JCSE Journal of Computing Science and Engineering

Regular Paper
Journal of Computing Science and Engineering,
Vol. 9, No. 3, September 2015, pp. 134-141

Optimizing Caching in a Patch Streaming Multimedia-on-Demand System

Dinkisa Aga Bulti*
Department of Electrical and Computer Engineering, Wolkite University, Wolkite, Ethiopia
dinqiisa@gmail.com

Kumudha Raimond
Department of Computer Science and Engineering, Karunya University, Coimbatore, India
kraimond@karunya.edu

Abstract
In on-demand multimedia streaming systems, streaming techniques are usually combined with proxy caching to obtain better performance. The patch streaming technique has no start-up latency inherent to it, but requires extra bandwidth to deliver the media data in patch streams. This paper proposes a proxy caching technique which aims at reducing the bandwidth cost of the patch streaming technique. The proposed approach determines media prefixes with high patching cost and caches the appropriate media prefix at the proxy/local server. Herein the scheme is evaluated using a synthetically generated media access workload and its performance is compared with that of the popularity and prefix-aware interval caching scheme (the prefix part) and with that of patch streaming with no caching. The bandwidth saving, hit ratio and concurrent number of clients are used to compare the performance, and the proposed scheme is found to perform better for different caching capacities of the proxy server.

Category: Ubiquitous computing

Keywords: Caching scheme; Batch Streaming; Patch streaming; Proxy caching; Multimedia-on-demand

I. INTRODUCTION

Multimedia files like audio and video files are usually called continuous because of their unique characteristics that are not found in discrete files. Due to its high data rate, large memory and real-time transfer requirements, streaming continuous media needs the server resources to be reserved in advance. In addition, the streaming tends to last for a relatively long period of time for watching or listening, which consumes large amounts of resources. The other most important characteristic of continuous data is the interactivity requirement to play, pause, and stop among others [1-3].

These continuous media data characteristics are the causes for the difficulties of media content delivery. To solve these problems, different mechanisms are proposed and are in use. One of these techniques is data and resource sharing which is usually achieved through caching and streaming techniques. Caching uses the reference locality in the media request pattern to serve subsequent clients and is usually done at the proxy or local server [1, 4].

Media streaming is another resource sharing method. It

Open Access http://dx.doi.org/10.5626/JCSE.2015.9.3.134 http://jcse.kiise.org

Received 20 October 2014; **Revised** 07 April 2015; **Accepted** 05 July 2015
*Corresponding Author

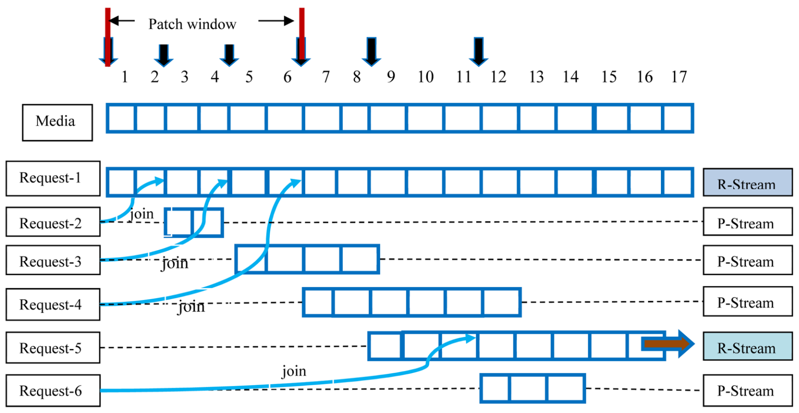

Fig. 1. Patch streaming.

involves playback of the media while receiving. The common streaming techniques are batch streaming [5], piggybacking [6], and patching [7]. Batch streaming is a technique with which requests by multiple clients for the same video arriving within a short time interval are batched together and serviced using a single multicast stream. Piggybacking [6] is a streaming technique with which the later stream is allowed to speed up its streaming bit rate so as to catch up the previous stream and join it. Patching [7] initially uses a complete multicast stream or regular stream (R-stream) and then a new unicast patch stream (P-stream) for each subsequent request. When the first request for a video arrives, the server sets up a multicast stream so that later users can also join the stream and receive the missed portion of the media in the patch stream, as shown in Fig. 1. This reduces the request waiting time but requires additional server bandwidth and buffer space at the client than does batch streaming [8].

Usually caching and streaming techniques are combined to enhance the performance of multimedia-on-demand (MOD) systems. A number of caching techniques are proposed [1, 4, 9-12], and some are optimized to work with a batch streaming system. These caching techniques are mostly used as a means of decreasing the start-up latency since batch streaming is characterized by large inherent latency. With a patch streaming system, there is no inherent start-up latency except network delay. The problem with patch streaming is the extra bandwidth requirement to deliver the P-streams. Therefore, the caching technique in patch streaming MOD system needs to be designed from the point of view of the requirement of bandwidth minimization.

The paper formulates the patching cost when a certain number of blocks of each media object are cached at the proxy server. It also proposes an optimization scheme using proxy caching of the media prefixes to minimize the total cost of patching for all media in P-streams. Then, it presents the system implementation and evaluations of the proposed scheme.

II. RELATED WORK

Due to the large size of media objects and clients' partial viewing patterns [2, 13-15], *selective* or *partial* caching is always being used. These techniques can be divided into two categories: interval-based [1, 9-11] caching and prefix and interval-based [4, 10, 12] caching.

Interval-based schemes: Interval-based caching schemes include interval caching (IC) [1], popularity-aware interval caching (PIC) [9], and client-assisted interval caching (CIC) [11]. All interval-based caching techniques are characterized by caching intervals between two consecutive requests on a single media object. The IC scheme introduced in [1] sorts the intervals by increasing size and caches the shortest. The largest interval is the victim of replacement. The PIC scheme [9] uses the reference popularity of multimedia objects as well as the time interval between two consecutive requests to predict the time of the next request and include it in the candidates for caching.

Prefix and interval-based schemes: Popularity and prefix-aware interval caching (2PIC) [10], optimal prefix caching and data sharing (OPC-DS) [10], and dynamic buffer allocation [12] are designed to minimize start-up latency and improve bandwidth utilization. The 2PIC scheme uses the reference popularity information to decide the targets of prefix caching and calculates the prefix interval as in (1).

$$PI_n = \alpha I_{n-1} + (1-\alpha) Avg_{n-1} \tag{1}$$

Here, α is a constant between 0 and 1, Avg_{n-1} is the $(n\text{-}1)^{th}$ average interval from the 1st to the $(n\text{-}1)^{th}$ request, and $I_{n\text{-}1}$ is the $(n\text{-}1)^{th}$ real interval.

However, the prefix interval thus obtained is not optimized in terms of the back-bone network bandwidth consumption and there is no consideration of the fact that the patch stream must be transmitted frequently to achieve minimal start-up latency. Li et al. [4] assume that the streaming technique does not involve any caching at the client and also that the client receives only a single stream (i.e., cannot be used with patch streaming).

III. OPTIMIZING PREFIX CACHE SIZE AND SELECTION

A. Patching Cost

In this work, the cost of patching is defined as the total data size that has to be delivered in P-streams (Fig. 1), which is directly related to the extra bandwidth requirement of the technique.

If a constant bit rate is assumed, the number of data blocks in each P-stream is related to the time gap between the arrival time of the request and the start time of recent R-stream. The number of blocks B_i in P-streams of one R-stream in seconds (assuming a streaming time of one block of data equivalent to one second) is:

$$B_i = \sum_{n=1}^{\frac{Pw_i}{s_i}} ns_i = \frac{1}{s_i} Pw_i(Pw_i+s_i)/2 \qquad (2)$$

Here, Pw_i is the patch window size in seconds, s_i is the average inter-arrival time for media-*i* in seconds, *n* is the n^{th} request that can join the existing R-stream, and ns_i is the arrival time for the n^{th} request.

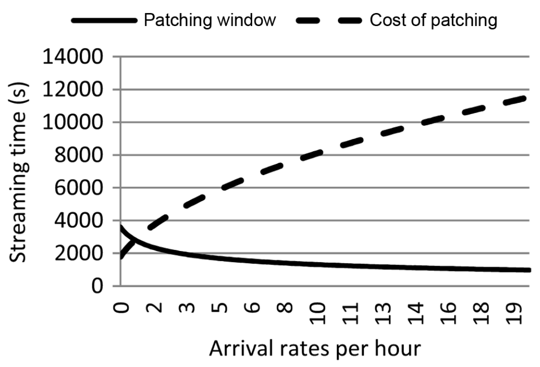

Fig. 2. Patch window and cost of patching vs. arrival rate.

From Fig. 2, it can be seen that the cost of patching increases with the arrival rate indicating that more popular media objects have higher cost than less popular media objects.

As a way of reducing this patching cost of data in the P-stream, *transition patching* and *recursive patching* have been proposed [16], but at an increased cost of handling multiple channels simultaneously.

B. Cost of Patching with Prefix Caching

If a few blocks of data are cached at the proxy server, the main media server has to stream only the portion that is not cached, as shown in Fig. 3, and subsequent requests can be served from the cached prefixes. Since the cost of patching depends on the popularity of the media object, the caching scheme should be able to select prefixes of the most popular media.

Modifying (2) to include the prefix size Cp_i, of the media object cached at the proxy server, the number of data blocks B_i to be delivered in P-streams for each R-stream of media-*i*, assuming a constant bit-rate, can be obtained as in (3) or (4) below.

$$B_i = \sum_{n=\frac{Cp_i}{s_i}+1}^{\frac{Pw_i}{s_i}} (ns_i - Cp_i) \qquad (3)$$

$$B_i = \left(\frac{1}{2s_i}\right)[Pw_i(Pw_i+s_i)+(Cp_i^2)-Cp_i(s_i+2Pw_i)] \qquad (4)$$

B_i, Pw_i, and Cp_i represent the times that are required to stream the data blocks in each media stream. For *M* media objects, the total amount of media data B_T in P-streams that has to be delivered in each of the respective patch windows of media objects from the main media

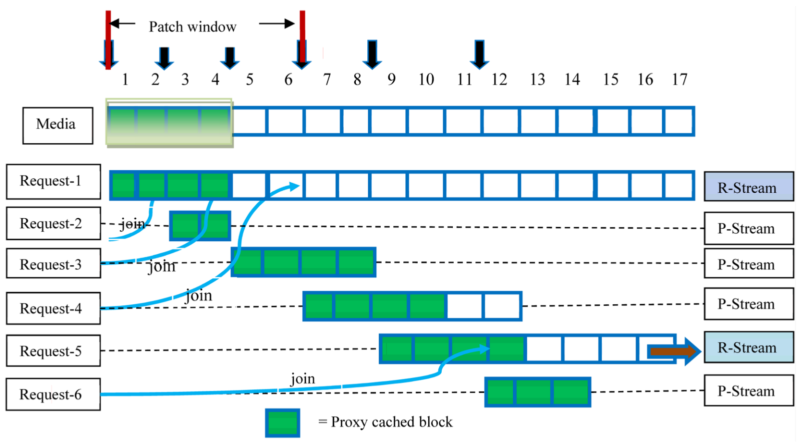

Fig. 3. Patching and caching.

server in terms of the time that is required is:

$$B_T = \sum_{i=1}^{M} B_i$$

$$B_T = \sum_{i=1}^{M}\left(\frac{1}{2s_i}\right)[Pw_i(Pw_i+s_i)+(Cp_i^2)-Cp_i(s_i+2Pw_i)] \quad (5)$$

In the time duration of one R-stream (or playback duration), T_i, there would be a number of R-streams established based on the arrival rate. Corresponding to a media file, the number of R-streams to be created is given by (6).

$$R_i = T_i/Pw_i \quad (6)$$

The number of P-streams, hence data to be transmitted in P-streams, can be approximately calculated as an integral multiple of this number, R_i. To include the effect of difference in the bit rate, we normalize the i^{th}-media bit rate b_i to a reference of 1 Mbps bit rate, and then the total data to be delivered in P-streams corresponding to a particular media object is a multiple of this normalized bit rate. For M media, the total number of blocks in seconds becomes:

$$B_T = \sum_{i=1}^{M}\left(\frac{T_i b_i}{2Pw_i s_i * 1Mbps}\right) [(Cp_i^2)-Cp_i(s_i+2Pw_i)+Pw_i(Pw_i+s_i)] \quad (7)$$

The above Eq. (7) shows the total cost of patching when Cp_i blocks of each media object is cached at the proxy server.

C. Optimizing Prefix Size Selection

The number of blocks to be cached to obtain minimum bandwidth cost is equal to the sum of patch window and half the inter-arrival time. This can be obtained by differentiating (7) above with respect to Cp_i and setting it to zero.

$$\frac{dB_T}{dCp_i} = 0 \quad (8)$$

If the inter-arrival time is assumed to be much less than the patch window size, the number of blocks needed to be cached is equal to Pw_i. This shows that if all data blocks in the P-stream are cached, there is no need for an explicit unicast connection and the only remaining data to be streamed is the R-stream which is already in the resource sharing scheme. However, this is hardly achievable for prefixes of all media objects because of caching capacity limitation. The problem is, can one cache media prefixes up to the maximum capacity and still minimize the total size of the blocks of data for all media in the P-stream?

$$\text{Minimize } B_T = \sum_{i=1}^{M} B_i$$

$$\text{With } \sum_{i=1}^{M} Cp_i \leq C_{max} \quad (9)$$

C_{max}: Maximum caching capacity of the proxy

The problem can be viewed as a fractional knapsack problem by enabling caching of a portion of the data in the P-stream, to minimize the cost. The greedy algorithm [17] is well suited for solving such problems since the cost associated with each size of cache blocks can be found as in (7). The approach is to select prefixes of media objects with higher costs and cache them at the proxy server. The proxy server sorts the media objects based on their patching cost in descending order. Then, the media prefixes are cached starting from the media prefix with highest patching cost to the least which still satisfy the capacity constraint.

IV. SYSTEM MODEL

The model shown in the following diagram (Fig. 4) was used to evaluate the performance of the proposed caching scheme and compare with the 2PIC scheme and patching without caching. The client sub-system is represented by the synthetic workload. The proxy receives client requests from the synthetically generated real-time traffic. The description for each component of the model is given in the following sub-sections.

A. Media Access Workload Models

A synthetic media server workload generator was developed, to simulate the real media access trace. The models used in developing the synthetic workload try to capture as many properties as possible in the user access pattern that are very important with regard to the performance of the system. The generation steps and models used are summarized in the following diagram (Fig. 5).

B. Proxy and Server Managers

The proxy has three threads of control: a thread which

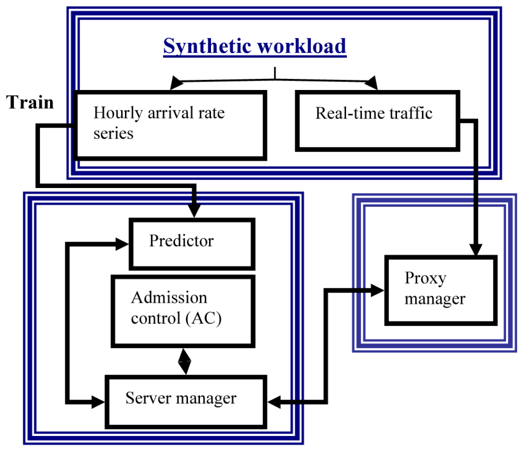

Fig. 4. System model.

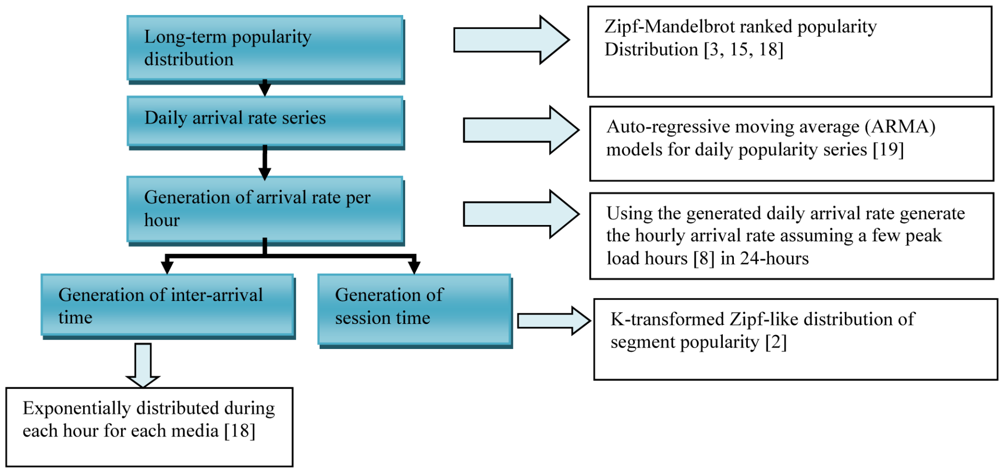
Long-term popularity distribution
Daily arrival rate series
Generation of arrival rate per hour
Generation of inter-arrival time
Generation of session time
Exponentially distributed during each hour for each media [18]
Zipf-Mandelbrot ranked popularity Distribution [3, 15, 18]
Auto-regressive moving average (ARMA) models for daily popularity series [19]
Using the generated daily arrival rate generate the hourly arrival rate assuming a few peak load hours [8] in 24-hours
K-transformed Zipf-like distribution of segment popularity [2]

Fig. 5. Diagram of the synthetic workload generator.

listens to client requests, a thread which listens to the server response, and threads which are responsible for media data streams in cases a request is to be served from the proxy cache.

For each client request, the first thread adds cache information and forwards it to the server. Since the response of the server is asynchronous to the request, the second thread is required to handle the response from the server. The proxy can run under three options: without caching, with caching using the 2PIC scheme, and caching using the proposed scheme.

The server manager establishes three main thread types: a thread which listens to client requests and responds, a thread to run the predictor module and regularly update the patch window size and patching cost information, and threads to control data stream delivery.

When the server accepts a request, it either establishes a new R-stream or the client joins an existing R-stream and receives the missing data in a P-stream.

C. The Predictor Module

In order to find the patching cost of each media and optimize the prefix selection, the arrival rate must be known. This requires the prediction of the arrival rate, at least for the next few hours. For this purpose, the reservoir computing (RC) [20, 21] approach is used in this work. RC has been used for media popularity prediction in [22].

V. RESULTS AND DISCUSSION

A. Bandwidth Savings

Here, the bandwidth saving is defined as the average normalized difference between the bandwidth required by two schemes, as given by (10). The reference required bandwidth is chosen to be of a patch streaming system without caching.

$$BW_{saving} = \frac{1}{N}\sum_{i=1}^{N}\left\{\frac{BW_{yi}-BW_{xi}}{BW_{yi}}\right\} \qquad (10)$$

Here, N is the sample size (the number of instants for which the required bandwidth is measured, in this case the number of playback *start* requests in a one day workload), and BW_{yi} and BW_{xi} are the required bandwidths for scheme y and x, respectively, at instant i (for each instant of request arrival). Here, scheme y is the reference required bandwidth when there is no caching and the scheme x can be either the 2PIC or the proposed scheme.

The maximum required bandwidth using each technique is measured by fixing the bandwidth capacity too large such that there is no media request rejected. The caching capacity of the proxy is varied from 500 MB to 2500 MB (that is equivalent to 3.35% to 16.7% of the total media size on the main server). These maximum

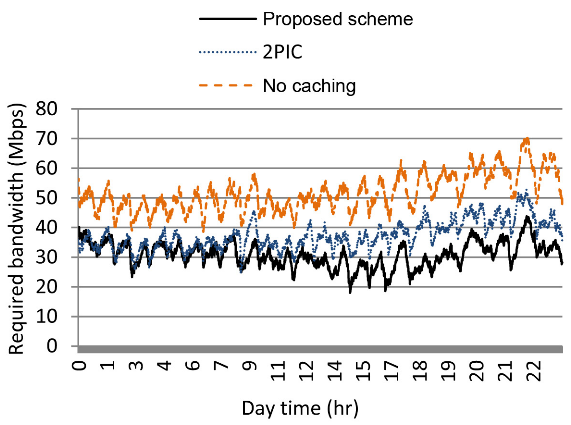

Fig. 6. Maximum required bandwidths with a proxy cache capacity of 2000 MB.

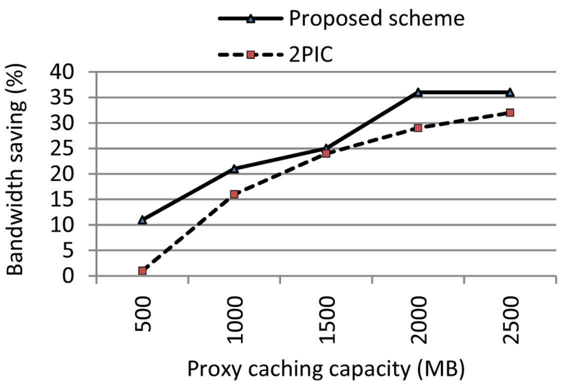

Fig. 7. Bandwidth savings.

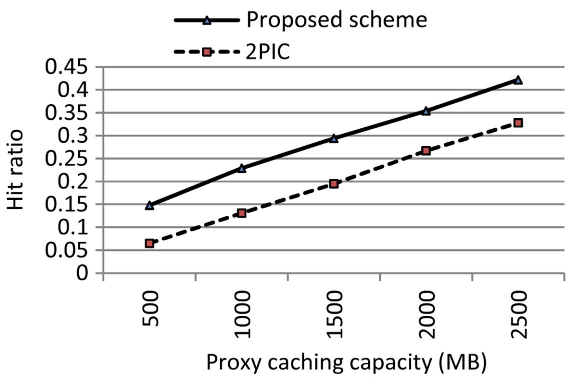

Fig. 8. Hit ratio.

required bandwidths for the patch streaming without caching (or no caching), 2PIC and proposed schemes are measured for all of the accepted requests in a 24-hour or one day workload. Fig. 6 shows the maximum required bandwidths for these schemes at each instance of *start* request arrival for a proxy caching capacity of 2000 MB.

The bandwidth saving is calculated for each proxy caching capacity as in (10) and is shown in Fig. 7. The bandwidth saving of the proposed scheme is larger than the 2PIC scheme for all cache sizes. As can be seen from Fig. 7, bandwidth saving of up to 36% and 32% can be obtained from the proposed scheme and from 2PIC schemes, respectively, for a proxy caching capacity of 2500 MB. The bandwidth saving of the proposed scheme even increases during the peak-load hours as can be seen from Fig. 6 which is the result of the increased number of P-streams that can be served from the proxy cache.

B. Hit Ratio

The variation of the hit ratio with different caching capacities of the proxy server was obtained by running the simulation on 30 days media request data, each with 100 media objects. Fig. 8 shows the hit ratio for both caching techniques (proposed scheme and 2PIC). The proposed scheme gives up to 0.1 higher hit ratio than the 2PIC scheme for each of the cache capacities.

C. Maximum Number of Concurrent Clients

The maximum number of concurrent clients that the admission control (AC) can accept depends on the available resources when the request arrives. Here, the cache capacity of the proxy is fixed to 500 MB and the maximum network bandwidth is varied from 25 to 50 Mbps. The same day media access workload is used for all schemes and the maximum number of concurrent clients is recorded and shown in Fig. 9. The result shows that the proposed scheme admits more clients (up to 90 clients more), followed in performance by the 2PIC scheme. In general, patching without caching accepts fewer requests, as is to be expected.

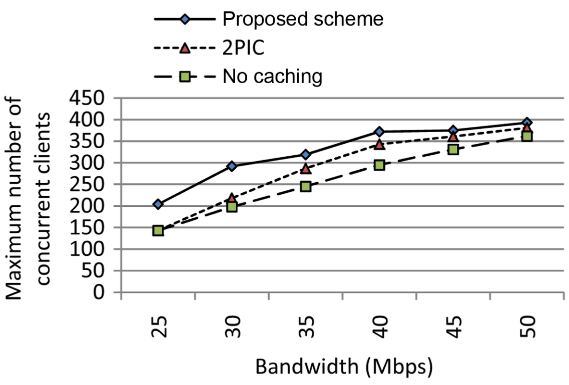

Fig. 9. Maximum concurrent clients.

VI. CONCLUSION

In this paper, a caching scheme which minimizes the bandwidth cost of patch streaming is designed and presented. This proposed approach determines media prefixes with high patching cost and caches the appropriate media prefix at the proxy/local server. The scheme was evaluated using a synthetically generated media access workload and its performance was compared with that of the popularity and prefix-aware interval caching scheme (the prefix part) and with that of patch streaming with no caching. The bandwidth saving, hit ratio and concurrent number of clients are used to compare the performance, and the proposed scheme is found to perform better for different caching capacities of the proxy server.

REFERENCES

1. A. Dan, D. M. Dias, R. Mukherjee, D. Sitaram, and R. Tew-

ari, “Buffering and caching in large-scale video servers,” in *Technologies for the Information Superhighway: Digest Of Papers (COMPCON'95),* San Francisco, CA, 1995, pp. 217-224.

2. J. Yu, C. T. Chou, X. Du, and T. Wang, “Internal popularity of streaming video and its implication on caching,” in *Proceedings of the 20th International Conference on Advanced Information Networking and Applications (AINA’06)*, Vienna, Austria, 2006.
3. W. Tang, Y. Fu, L. Cherkasova, and A. Vahdat, “Modeling and generating realistic streaming media server workloads,” *Computer Networks*, vol. 51, no. 1, pp. 336–356, 2007.
4. K. Li, C. Xu, Y. Zhang, and Z. Wu, “Optimal prefix caching and data sharing strategy,” in *Proceedings of IEEE International Conference on Multimedia & Expo (ICME2008)*, Hannover, Germany, 2008, pp. 465-468.
5. H. Ma and K. G. Shin, “Multicast video-on-demand services,” *ACM SIGCOMM Computer Communication Review*, vol. 32, no. 1, pp. 31-43, 2002.
6. L. Golubchik, J. C. Lui, and R. R. Muntz, “Adaptive piggybacking: a novel technique for data sharing in video-on-demand storage servers,” *Multimedia System*, vol. 4, no. 3, pp. 140-155, 1996.
7. K. A. Hua, Y. Cai, and S. Sheu, “Patching: a multicast technique for true video-on-demand services,” in *Proceedings of the 6th ACM International Conference on Multimedia*, Bristol, UK, 1998, pp. 191-200.
8. J. Choi, M. Yoo, and B. Mukherjee, “Efficient VoD streaming for broadband access networks,” in *Proceeding of IEEE Global Telecommunications Conference (GLOBECOM)*, New Orleans, LA, 2008, pp. 1-6.
9. T. Kim, H. Bahn, and K. Koh, “Popularity-aware interval caching for multimedia streaming servers,” *Electronics Letters*, vol. 39, no. 21, pp. 1555-1557, 2003.
10. O. Kwon, H. Bahn, and K. Koh, “Popularity and prefix aware interval caching for multimedia streaming servers,” in *Proceedings of 8th IEEE International Conference on Computer and Information Technology (CIT2008)*, Sydney, 2008, pp. 555-560.
11. L. Wujuan, L. S. Yong, and Y. K. Leong, “A novel interval caching strategy for video-on-demand systems,” in *Proceedings of 14th IEEE International Conference on Networks (ICON'06)*, Singapore, 2006, pp. 1-5.
12. T. R. Gopalakrishnan Nair and P. Jayarekha, “A strategy to enable prefix of multicast VoD through dynamic buffer allocation,” *International Journal of Computer Science Issues (IJCSI)*, vol. 7, no. 1, pp. 42-48, 2010.
13. B. Chang, L. Dai, Y. Cui, and Y. Xue, “On feasibility of P2P on-demand streaming via empirical VoD user behavior analysis,” in *Proceedings of 28th International Conference on Distributed Computing Systems Workshops (ICDCS'08)*, Beijing, China, 2008, pp. 7-11.
14. F. T. Johnsen, T. Hafsoe, and C. Griwodz, “Analysis of server workload and client interactions in a news-on-demand streaming system,” in *Proceedings of 8th IEEE International Symposium on Multimedia (ISM'06)*, San Diego, CA, 2006, pp. 724-727.
15. L. Cherkasova and M. Gupta, “Analysis of enterprise media server workloads: access patterns, locality, content evolution, and rates of change,” *IEEE/ACM Transactions on Networking,* vol. 12, no. 5, pp. 781-794, 2004.
16. Y. W. Wong and J. Y. B. Lee, “Recursive patching: an efficient technique for multicast video streaming, databases and information systems integration,” in *Proceedings of 5th International Conference on Enterprise Information Systems (ICEIS)*, Angers, France, 2003, pp. 306-312.
17. T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed. Cambridge, MA: MIT Press, pp. 370-384, 2002.
18. W. Tang, Y. Fu, L. Cherkasova, and A. Vahdat, “MediSyn: a synthetic streaming media service workload generator,” in *Proceedings of the 13th international workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV’03),* Monterey, CA, 2003, pp. 12-21.
19. G. Gursun, M. Crovella, and I. Matta, “Describing and forecasting video access patterns,” in *Proceedings of IEEE INFOCOM*, Shanghai, China, 2011, pp. 16-20.
20. Echo state networks, http://reservoir-computing.org/.
21. T. Mezzano, “Echo state networks application on maze problems,” M.Sc. thesis, Katholieke Universiteit Leuven, Faculty of Engineering Science, Leuven, Belgium, 2007.
22. T. Wu, M. Timmers, D. D. Vleeschauwer, and W. V. Leekwijck, “On the use of reservoir computing in popularity prediction,” in *Proceedings of 2010 2nd International Conference on Evolving Internet (INTERNET),* Valencia, Spain, 2010, pp. 19-24.

Dinkisa Aga Bulti

Dinkisa Aga Bulti has been working as a lecturer in the Department of Electrical and Computer Engineering, Wolkite University, Ethiopia since 2011. He also served as Dean of College of Computing and Informatics, Wolkite University. Dinkisa received his M.Sc. Degree in Computer Engineering and B.Sc. Degree in Electrical and Computer Engineering from Addis Ababa Institute of Technology in 2011 and 2009, respectively. His research interest includes: Multimedia processing and communications, Broadband Wireless communication systems, Network Planning and Optimizations, Wireless Sensor Networks, Big-data analysis, etc.

Kumudha Raimond

Kumudha Raimond is research oriented who earned her Ph.D. from Indian Institute of Technology (IIT) Madras, India. Currently she is Director of School of Computer Science and Technology, Karunya University, Coimbatore. Her research focus is on the development of efficient models using hybrid intelligent techniques for various applications in the area of biometrics, biomedical, bioinformatics, compression, watermarking, etc. Her further areas of interest are Big Data Analytics, Watermarking, Wireless Sensor Networks, and Compression and Image Retrieval. She has a good number of research publications in peer reviewed national and international journals, proceedings of international conferences and book chapters to her credit. Besides having 16 years of teaching experience, she also has 3 years of MNC experience at John F. Welch Technology Research Centre, a Research wing of General Electric (GE), Bangalore. She worked as an Energy System Analyst in the Remote Monitoring and Diagnostic Lab of GE, was involved in analysing and predicting the status of remote machines such as steam turbines and locomotives and was awarded the Hats-Off Award (GE, 2002) for her contribution to that research field.