards the packets available locally in its memory or retrieved from a higher layer to the requesting server in the adjacent lower level layer.

Each server in this layer has a small set of child nodes from the adjacent lower layer that depend on it to locate and deliver multimedia objects; at the same time, each server in this layer has exactly two servers in the adjacent higher layer that can help it in realizing the streaming of a requested multimedia object.

The servers of the intermediate level are given medium bandwidth and storage capacities as they tend to the needs of a small set of the system requests and as such do not have a very high bandwidth requirement.

Furthermore, they keep the currently streamed objects as well as recently streamed objects in their repository and hence need a higher storage capacity, as shown in Table 1.

In a large system containing multiple intermediate level layers, as you go higher in the hierarchy the bandwidth requirements are reduced. On the other hand, the storage requirements increase for each increased level in the hierarchy.

3.4. Root level layer

This layer is the main repository of all streams available to the system. If a requested object is not found in the servers of the root level, then that object is not available in any server of the system. The servers in the root level layer are called root servers. Each of these root servers is responsible for requests coming from a subset of servers from the lower level layer (intermediate layer). These servers need high storage capacity as they keep records of all multimedia objects in the system. The root servers should not be actively participating in most of the streaming activities in the system. Hence, low bandwidth capacity is sufficient for these servers.

3.5. Server connections

Physical communication connections exist between the servers of different layers. All connections are duplex and hence can be used for bidirectional communication. There are no connections between the servers of the same layer. The network architecture used is based on Red-HI [13].

In Red-HI a node is connected to exactly two nodes of the higher layer in the hierarchy; the redundancy in Red-HI is that of links and not bandwidth. This means that instead of doubling the bandwidth to increase the link, the available bandwidth is divided among the two links. This gives a higher degree of connectivity and it is a solution to the bottleneck problem. This also provides continued service provision in face of node/link failures [13].

4. Request life cycle

A request for a multimedia object is created by a client. The client sends this request to the system. The system receives the request through an entry level server (head-end). The head-end checks its memory for the requested object, depicted in Figure 2. If the requested object is found it is streamed to the requesting client. If the requested object is not found in the memory of the head-end, the head-end generates a query packet and forwards a copy of that query packet to both intermediate level servers who are its parents.

These intermediate level servers, upon receiving the query packet, check their own repository for the queried item. If the object is available, a positive acknowledgement is sent to the server querying for the object, or else the query packet is forwarded to the two servers in the higher layer of the hierarchy that are parents of the current server. In this way, the query packet will eventually reach a server that contains the requested media or a negative acknowledgement will be generated at the root server.

In case the requested media is located at an intermediate or root server, a positive acknowledgement is passed down the hierarchy to the head-end responsible for initializing this query packet. The head-end may receive multiple acknowledgements. In that case it would select the path with higher resources. The resources available to the server, and all intermediate servers, will be mentioned in the positive acknowledgement packet.

Once the path with the highest resources is selected and reserved for streaming, the head-end propagates a packet through the reserved path to inform the server containing the multimedia packet to start streaming. Upon receiving this packet, the server starts streaming the multimedia object down the hierarchy.

5. Dynamic object placement

Objects are placed dynamically using the 'popularity' counter of each streamed object. Each server in the system maintains its own counter of popularity for each object streamed through it.

Each server that receives a streaming packet from its parent server saves that packet in its local cache memory and then forwards it down the stream. This saving is done to minimize the number of servers involved to manage a possible packet-loss event at the lower layers.

The server receiving the stream packet increments the popularity counter of the currently streamed object by one whenever a new request/query message for this particular object is received by this server. Once the popularity counter reaches a predefined popularity threshold, that popular object is moved from the cache memory into the main memory of that server, as shown in Figure 3.

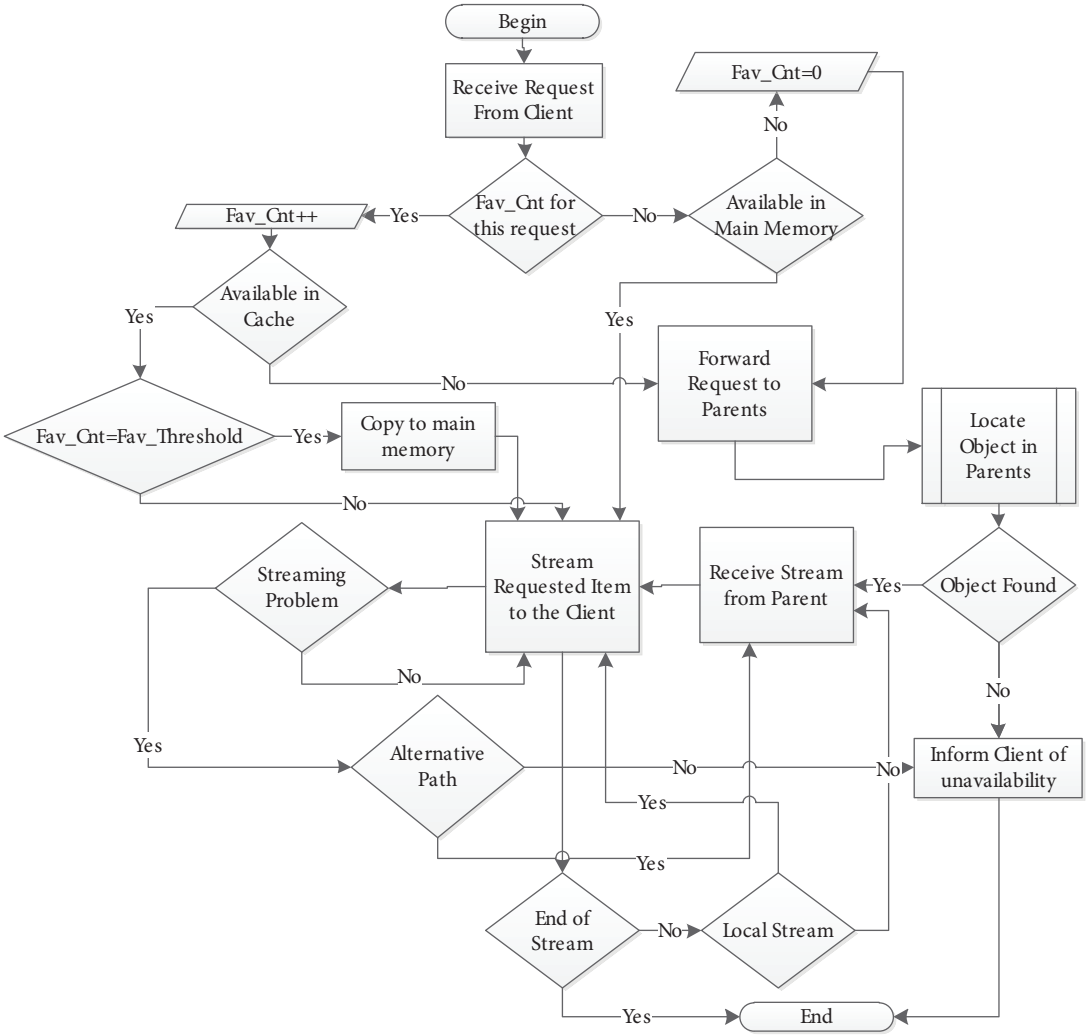

Figure 2. Basic flowchart representing the functioning of a node.

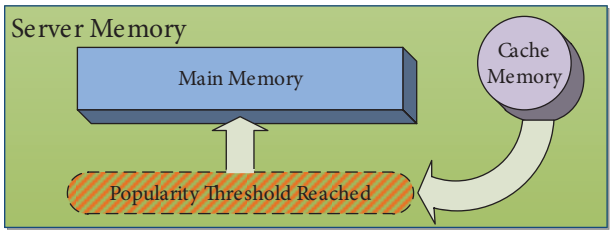

Figure 3. Moving an object from cache to main memory of a server.

The popularity counter helps in determining the popularity of objects among a set of servers and dynamically places those objects in the close vicinity of the requesting clients. Dynamic object placement makes it possible to populate the multimedia objects based on their demand zone and hence optimizes the servers' usage of memory and bandwidth resources.

Once the main memory gets low on storage space, the least recently used media object is removed from the node's memory. Similarly, the least recently used video object is removed in case the cache memory becomes full. No object is removed from the root nodes.

6. Fault tolerance

Figure 4 shows a generic client-server model for end-to-end multimedia data delivery. At the source, encoded/compressed media data stored in storage devices are available. Through media server software these media data are retrieved from the disk for transmission over the network. The system proposed in this study is a server system and hence considers the failures that may occur in a server component.

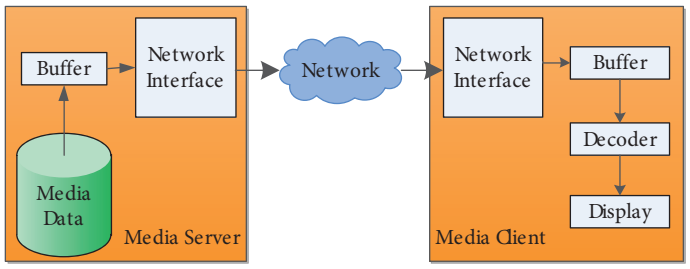

Figure 4. Basic building blocks of a multimedia system.

A media application/transport protocol is used to deliver the media data to the client hosts, where the media data are first buffered and then decoded and presented to the end user.

A problem in any one of the system components, such as storage subsystem, buffers, network, processor, or media codec, can degrade the performance of the whole system. It is imperative that any end-to-end multimedia system provide tolerance to a failure that occurs in as many of these components as possible.

In the object location phase, all possible paths to the requested object are saved at all included nodes. The best path is selected for streaming initially. However, whenever a node feels that problems in its currently streaming parent node are inhibiting its performance, it changes the streaming path using the other parent node, if possible. As all available multimedia objects are available in both root servers, it is always possible for a head-end to find at least two different paths for a given multimedia object, provided that the root servers are not the ones with the fault.

For popular objects, multiple copies at multiple nodes on multiple layers are available, making them more accessible. Even in case of node/link failures in the streaming path, the worst-case scenario would be the delay caused by a change in the nodes involved in the streaming. This delay is of negligible time as it would not be required to locate the object from beginning.

Any failure that occurs during object delivery can be managed by changing the nodes included in the streaming path, other than a physical failure in the head-end. In the case that a head-end component fails, the client has to redirect its request to some other head-end after a timeout. The redirect process will include fresh new object location and object delivery phases.

Figure 5 gives the flowchart for the node that detects a problem in the streaming process. Upon realizing that the streaming process is suffering due to some server component in the higher layers, the node changes the path from itself up to the server that commences the streaming process.

7. Performance evaluation

This section presents the performance evaluation of Red-HI and the performance comparison of the Red-HI and NonRed-HI models. The NonRed-HI environment is one in which the servers are connected using pure hierarchy, i.e. where each server in the system has exactly one parent [13].

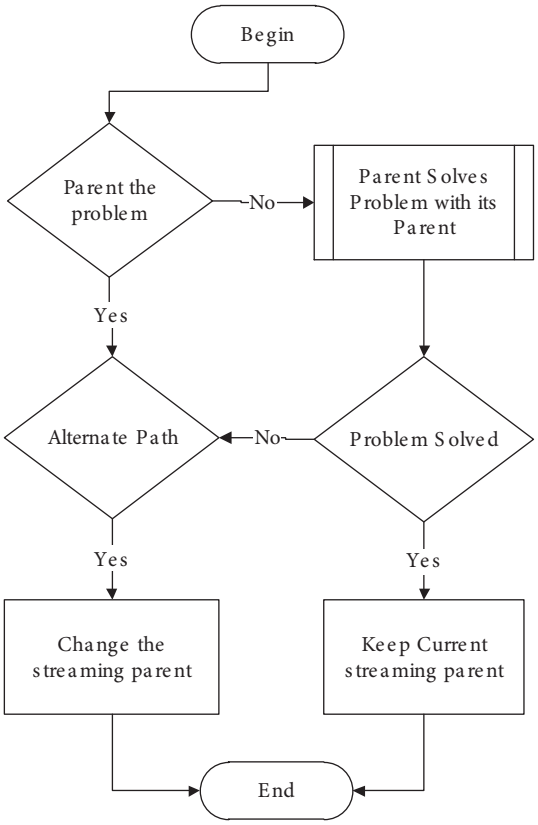

Figure 5. Path change flowchart at a node.

7.1. Performance measures

In this paper, the performance of the Red-HI scheme was evaluated in terms of average transmission delay, average communication delay of successful requests, average number of control messages of successful requests, average number of traversed nodes of successful requests, average number of hops of successful requests, blocking ratio, and load distribution, while the Red-HI and NonRed-HI models were compared based on blocking ratio and load distributions.

Average transmission delay (ATD) is defined as follows:

$$ATD = Ds/Ns, \tag{1}$$

where Ds is the total delay of successful requests and Ns is the total number of successful requests.

Average communication delay of successful requests (ACDSR) is defined as follows:

$$ACDSR = Dc/Ns, \tag{2}$$

where Dc is the total communication delay of successful requests.

Average number of control messages of successful requests (ANCMSR) is defined as follows:

$$ANCMSR = Mc/Ns, \tag{3}$$

where Mc is the total number of control messages of successful requests.

Average number of traversed nodes of successful requests (ANTNSR) is defined as follows:

$$ANTNSR = Tt/Ns, \tag{4}$$

where Tt is the total number of traversed nodes by successful requests.

Average number of hops of successful requests (ANHSR) is defined as follows:

$$ANHSR = Ht/Ns, \tag{5}$$

where Ht is the total number of hops by successful hops.

Blocking ratio (BR) is defined as follows:

$$BR = Nb/Nt, \tag{6}$$

where Nb is the total number of blocked (rejected) requests and Nt is the total number of requests.

7.2. Simulation framework

In this work, all simulation studies were developed using the GPSS World Simulation Tool (http://www.minutemansoftware.com/simulation.htm). We simulated the behavior of the Red-HI and NonRed-HI models depicted in Figures 6 and 7, respectively. Both Red-HI and NonRed-HI architectures consist of an entry level layer with five nodes, intermediate level layer 1 with four nodes, intermediate level layer 2 with three nodes, a and a root level layer with two nodes and one node, respectively.

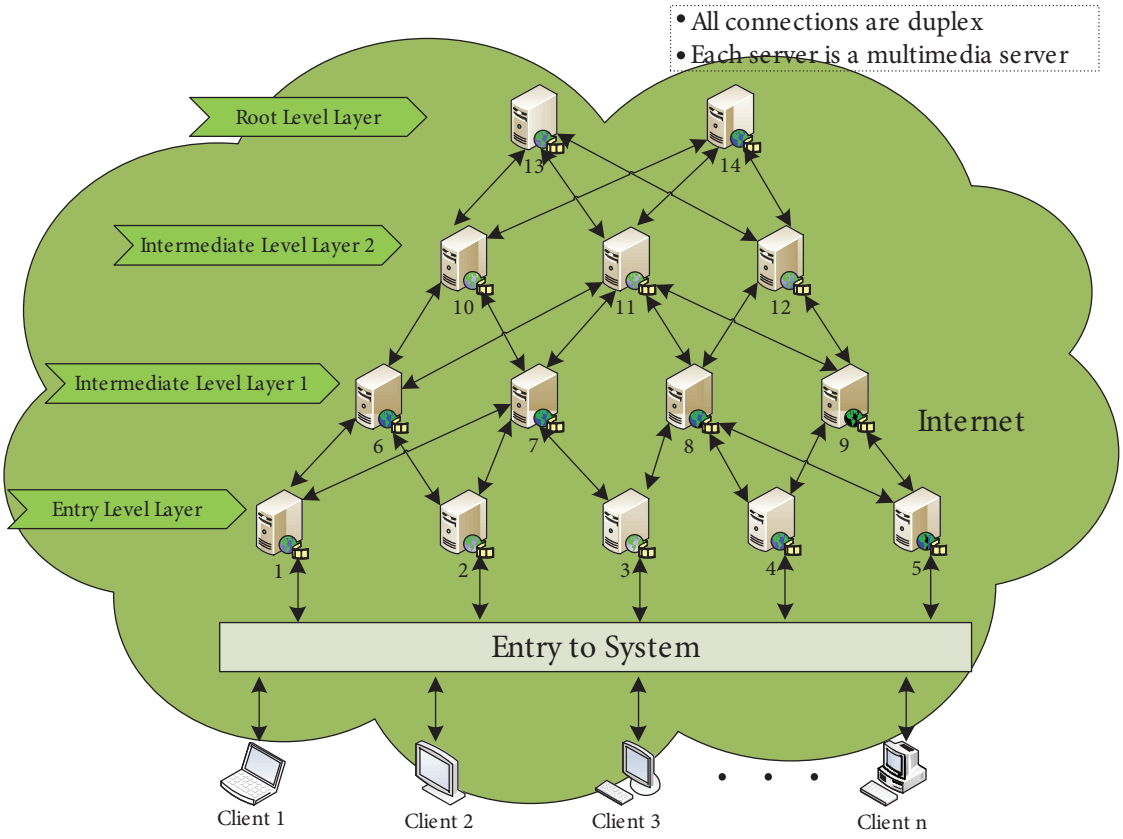

Figure 6. Network architecture of the Red-HI model.

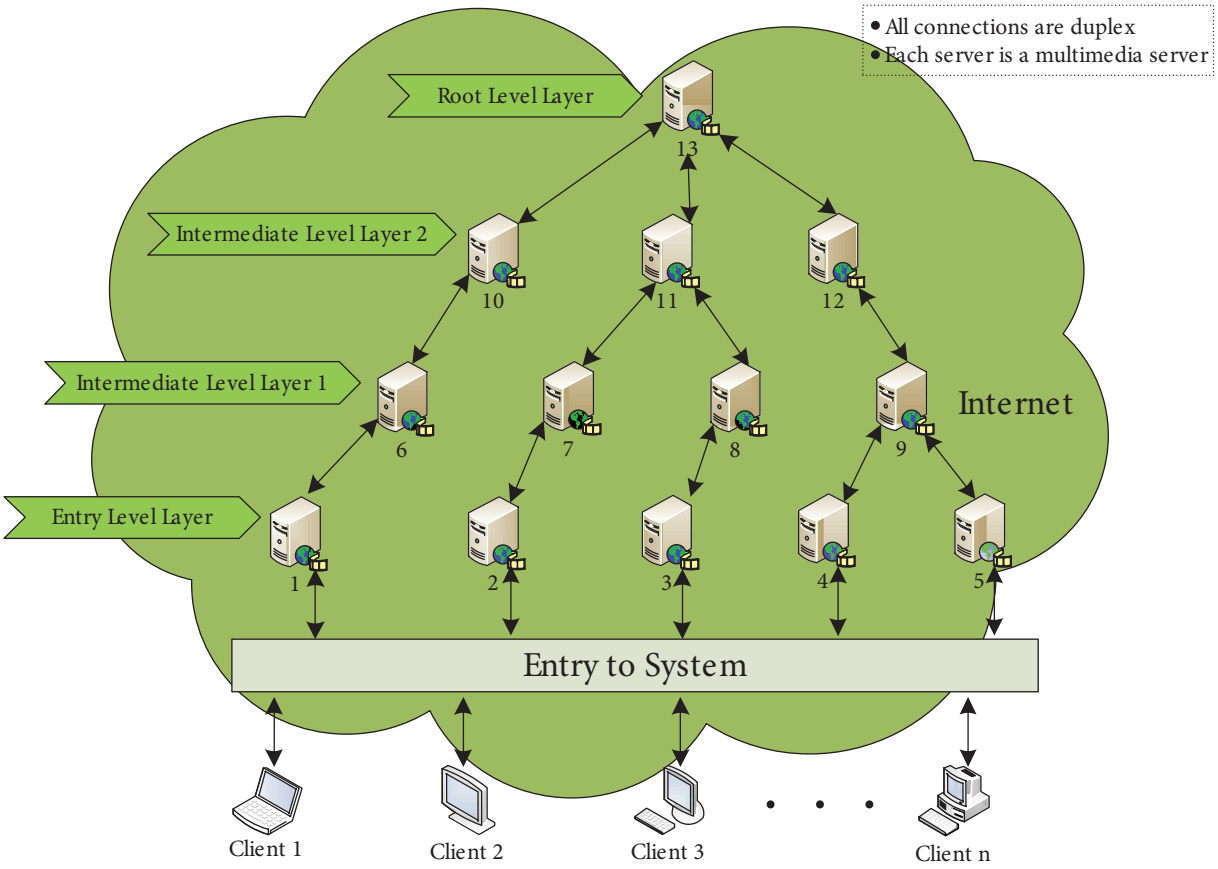

Figure 7. Network architecture of the NonRed-HI model.

We assume that the propagation delay of links follows a normal distribution with mean of 0.07 s and standard deviation of 0.014. The flow of all the client requests in both of the models is modeled as a Poisson process whose rate is λ . The simulation results were obtained for arrival rates of 0.05, 01, 0.15, and 0.2.

In each simulation, 1,000,000 requests were generated and each simulation was run 20 times to ensure that the obtained results were in the 95% confidence interval level. Table 2 contains the values of the simulation parameters.

Table 2. Values of the simulation parameters.

Parameter	Values
Request arrival rate (λ)	0.05, 0.1, 0.15, 0.2 s^{-1}
Popularity threshold	5, 25, 50
Storage size of an object	0.5 GB
Bandwidth requirement of an object	1 Mb/s
Total number of objects	200

7.3. Performance analysis of Red-HI

In this section, simulation results of the Red-HI scheme are discussed. We investigate the effect that variation in the popularity threshold and arrival rate has on the performance of the Red-Hi model.

7.3.1. Average transmission delay

We start with ATD versus arrival rate with popularity threshold values of 5, 25, and 50 as depicted in Figure 8. The graph clearly displays that there is not a significant increase in the transmission delay caused by increasing the number of servers or with substantial increase in the arrival rate of the requests.

7.3.2. Average communication delay and average number of control messages of successful requests

We continue by examining how ACDSR and ANCMSR are affected by the varying arrival rate and popularity threshold. It is seen in Figures 9 and 10 that with increase in the arrival rate the communication delay and the number of control messages in the system remain fairly constant. This gives the system scalability in the event of a large number of requests being handled.

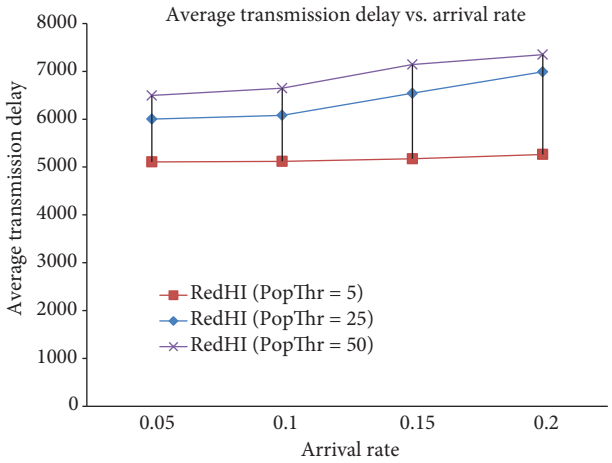

Figure 8. ATD versus arrival rate for Red-HI scheme with popularity threshold values of 5, 25, and 50.

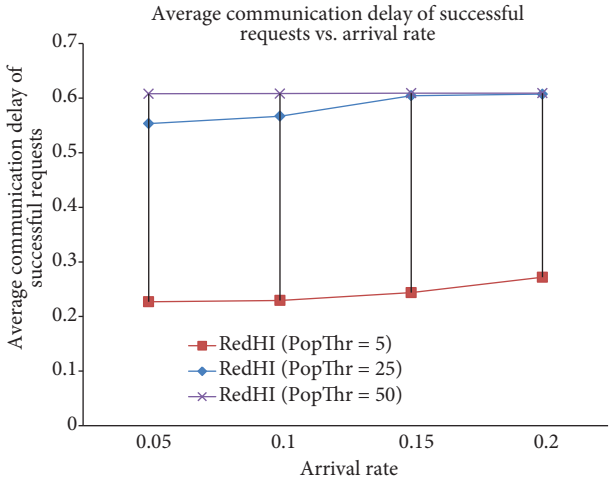

Figure 9. ACDSR versus arrival rate for Red-HI scheme with popularity threshold values of 5, 25, and 50.

7.3.3. Average number of traversed nodes and average number of hops of successful requests

Next we consider how Red-HI performs in terms of ANTNSR and ANHSR. It is obvious that in a redundant hierarchy the number of nodes traversed and the number of hops for a higher popularity threshold would be higher, as is depicted in results graphed in Figures 11 and 12. With popularity threshold of 50 requests for the same object, the system has to stream from the root level much more than for a popularity level of 5 requests making an object popular.

7.3.4. Blocking ratio

Now we look at the effects of varying arrival rate and popularity threshold on the BR. Figure 13 shows that BR values are increasing as the popularity threshold increases. The Red-HI model with popularity threshold 5 achieves the lowest BR. When the popularity threshold is fairly low it causes the content to reach the head-ends faster; as such, the clients get service directly from the head-ends without the need to get service from the intermediate or root nodes. This naturally reduces the BR in load-balanced Red-HI.

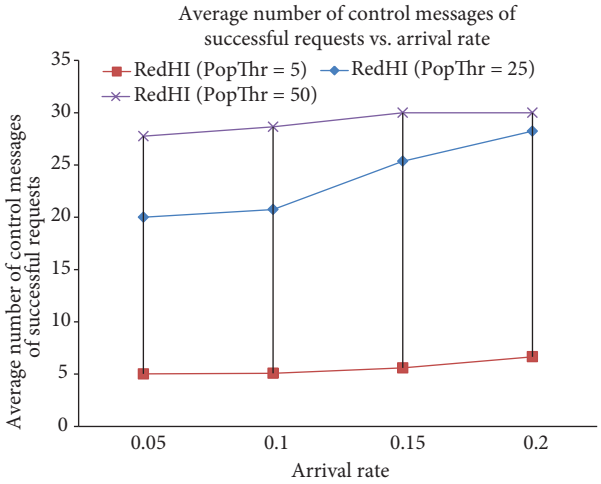

Figure 10. ANCMSR versus arrival rate for Red-HI scheme with popularity threshold values of 5, 25, and 50.

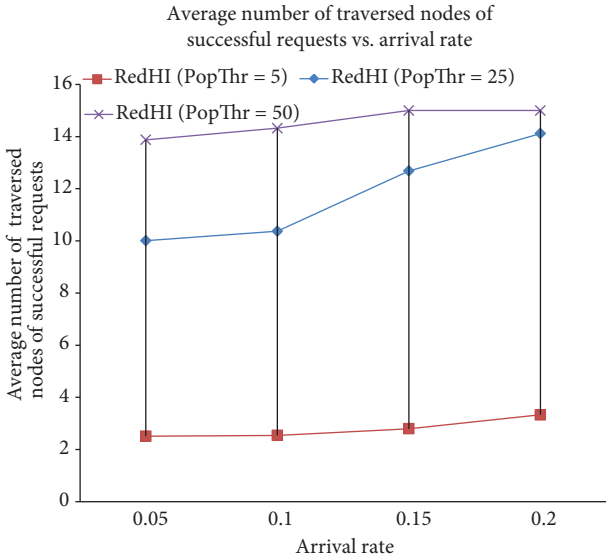

Figure 11. ANTNSR versus arrival rate for Red-HI scheme with popularity threshold values of 5, 25, and 50.

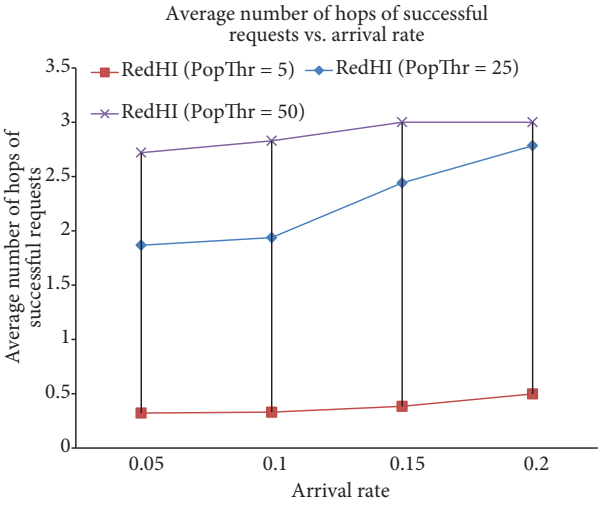

Figure 12. ANHSR versus arrival rate for Red-HI scheme with popularity threshold values of 5, 25, and 50.

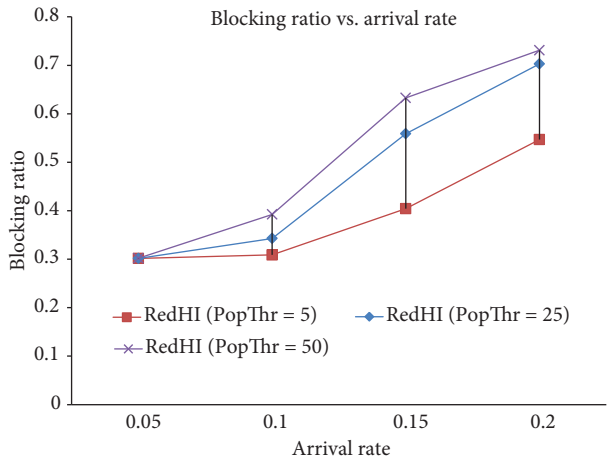

Figure 13. BR versus arrival rate for Red-HI scheme with popularity threshold values of 5, 25, and 50.

7.3.5. Load distribution

We finalize the performance evaluation of the Red-HI scheme by inspecting its performance in terms of load distribution. The following figures depict load distribution of Red-HI under various arrival rates.

Looking at Figures 14–17, it is clearly seen that with increasing arrival rate, the system remains very scalable and the load on the serves remains balanced. These results show that the system achieves its purpose of load balancing under a very high arrival rate of requests for streaming.

To sum up, Figures 8–13 illustrate that Red-HI with popularity threshold 5 outperforms Red-HI with popularity thresholds of 25 and 50 in terms of average transmission delay, average communication delay of successful requests, average number of control messages of successful requests, average number of traversed

nodes of successful requests, average number of hops of successful requests, and blocking ratio. Moreover, Red-HI with popularity threshold 50 demonstrates the worst performance. However, Figures 14–17 show that Red-HI with popularity threshold 25 attains better load distribution then Red-HI with popularity thresholds of 5 and 50.

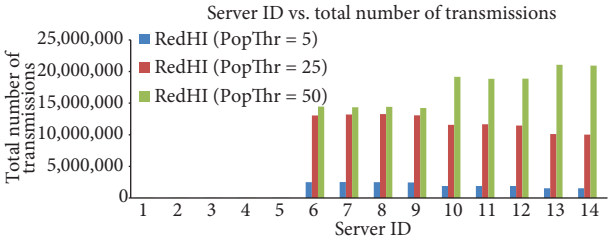

Figure 14. Server ID versus total number of transmissions with arrival rate of 0.05 and popularity threshold values of 5, 25, and 50.

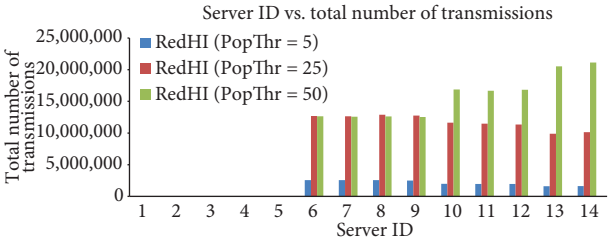

Figure 15. Server ID versus total number of transmissions with arrival rate of 0.1 and popularity threshold values of 5, 25, and 50.

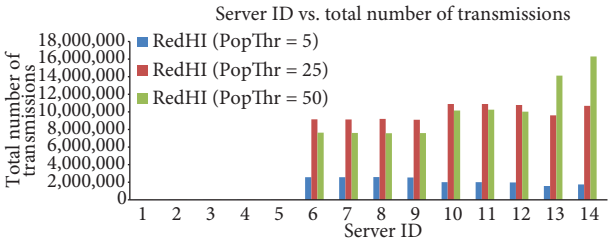

Figure 16. Server ID versus total number of transmissions with arrival rate of 0.15 and popularity threshold values of 5, 25, and 50.

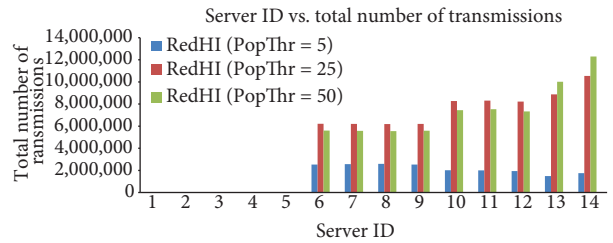

Figure 17. Server ID versus total number of transmissions with arrival rate of 0.2 and popularity threshold values of 5, 25, and 50.

7.4. Comparison of Red-HI and pure hierarchy

In this section, we compare the performances of Red-HI and NonRed-HI (pure hierarchy) in terms of blocking ratio and load distribution.

In earlier graphs we saw how different levels of popularity provided load balancing and scalability at different arrival rates. Figure 18 depicts a comparison of Red-HI and pure hierarchy in terms of blocking ratio.

In Figure 18 we see that compared to a pure hierarchy, our proposed system works better under all arrival rates. This is achieved by the availability of the multiple paths to the same multimedia object, i.e. if one path fails, object transmission will continue through an alternate path. Pure hierarchy, however, lacks the same flexibility, and hence it achieves a significantly higher blocking ratio compared to our proposed system.

Figures 19–22 provide a comparison of Red-HI and pure hierarchy in terms of load distribution. By examining the figures it can be seen that for Red-HI the total number of transmissions is fairly distributed among the servers of the same level. For example, servers 6, 7, 8, and 9 have similar numbers of total number of transmissions; the situation is the same for the servers of all the layers in the system. However, pure hierarchy does not provide fair distribution of total number of transmissions for the servers of the same level. For instance, servers 6, 7, 8, and 9 do not achieve a just distribution of total number of transmissions; the situation is same for the servers of all the layers in the system. We can conclude that Red-HI achieves better load distribution then pure hierarchy under various arrival rates. It should be noted that in these graphs there is no node to

compare with node 14 of pure hierarchy, as pure hierarchy is single-rooted and has highest node of 13 in these simulations.

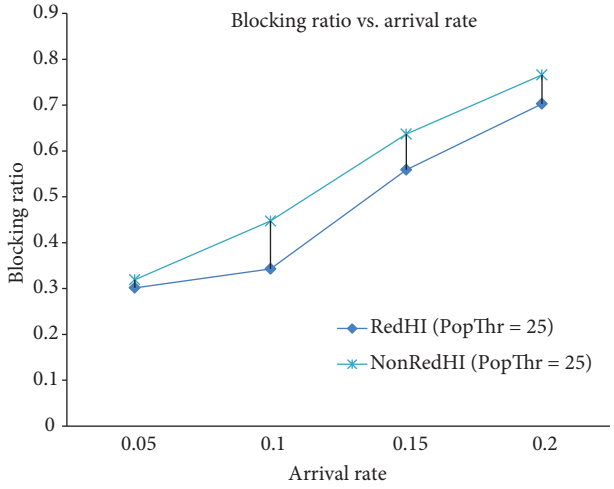

Figure 18. BR versus arrival rate for Red-HI and pure hierarchy with popularity threshold of 25.

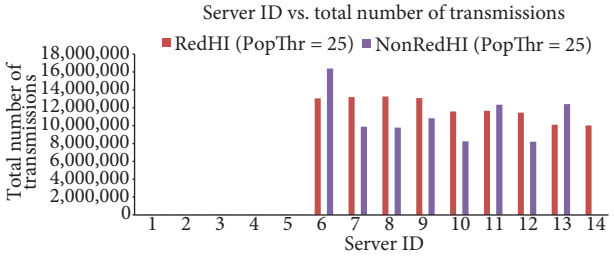

Figure 19. Server ID versus total number of transmissions for Red-HI and pure hierarchy with arrival rate of 0.05 and popularity threshold of 25.

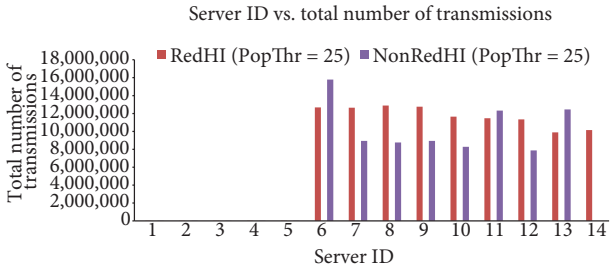

Figure 20. Server ID versus total number of transmissions for Red-HI and pure hierarchy with arrival rate of 0.1 and popularity threshold of 25.

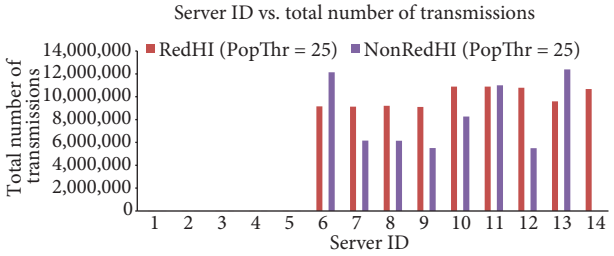

Figure 21. Server ID versus total number of transmissions for Red-HI and pure hierarchy with arrival rate of 0.15 and popularity threshold of 25.

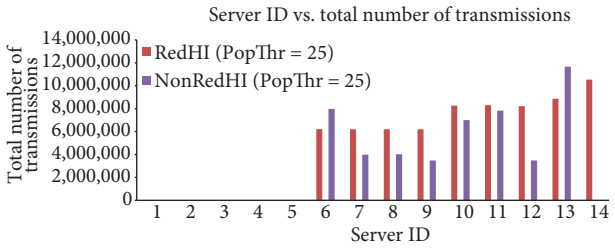

Figure 22. Server ID versus total number of transmissions for Red-HI and pure hierarchy with arrival rate of 0.2 and popularity threshold of 25.

8. Conclusion

A load-balancing and inherently fault-tolerant policy for streaming multimedia objects in a distributed environment is proposed. The proposed system uses Red-HI topology and a two-parent policy per node as the basis for its working.

It is shown that this system dynamically places the multimedia objects in the distributed multimedia server system based on the popularity zone of that particular multimedia object. When storage space demands removal of some multimedia objects on a server, the least popular object is removed. The removed object is always available at a higher level in the hierarchy.

Flexibility is achieved as there are multiple paths to a multimedia object. This not only facilitates near access to a resource but also gives a chance to flexibly select the most suitable node for a streaming session.

The performance of the system is shown in Section 7 of this paper. It is evident from the results that the system is highly scalable, resiliently reliable, and fault-tolerant; furthermore, it provides a superior delivery mechanism as compared to the traditional pure hierarchy used in most DMS systems simply by better allocating the resources available.

References

[1] Sarper H, Aybay I. Improving VoD performance with LAN client back-end buffering. IEEE Multimedia 2007; 14: 48–60.

[2] Amir Y, Danilov C, Goose S, Hedqvist D, Terzis A. An overlay architecture for high-quality VoIP streams. IEEE T Multimedia 2006; 8: 1250–1262.

[3] Guo L, Chen S. Zhang X. Design and evaluation of a scalable and reliable P2P assisted proxy for on-demand streaming media delivery. IEEE T Knowl Data En 2006; 18: 669–682.

[4] Xia Z, Hao W, Yen IL, Li P. A distributed admission control model for QoS assurance in large-scale media delivery systems. IEEE T Parall Distr 2005; 16: 1143–1153.

[5] Zhang A, Song Y, Mielke M. Netmedia: streaming multimedia presentations in distributed environments. IEEE Multimedia 2002; 9: 56–73.

[6] Dimitrova N, Zhang HJ, Shahraray B, Huang I, Zakhor A. Applications of video-content analysis and retrieval. IEEE Multimedia 2002; 9: 42–55.

[7] Wu D, Hou YT, Zhu W, Lee HJ, Chiang T, Zhang YQ, Chao HJ. On end-to-end architecture for transporting MPEG-4 video over the internet. IEEE T Circ Syst Vid 2000; 10: 923–941.

[8] Luling R. Static and dynamic mapping of media assets on a network of distributed multimedia information servers. In: IEEE 1999 International Conference on Distributed Computing Systems; 31 May–4 June 1999; Austin, TX, USA. New York, NY, USA: IEEE Computer Society. pp. 253–260.

[9] Tran DA , Hua KA, Do TT. A peer-to-peer architecture for media streaming. IEEE J Sel Area Comm 2004; 22: 121–133.

[10] Chakareski J, Frossard P. Adaptive systems for improved media streaming experience. IEEE Commun Mag 2007; 45: 77–83.

[11] Li VO, Liao W. Distributed multimedia systems. P IEEE 1997; 85: 1063–1108.

[12] Furht B, Kalra D, Rodriguez AA, Wall WE. Design issues for interactive television systems. IEEE Computer 1995; 28: 25–39.

[13] Shahabi C, Banaei KF. Decentralized resource management for a distributed continuous media server. IEEE T Parall Distr 2002; 13: 710–727.

[14] Zhou X, Xu C. Efficient algorithms of video replication and placement on a cluster of streaming servers. J Netw Comput Appl 2007; 30: 515–540.

[15] Kusmierek E, Du D. Proxy-assisted periodic broadcast for video streaming with multiple servers. Multimed Tools Appl 2008; 36: 243–266.

[16] Kawano Y, Hashimoto K, Shibata Y. Design of a cooperative video streaming system on community based resource sharing networks. In: 3PGCIC 2010 International Conference on P2P, Parallel, Grid, Cloud and Internet Computing; 4–6 November 2010; Fukuoka, Japan. New York, NY, USA: IEEE Computer Society. pp. 483–488.

[17] Tsai Y, Hsu JK, Wu YE, Huang WF. Distributed multimedia content processing in ONVIF surveillance system. In: ICFCSA 2011 International Conference on Future Computer Sciences and Application; 18–19 June 2011; Hsinchu, Taiwan. New York, NY, USA: IEEE Computer Society. pp. 70–73.

[18] Gramatikov S, Jaureguizar NF, Cabrera QJ, García SN. Content delivery system for optimal VoD streaming. In: International Conference on Telecommunications; 15–17 June 2011; Graz, Austria. New York, NY, USA: IEEE Computer Society. pp. 487–494.

[19] Jin X. A scalable distributed multimedia service management architecture using XMPP. Proceedings of the 2011 2nd International Congress on Computer Applications and Computational Science - Advances in Intelligent and Soft Computing 2012; 145: 139–145.

[20] Song FF, Gao W, Zhang G, Gao D, Jiang H. A P2P based video on demand system for embedded Linux. Procedia Engineering 2012; 29: 3070–3074.

[21] Almeida JM, Eager DL, Vernon MK, Wright SJ. Minimizing delivery cost in scalable streaming content distribution systems. IEEE T Multimedia 2004; 6: 356–365.

[22] Boufkhad Y, Mathieu F, Montgolfier FD, Perino D, Viennot L. Achievable catalog size in peer-to-peer video-on-demand systems. In: International Workshop on Peer-to-Peer System; February 2008; Tampa Bay, Florida, USA. Ithaca, NY, USA: IPTPS. pp. 1–6.

[23] Zhou X, Xu CZ. Efficient algorithms of video replication and placement on a cluster of streaming servers. J Netw Comput Appl 2007; 30: 515–540.

[24] Laoutaris N, Zissimopoulos V, Stavrakakis I. On the optimization of storage capacity allocation for content distribution. Comput Netw 2005; 47: 409–428.

[25] Wu J, Li B. Keep cache replacement simple in peer-assisted VoD systems. In: Proceedings of the IEEE International Conference on Computer Communications; April 2009; Rio de Janeiro, Brazil. New York, NY, USA: IEEE Computer Society. pp. 2591–2595.

[26] Tan B, Massoulie L. Brief announcement: adaptive content placement for peer-to-peer video-on-demand systems. In: Proceedings of the ACM Symposium on Principles of Distributed Computing; July 2010; Zurich, Switzerland. New York, NY, USA: ACM. pp. 293–294.

[27] Applegate D, Archer A, Gopalakrishnan V, Lee S, Ramakrishnan KK. Optimal content placement for a large-scale VoD system. In: Proceedings of the 6th International ACM Conference on Emerging Networking Experiments and Technologies; December 2010; Philadelphia, PA, USA. New York, NY, USA: ACM. pp. 1–12.

[28] Borst S, Gupta V, Walid A. Distributed caching algorithms for content distribution networks. In: Proceedings of the IEEE International Conference on Computer Communications ; March 2010; San Diego, CA, USA. New York, NY, USA: IEEE Computer Society. pp. 1–9.

[29] Valancius V, Laoutaris N, Massoulie L, Diot, Rodriguez P. Greening the internet with nano data centers. In: Proceedings of the 5th International ACM Conference on Emerging Networking Experiments and Technologies; December 2009; Rome, Italy. New York, NY, USA: ACM. pp. 37–48.

[30] Zhou X, Xu CZ. Optimal video replication and placement on a cluster of video-on-demand servers. In: Proceedings of the International Conference on Parallel Processing; August 2002; Vancouver, Canada. New York, NY, USA: IEEE Computer Society. pp 547–555.

[31] Liu Y, Liu X, Xiao L, Ni LM, Zhang X. Location-aware topology matching in p2p systems. In: Proceedings of the IEEE International Conference on Computer Communications; March 2004; Hong Kong. New York, NY, USA: IEEE Computer Society. pp. 2220–2230.

[32] Laoutaris N, Carra D, Michiardi P. Uplink allocation beyond choke/unchoke. In: Proceedings of the 4th International ACM Conference on Emerging Networking Experiments and Technologies; December 2008; Madrid, Spain. New York, NY, USA: ACM. pp. 18–18.

[33] Aggarwal V, Akonjang O, Feldmann A. Improving user and ISP experience through ISP-aided P2P locality. In: Proceedings of the IEEE International Conference on Computer Communications; April 2008; Phoenix, AZ, USA. New York, NY, USA: IEEE Computer Society. pp. 1–6.

[34] Zhang S, Shao Z, Chen M. Optimal distributed P2P streaming under node degree bounds. In: Proceedings of the IEEE International Conference on Network Protocols; October 2010; Kyoto, Japan. New York, NY, USA: IEEE Computer Society. pp. 253–262.