Peer-to-Peer Netw. Appl. (2017) 10:156–169
DOI 10.1007/s12083-015-0414-3

Popularity-based caching for IPTV services over P2P networks

Sajal K. Das[1] · **Zohar Naor**[2] · **Mayank Raj**[1]

Received: 23 February 2014 / Accepted: 30 September 2015 / Published online: 29 October 2015

Abstract This study suggests to use popularity based caching for IP-based TV (IPTV) services over peer-to-peer (P2P) networks. Each peer in a P2P network can use two levels of cache hierarchy: an internal cache and a neighboring peer cache . Using this property, our main focus is on caching the globally most popular video files nearby the clients, in order to reduce the IPTV service delay, increase the quality of service provided to the clients, and reduce the traffic over the Internet backbone. The proposed framework was applied on real data traces from live P2P networks. The results demonstrate a significant improvement over the Least Recently Used (LRU) and the Least Frequently Used (LFU) cache management schemes. This study is motivated by the vision of large P2P networks consisting of many volunteers serving as peers, each of which has a relatively small cache size, in terms of the number of video items it can store. Since the performance of both the LRU and LFU schemes is very poor for small cache, there is a need for another cache management scheme, which outperforms these schemes, especially for small cache size. The proposed distributed popularity-based caching scheme can significantly increase the performance of P2P networks used for video streaming, with respect to the existing networks, that use the LRU or LFU schemes. The performance metric used for comparison is the cache hit ratio and the expected delay for content delivery. In both parameters a significant improvement is demonstrated.

Keywords IPTV · Content distribution networks · Video streaming · Caching · P2P networks

1 Introduction

IP-Based TV (IPTV) is an exciting emerging research field that have potential for a significant impact on our daily lives in the modern world and it may in fact change the way we watch TV. However, IPTV services also bring in challenges. For example, owing to the huge bandwidth consumed by such services, it is extremely important to reduce the traffic generated over the Internet backbone. Further more, due to the huge memory size required to store a typical video file such as a movie, or a TV show, and given the large number of video items, it is not practical to store all the video files nearby their potential clients. Therefore, there is a demand for identifying the most popular items to be stored locally near the potential clients. There is also a strong evidence that most of the demands for IPTV services (about 60–70 %) are for relatively very few items [19]. Thus, identifying these popular items can significantly increase the quality of service (QoS) provided to the clients, as well as reduce the overall bandwidth consumed by this service.

✉ Zohar Naor
zohar@math.haifa.ac.il

Sajal K. Das
sdas@mst.edu

Mayank Raj
mayank.raj@mavs.uta.edu

[1] Department of Computer Science, Missouri University of Science and Technology, Rolla, MO, USA

[2] Department of Mathematics, Physics, and Computer Science, University of Haifa, Haifa, Israel

Basically, there are three representative approaches to the problem of IPTV services within the general context of content distribution: A) usage of a dedicated content delivery network (http://www.akamai.com); B) utilization of existing proxies to cache media data [9, 35]; and C) usage of peer-to-peer (P2P) overlay networks whose goal is content distribution [17, 20, 31, 33].

The usage of P2P networks for IPTV services is expected to have tremendous growth over the next years [20, 31, 33]. Since a typical node in a P2P overlay Content Delivery Network (CDN) has a relatively small cache (in comparison with the typical size of a video movie), the efficient utilization of this cache is crucial for the QoS provided by the CDN. The usage of caching for content distribution has extensively been addressed in the literature (for example, [6, 9, 24]). However, the focus of these studies has mostly been on the issue of content placement algorithm. In other words, they concentrate on where to place the content within the network hierarchy instead of how to select the most popular files to be cached, which has received relatively less attention. In fact, caches are still managed by the least-recently-used (LRU) heuristic algorithm. This motivates us to explore popularity-based caching for IPTV services over P2P networks.

1.1 Background and related work

In the context of cache management, an optimal algorithm for replacing the cache content is to select the cache item that will not be used for the longest time as the first candidate to be removed [29]. Since this algorithm must use knowledge of the future, it cannot be implemented on-line. In practice, the most popular cache replacement algorithm is the heuristic known as the least-recently-used (LRU) strategy [29], that selects the item that was not used for the longest time period to be removed from the cache. This strategy has many variants, such as the LRU-K [26] that selects the item having the longest time period for the last K arrivals, where k is a parameter, and the Least Frequently Used (LFU) strategy [29]. Another method is First in First Out (FIFO) [29] that ignores the usage of each item but considers only the first time it was used.

Live networks using P2P based architecture for delivering IPTV and Video-on-Demand (VoD) services were studied and presented in the literature. See for example, $PPLIVE$ (http://www.pplive.com), $PPStream$ (http://www.ppstream.com), and $Coolstreaming$ [38]. P2P traffic is repetitive in nature and responds well to caching [22]. An analysis of the P2P traffic has revealed that the majority of requests occur for a small number of content [28] and much higher hit ratio can be achieved by the cache replacement algorithms, if they take into consideration the popularity of the content [2, 5, 30].

Various cache replacement algorithms for video streaming have been proposed. However, the solutions proposed in the literature adopted traditional simple greedy cache replacement algorithms, namely least frequently used (LFU) [2], least recently used (LRU) [2] and their variants, like LRU-K [15], and later evolved into more complex techniques, which utilize collaboration among peers [7, 15], partial caching of video content for maximum utilization of cache storage space [23] and ranking of content based on their demand and availability [13, 36]. However, analysis of the P2P traffic has revealed that most of the requests occur for a small number of content [28] and much higher hit ratio can be achieved by the cache replacement algorithms, if they take into consideration the popularity of the content [2, 5, 30]. Even though the authors in [5, 6, 12] consider the popularity of the video content in designing their cache replacement algorithm, the popularity was considered as a **static** parameter. In realty, the popularity of the content continuously evolves with time and its determination is a challenging task. A caching algorithm, which takes into consideration the variation of the content popularity over time can significantly enhance the cache hit ratio and reduce the network load [2, 12, 34]. In this paper, we design a low complexity cache replacement algorithm, which takes into consideration the current and long-term historical demand of the content to determine the popularity of the content. Through analysis and simulations, we show that the proposed mechanism can result in significant improvement of network performance.

Partial results of this study were presented in IEEE Globecom 2012 conference [11]. After the presentation of [11], the idea of using popularity-based caching was suggested in [4, 10, 27]. However, these studies consider content-centric networks. The focus in this study is to use distributed popularity-based caching for P2P networks over conventional networks. Caching the most popular content (MPC) was proposed in [4] for content-centric networks. However, the method proposed in [4] has slow convergence of hitting rate and unstable hitting rate. Moreover, it uses a huge popularity table. Therefore, it is feasible for routers in content-centric networks, but its implementation for volunteer peers in P2P networks, having small cache size may not be feasible. A fine grained popularity-based caching ($FGPC$), having better performance than [4], was proposed in [27]. However, both schemes are applicable for content-centric networks and require a large cache size and a huge popularity table. Therefore, they are not suitable for network elements having small cache size.

1.2 Our contributions

The method proposed in this study is to use hierarchical popularity-based caching. The most popular items may have

duplicate copies nearby their potential clients, while less popular items may be stored at a local server or a neighboring peer, and rarely requested items can be retrieved from the Internet backbone.

We consider a P2P network consists of many peers which provides IPTV services to its clients. Each peer is used as a local server to its nearby clients and also serves other peers. This server can either cache the video items required by its clients, or retrieve this content either from its nearby peers or from the Internet backbone. Due to the huge memory size of a typical video item, and their large number, it is impractical to cache even most of the video items expected to be required. Moreover, typically each peer uses a relatively small cache size, which can store only a small number of video films. Thus, there is a need to handle a cache management strategy aiming to select the video items which are most likely to be required, and cache these items. In practice, the most popular cache management strategy is the LRU scheme. Popularity-based caching can be applied to any cache, since it does not depend on the network architecture. However, the simulation experiments were conducted on live P2P networks, which are currently the most popular architecture for IPTV services.

The proposed caching method is based on two observations made by previous studies: First, most of the video requests are for a relatively small number of items. For instance, it was shown in [5] that by caching 4 % of the video items a reduction of almost 50 % of the peak load was achieved. The second observation is that on the other hand, the distribution of video requests has a long tail, which means that the rest of the video requests (about 50 % according to [5]) are distributed among many video items. The "long tail" distribution of video content popularity has been established by many studies, see for example [3, 8, 18]. These two observations may suggest that a few most popular video items should be cached close to the clients, and have many copies. These items are selected using our proposed popularity-based caching method. P2P networks can use two levels of cache hierarchy - an internal cache managed by each peer, and using the cache of a nearby peer in case of cache miss. Thus, globally popular content can be retrieved either from the internal cache of the serving peer or from a nearby peer, using our proposed popularity-based caching method. Local caching strategy, such as LRU, or LFU, should handle the items that are not globally popular and are not cached by this scheme. The popularity-based caching should be used for the 4 % that are most popular (see [5]), while the LRU and LFU schemes are more suitable for the other, "long tail" items.

The rest of this paper is organized as follows. The proposed popularity-based cache management scheme is given in Section 2, and analyzed in Section 3. Performance evaluation and simulation results are given in Section 4. Finally, Section 5 offers concluding remarks.

2 Proposed popularity-based cache management scheme

In this section we describe the proposed popularity-based cache management scheme. The goal of the proposed scheme is to accurately predict the demand for each item, in order to cache the most popular items. For this purpose, we use history in order to evaluate the popularity of each item. Then, we compare history to the present in order to predict future demand based on the past. For instance, an item for which the demand is rapidly decreasing should get a smaller value for future demand, while an item for which the demand is rapidly increasing should get a higher value for the expected demand. Items for which the demand is steady are expected to show similar behavior in the future. These goals are achieved by assigning weights to the demands for each video item, such that historical popularity decays with time. This method is based on the observation that the demand for video items is not constant, and the popularity of the requested items may change in time. Real world traces of Video on Demand (VoD) services [19] and a study of the pattern of requests to VoD service and video rental stores [16], indicate that most of the VoD demands are for only for a few items. Thus, FIFO and the LRU and LFU methods or their variants have inherent limitations. For instance, given a buffer capacity of 10 video items, and a popularity distribution for which 80 % of the demand is homogeneously distributed between 10 items while the demand for each one of all the other items is extremely rare, then using popularity-based caching we may store the 10 most popular items in the cache in order to get 80 % hit ratio for a relatively small cache size. On the other hand, using variants of FIFO or LRU would be a poor choice for this pattern of content demands. This is because each popular file has 8 % popularity and the expected number of arrivals for other files between two consecutive arrivals of the same popular file is about 12–13 arrivals (for other files). Thus, using LRU or FIFO cache management strategy, a popular file is likely to be removed from the cache before it is requested again. Since real world traces of VoD services [19] and IPTV services [5] indicate that most of the traffic is for very few files, this illustrative example is realistic. It implies that the proposed method can offer, under certain conditions, a significant gain over the LRU scheme and its variants.

2.1 System model

We consider an overlay CDN over IP network. The CDN constitutes on local video servers. Each local video server

is responsible to provide video streaming services to its nearby clients. In order to efficiently achieve this goal, each local video server is located at the gateway between the access network and the WAN. In addition, all the local video servers form a peer-to-peer (P2P) network such that each local video server acts as a peer in the P2P network, that provides services to the other peers in addition to the service it provides to its clients. The system model of the P2P network is illustrated in Fig. 1. Each peer in the P2P network maintains a local cache that stores the most popular video items requested by its clients. In order to accurately evaluate the popularity for each video items, the local server holds a popularity table. The access key to this table is the identification of the video item. There are many encoding methods to identify data items, and this issue is not considered in this work. Each demand for a video item updates the popularity of this items. The popularity of each video item is evaluated by the local video server, using the history of demands for this item. Those video items which get the highest popularity ranks are stored in the local cache. Whenever the local video server receives a request for an item stored in the cache, this request is served directly from the cache. If the video item is not cached, the request is forwarded to the other peers, and if it cannot be served by any peer - the request is forwarded to the relevant content provider. The CDN is responsible to provide video services to its clients. Therefore each client is charged by the CDN, and the CDN is charged by the content providers. Thus, the CDN is charged for the video items stored in a cache by the content providers based on their actual usage, while each end client is charged by the CDN based on its own usage.

2.2 The cache management algorithm

Given a cache size S', our goal is to maximize the cache hit ratio, defined as the probability to provide a requested item from the cache with no need to retrieve the information from the Internet backbone. In order to achieve this goal, we define the *popularity density* ρ by:

$$\rho_i = \frac{p_i}{M_i}. \tag{1}$$

Where M_i and P_i are the memory size and the popularity of the item i, respectively. The value of P_i is derived in Section 3.1. The problem of maximizing the cache hit ratio is NP-hard. We prove it by a reduction from the Knapsack problem, which is known to be NP-hard [14]. Given a Knapsack problem with a Knapsack capacity C and a list of items having a profit d_i and a weight w_i, we associate with each i from the original Knapsack problem a video item with a popularity $p_i = d_i$ and a memory size $M_i = w_i$. The algorithm which maximizes the cache hit ratio must also solve the original Knapsack problem. As opposed to the traditional schemes, such as LRU and LFU, which consider only the content *popularity*, the *popularity density* distinguishes between short video clips and long video films, and considers the video memory size as well as its popularity. Since the problem of optimal cache management strategy is NP-hard, we propose the following approximation algorithm for popularity-based caching:

- Sort the video items in a non-increasing order by their *popularity density* ρ.
- Select the most valuable items until the cache is filled. For this part of the algorithm we can use any of the many approximation algorithms suggested in the literature for the Knapsack problem.
- Whenever there is a need to add a new item i to the cache, it should replace the k items having the least

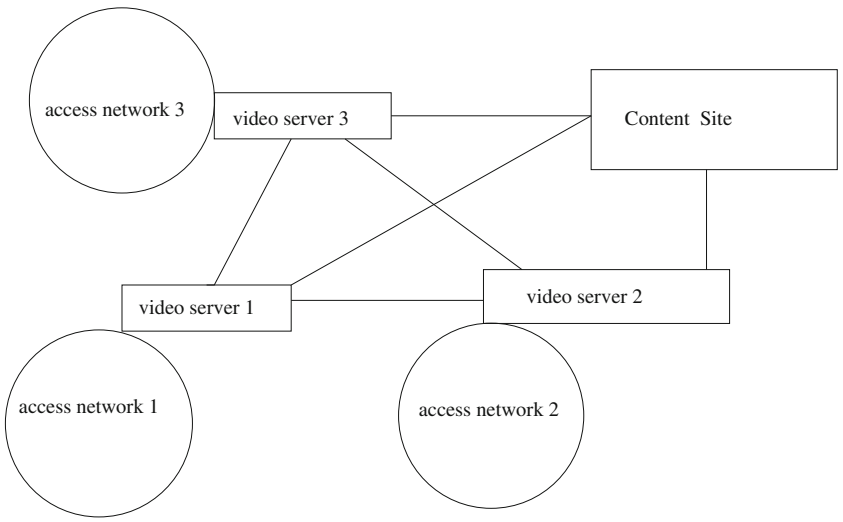

Fig. 1 An overlay P2P network of video servers. Each video server is located at the gateway between the access network and the WAN

popularity density, where k is the minimal integer that satisfies both conditions:

$$\sum_{j=1}^{k} M_j + F \geq M_i. \tag{2}$$

Where F is the size of the free space in the cache.

$$\sum_{j=1}^{k} p_j \leq p_i. \tag{3}$$

Where the items removed from the cache are sorted in a non-decreasing order of the popularity density: $\rho_1 \leq \rho_2 \leq \rho_3 \leq \ldots \leq \rho_k$. A new item j can replace existing items in the cache when both conditions specified in Eqs. 2 and 3 hold.

3 Analysis

In this section we analyze the proposed popularity-based cache management scheme and compare its performance with the performance of the LRU cache management strategy. The goal of the proposed scheme is to accurately predict the demand for each item, in order to cache the most popular items. For this purpose, we use history in order to evaluate the popularity of each item. For instance, an item for which the demand is rapidly decreasing should get a smaller value for future demand, while an item for which the demand is rapidly increasing should get a higher value for the expected demand. Items for which the demand is steady are expected to show similar behavior in the future. This method is based on the observation that the demand for video items is not constant, and the popularity of the requested items may change in time. Real world traces of VOD services [19] and a study of the pattern of requests to VOD service and video rental stores [16], indicate that most of the VOD demands are for only for a few items. Thus, FIFO and the LRU method or their variants have inherent limitations. For instance, given a buffer capacity of 10 video items, and a popularity distribution for which 80 % of the demand is homogeneously distributed between 10 items while the demand for each one of all the other items is extremely rare, then using popularity-based caching we may store the 10 most popular items in the cache in order to get 80 % hit ratio for a relatively small cache size. On the other hand, using variants of FIFO or LRU would be a poor choice for this pattern of content demands. This is because each popular file has 8 % popularity and the expected number of arrivals for other files between two consecutive arrivals of the same popular file is about 12–13 arrivals (for other files). Thus, using LRU or FIFO cache management strategy, a popular file is likely to be removed from the cache before it is requested again. Since real world traces of VOD services [19] indicate that most of the traffic (about 70–80 %) is for very few files, this illustrative example is realistic. It implies that the proposed method can offer, under certain conditions, a significant gain over the LRU scheme and its variants. This gain is analyzed, quantified and validated in Section 3.2.

3.1 Popularity evaluation

In this section we describe the method for evaluating the popularity of each video item, based on the history of demands for this item. This popularity is re-evaluated every pre-defined time period having a length of τ time units. This time interval is denoted as the *measurement interval*. The number of demands for each video item during the *measurement interval*, is stored and used to re-evaluated the updated popularity for each video item. We distinguish between three time intervals - the *long term* time interval reflects the long term history. The *intermediate* time interval reflects the previous *measurement interval*, that is - the last *measurement interval* before the last update of the popularity. The last time interval is the *current* time interval - which is the time period since the last time the popularity was updated. We denote by t_1 the last time the files popularity was updated, and by t_0 the last popularity update time before t_1. Let us consider a video item, say i. The popularity of i at time t, $t \geq t_1 > t_0$, is obtained as follows: the number of demands for i during the time interval $[t_1, t]$ is denoted by $\xi_c(i)$, the number of demands for i during the time interval $[t_0, t_1]$ (the last *measurement interval*) is denoted by $\xi_p(i)$, and the total number of demands for i before t_0 is denoted by $\xi_l(i)$. Every fixed $\tau = t_1 - t_0$ time units, at the end of the *measurement interval*, the popularity of i $\Xi(i)$ is re-evaluated as follows:

$$\Xi(i) = \xi_c(i) + \frac{\xi_p(i)}{2} + \frac{\xi_l(i)}{4}. \tag{4}$$

Where $\xi_c(i)$ reflects the current demand for i, $\xi_p(i)$ reflects the demand for i during the previous measurement interval, and $\xi_l(i)$ reflects the long term history demand for i. Immediately after re-evaluating $\Xi(i)$ using Eq. 4, the values of $\xi_c(i)$, $\xi_p(i)$, and $\xi_l(i)$ are updated in the following order:

$$\xi_l(i) = \xi_l(i) + \frac{\xi_(p)(i)}{2}. \tag{5}$$

$$\xi_(p)(i) = \xi_c(i). \tag{6}$$

$$\xi_(c)(i) = 0. \tag{7}$$

3.2 Comparing popularity-based caching to LRU

In this section we compare the proposed cache management scheme to the LRU scheme. For the sake of simplicity, it is assumed that all content items have the same memory size. The size of the cache, in terms of maximum number of items that can be stored in the cache, is denoted by S'. Let us consider an item i having an arrival probability of p_i. Holding the item i permanently in the cache we get, for the item i alone, a hit ratio of p_i. Thus, the hit ratio contribution of i to the performance of the popularity-based strategy is given by:

$$h_{pop}(i) = p_i. \tag{8}$$

On the other hand, using LRU, the condition under which the item i should be removed from the cache is that during S arrivals the item i was not required, where $S \geq S'$. Since the arrivals for the same content i are independent of each other, the probability that i is not required for S sequential arrivals is given by

$$P_{lru}(i) = (1 - p_i)^S. \tag{9}$$

Hence, using the LRU strategy, a necessary but not sufficient condition under which the item i must be removed from the cache is that i was not requested for at least S' arrivals. The probability for this condition is given by

$$Pr_{lru}[remove] < (1 - p_i)^{S'}. \tag{10}$$

The reason for the inequality in Eq. 10 is the existence of sequences in which i is not requested during S' arrivals, but yet i is not removed from the cache. For instance, if the next arrival after i, say j, is requested again and again for the next $S' - 1$ arrivals and $S' > 1$, then i is not removed from the cache, even though the condition specified in Eq. 9 holds.

It follows from Eq. 9 that the contribution of the item i to the hit ratio of an LRU cache is the probability that: A) there is an arrival of i, with probability p_i, AND B)we have at least another arrival of i during the next k arrivals (i.e. after the first, initial arrival of i). The number $S' \geq k \geq 1$ must be sufficiently small such that i is not removed from the cache before its next arrival. This contribution of i to the hit ratio of LRU cache is given by:

$$\phi_i = p_i[1 - (1 - p_i)^k], 1 \leq k \leq S'. \tag{11}$$

Hence, a sufficient condition under which the popularity-based strategy outperforms the LRU strategy is that for each item i, where i is one of the S' - most popular items, we have:

$$p_i > \phi_i = p_i[1 - (1 - p_i)^k]. \tag{12}$$

It follows from Eq. 12 that the condition under which the popularity-based strategy outperforms the LRU strategy is given by:

$$1 > 1 - (1 - p_i)^k. \tag{13}$$

Since $1 > p_i > 0$, Inequality Eq. 13 always holds. Note that if $p_i = 1$, we have equality since i is never removed from the cache. Applying the above analysis for the S' most popular items, we find that it is better to hold each sufficiently popular item permanently in the cache than using LRU. Since the analysis is for the S' most popular items, no item in the cache can be replaced by another, more popular item.

In order to evaluate the performance improvement of our proposed scheme over the LRU scheme, we first identify the condition under which the popularity-based scheme outperforms LRU. The condition under which it is better to store a content item i permanently in the cache rather than

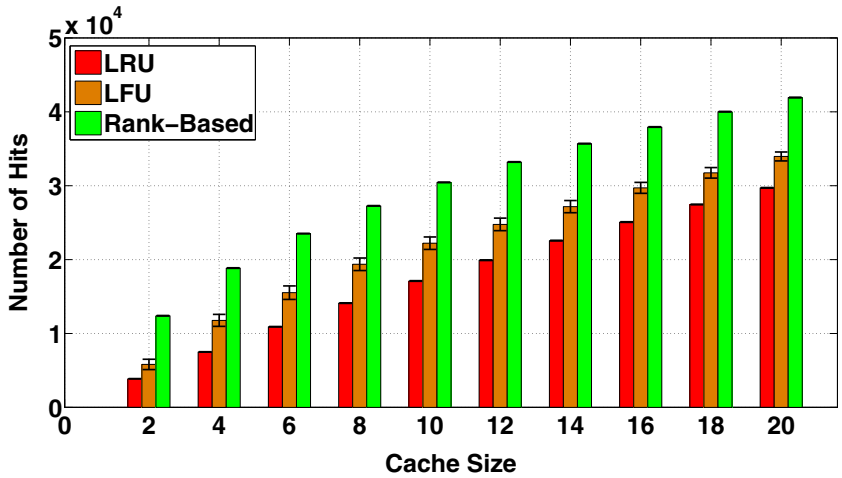

Fig. 2 Comparison of number of hits of popularity-based and LRU and LFU schemes with varying cache size

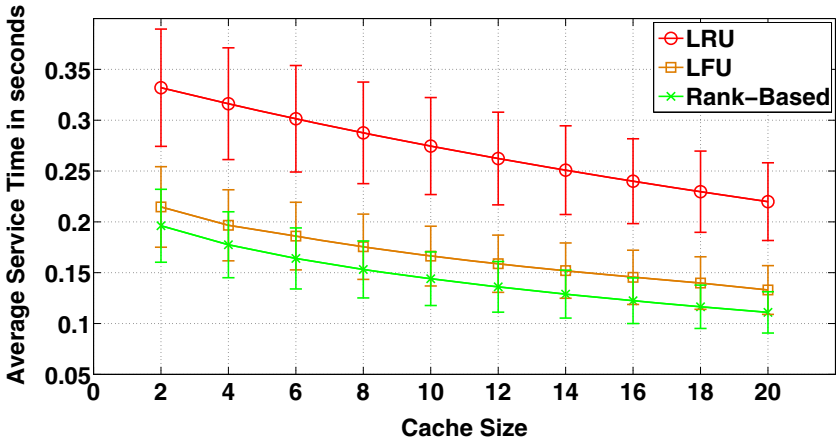

Fig. 3 Comparison of average service time of popularity-based and LRU and LFU schemes with varying cache size

using LRU is that the content item i (already stored in the cache) does not arrive for at least S' sequential arrivals, where S' is the cache size, in terms of the number of content items that can be stored in the cache (for the sake of simplicity, it is assumed that all content items have the same memory size), and then, after removing item i from the cache according to the LRU scheme, i is requested again. The probability for this situation is given by:

$$\Delta = \sum_{k=S'}^{\infty} (1-p_i)^k p_i = p_i \sum_{k=S'}^{\infty} (1-p_i)^k. \tag{14}$$

Therefore, it follows that

$$\Delta = p_i \left[\sum_{k=1}^{\infty} (1-p_i)^k - \sum_{k=1}^{S'-1} (1-p_i)^k \right]. \tag{15}$$

Which finally yields:

$$\Delta = p_i \left[\frac{1}{p_i} - \frac{1-(1-p_i)^{S'-1}}{p_i} \right] = (1-p_i)^{S'-1}. \tag{16}$$

Equation 16 implies that the performance improvement of the popularity-based scheme over the LRU scheme decreases with the cache size S' and with the arrival probability p_i of the content item i. The reason for this behavior is that for large cache size, very popular items should be very rarely removed from the cache. For instance, using Eq. 10 we get that for $p_i = 0.5$ and cache size $S' = 5$ the probability of removing i from the cache is less than $2^{-5} = \frac{1}{32}$. However, the content items provided by IPTV services are typically scattered among many video files. The most popular files consume most of the client demands, but the popularity of each one of the most popular files is not sufficiently high, in terms of the condition specified in Eq. 16. For instance, given cache size $S' = 10$ and $p_i = 0.05$, it follows from Eq. 16 that the performance improvement of the popularity-based scheme over the LRU scheme is $\Delta = 0.95^9 = 0.63$. Given a cache size of 10 items where the popularity of each one of the 10 most popular files is $p_i = 0.05$ for each file, a popularity-based scheme can offer 50 % hit ratio. which is much better than the expected hit ratio offered by the LRU scheme.

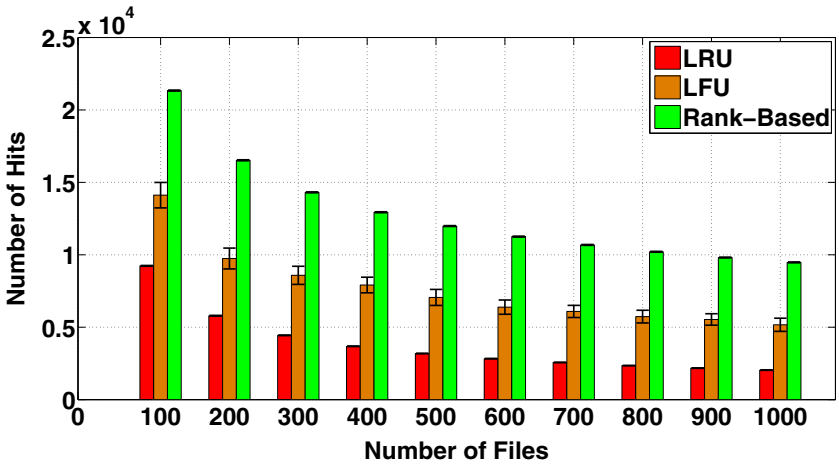

Fig. 4 Comparison of number of hits of popularity-based and LRU and LFU schemes with varying network load

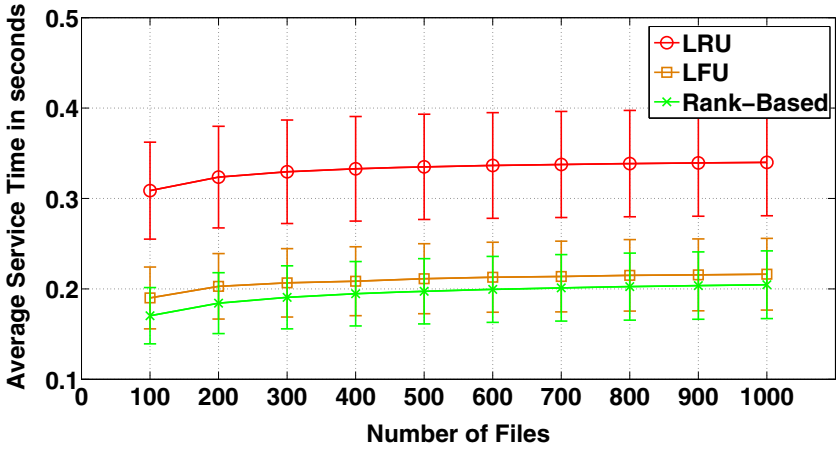

Fig. 5 Comparison of average service time of popularity-based and LRU and LFU schemes with varying network load

4 Performance evaluation and simulation results

We carried out simulation experiments to evaluate the performance of the proposed popularity-based caching scheme in comparison with the least recently used (LRU) and the least frequently used (LFU) caching schemes. We considered a VoD application scenario, wherein the video files from the service provider are distributed across multiple servers based on the Chord [32] protocol. Chord is a distributed P2P protocol which provides scalability, load-balancing and ensures availability of shared resources under dynamic network conditions. In the Chord protocol, each video file is assigned a key and is stored at the server responsible for the key. On arrival of a request for a file, a look is performed for the server responsible for storing the video file using the Chord protocol. On successful identification of the server responsible for storing the video file, the request is forwarded to the server. Upon receiving the request, the server starts streaming the video directly to the client. To reduce the service time, i.e. the time between the request for a video arrives at a server and the server begins streaming the video, the server caches additional video files locally. Thus, on arrival of a request the server first checks for the availability of video files in its cache before performing a look up in the network.

In the following discussion, we outline the general configuration and framework of the simulation platform used. We modified the Oversim P2P simulation framework [1] based on OMNET++ Network Simulator [25] to simulate the proposed popularity-based architecture and caching scheme. The network is assumed to be constituted of 100 servers which store and stream the video files. The arrival of peers is simulated based on the traces of real-world data collected from P2P file sharing networks (http://p2pta.ewi.tudelft.nl/datasets/t505small, http://p2pta.ewi.tudelft.nl/datasets/t1103) [39, 40]. Based on the duration for which the traces have been collected, the simulation time was configured and kept constant at 3553.38 h. On arrival of a peer, the video file to be requested by the peer is determined based on $Zipf$ -like distribution [19, 37], i.e. the probability of choosing the video file i is given as

$$P_i = \frac{1}{k^Z \sum_{j=1}^{N} \frac{1}{j^Z}}. \tag{17}$$

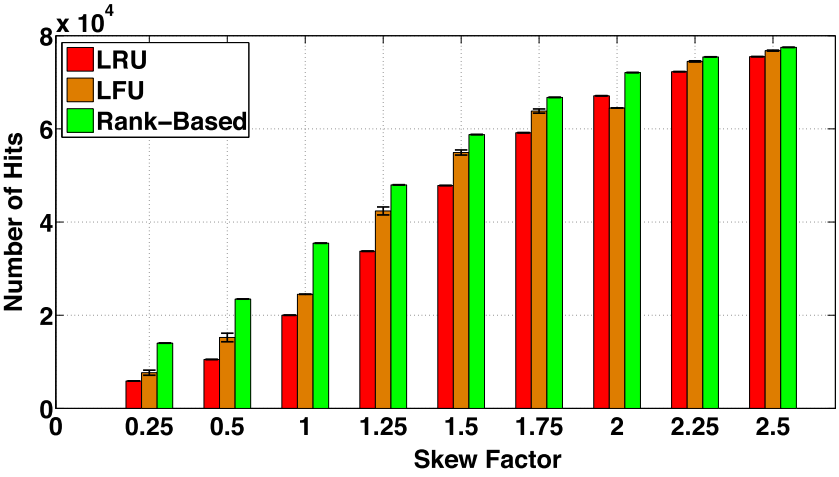

Fig. 6 Comparison of number of hits of popularity-based and LRU and LFU schemes with varying skew factor

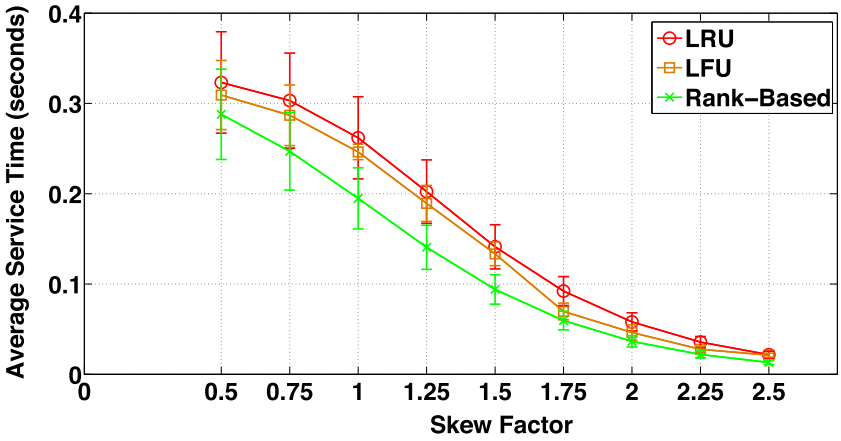

Fig. 7 Comparison of average service time of popularity-based and LRU and LFU schemes with varying skew factor

Where N is the total number of video files in the system, and Z is the *skew factor* and k is the rank of the video file i. Based on real world traces of VoD services [19], and a study of the pattern of file requests in VoD services and video rental stores [16], the value of the *skew factor* Z was set to 0.7 [21]. We assumed the total number of files in the system to be constant at 100.

We measured the performance of the three caching schemes as a function of the service time as well as the total number of hits each caching scheme provides. Hits are defined as when a video file request arrives at a server and is found stored locally in the cache, hence, obviating the need for file look up in the network. Thus, higher the number of hits, the more efficient the caching scheme is and it will have lower service time. For popularity-based scheme, we assume the rank of a video file is known and assigned randomly to the files, with the highest rank being assigned to the most popular video. We evaluated the performance of the three schemes under different network conditions based on the parameters of cache size, network load and popularity of files. For each data point in the results, i.e., the total number of hits and service time of each caching scheme, we report the average and confidence interval using data collected at each of the 100 servers.

1) Cache size: We conducted simulations to study the impact of cache size on the three caching scheme. We define the parameter cache size as the average size of the input files, multiplied by the number of video files that can be stored in the cache on the server. The skew factor and the total number of files in the network were kept constant for the duration of simulations at $Z = 0.7$ and $N = 100$ respectively. The cache size was varied from 2 to 20.

Figures 2 and 3 show the variation of the number of hits and average service time for the files with varying cache size. In the LRU and LFU schemes, as the cache size is increased, the probability that a file will be requested after it has been removed from the cache decreases, as more popular items are rarely removed. But as the LRU and LFU schemes do not take into consideration the overall popularity of the file over time, they may still replace a popular file in the cache with a less popular one based on local history. Thus, resulting in a cache miss in the future. Thus, while the

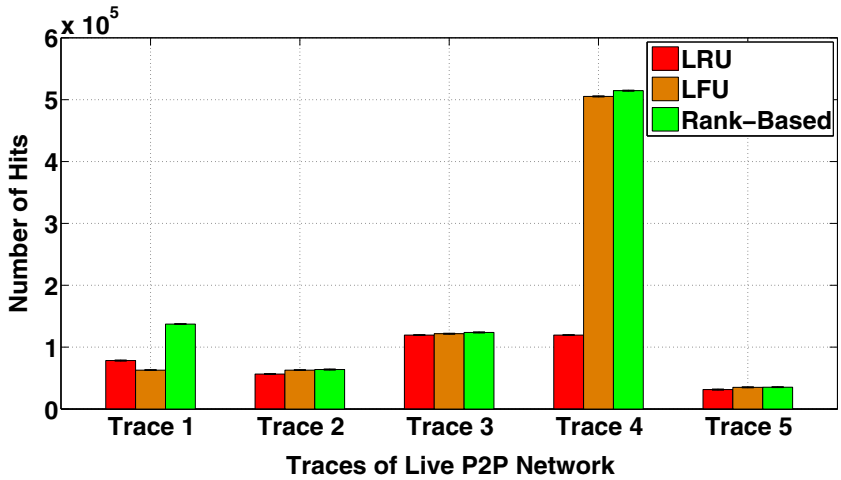

Fig. 8 Comparison of number of hits of popularity-based and LRU and LFU schemes using real-world data simulation

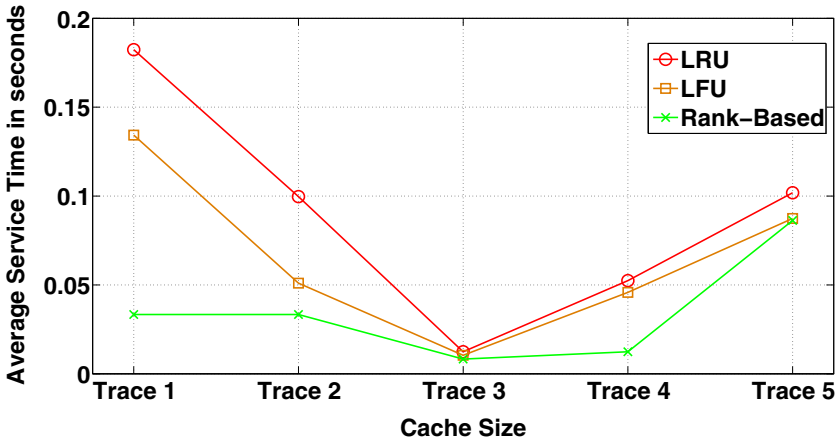

Fig. 9 Comparison of average service time of popularity-based and LRU and LFU schemes using real-world data simulation

performance of both the LRU and the LFU schemes improve with increase in cache size, popularity-based scheme yields better performance even for larger cache. The performance of the popularity-based caching increases with the total popularity of the S' most popular files, where S' is the cache size.

2) Network load We conducted experiments to study the impact of the network load on the three caching schemes. We define network load as the total number of files deployed by the service provider. The skew factor and cache size were both kept constant for the duration of simulations at $Z = 0.7$ and 5 respectively. The number of files was varied from 100 to 1000.

With constant skew factor, as the network load increases the probability of arrival of a file decreases. In the LRU and LFU schemes, the probability that a video file will be requested after it is removed from the cache increases, as more popular files are replaced more frequently. Thus, the performance of both the LRU and LFU schemes degrade, i.e., lower number of hits and higher service time, with increase in network load. Similarly, for the popularity-based caching scheme, as the network load increases, the total popularity of the files in the cache decreases. However, the cache still holds the most popular files constituting of majority of the requests. Thus, it delivers higher number of hits and lower service time than the LRU and LFU schemes, as shown in Figs. 4 and 5.

3) File popularity We studied the impact of varying the degree of file popularity on the three caching scheme. As the skew factor increases, the frequency of requests for the most popular files increases. The number of files in the network and cache size of each server were kept constant for the duration of simulations at $N = 100$ and $S' = 5$, respectively. The skew factor is varied from 0.5 to 2.5.

As shown in Figs. 6 and 7, for LRU and LFU schemes, with increase in the skew factor, the frequency of requests for the popular files increases and the probability of them being removed from the cache decreases. Hence the performance of both the LRU and LFU schemes approaches

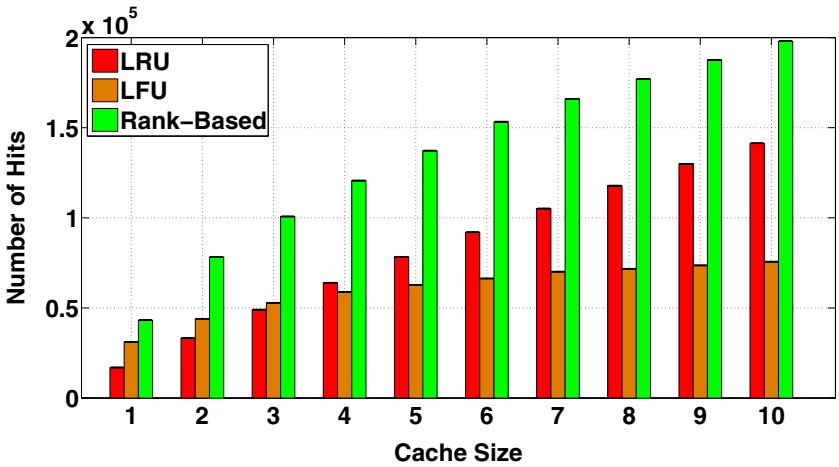

Fig. 10 Comparison of number of hits of the popularity-based and the LRU and LFU schemes with varying cache size in trace 1

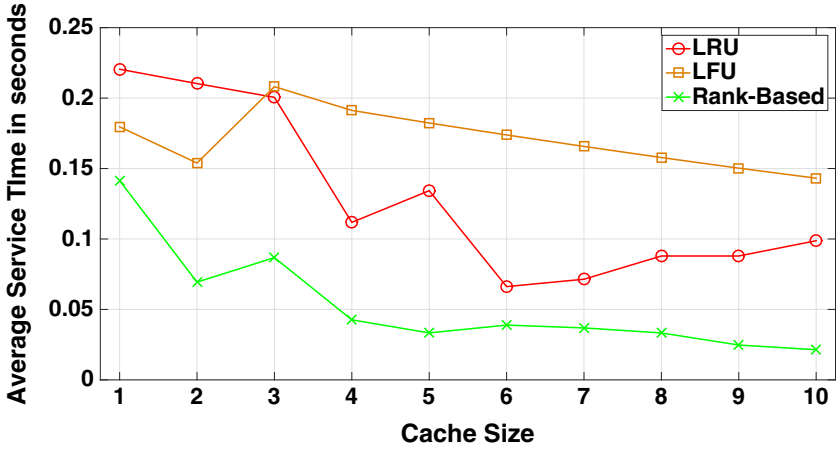

Fig. 11 Comparison of average service time of the popularity-based and the LRU and LFU schemes with varying cache size in trace 1

that of Popularity-Based scheme as the most popular files get replaced less frequently. But, since the popularity-based scheme always cache the most popular files, it yields higher hit ratio and lower service time than the LRU and the LFU schemes.

4.1 Performance comparison using real-world data

To further validate our claim, we used traces (http://p2pta.ewi.tudelft.nl/datasets/t505small, http://p2pta.ewi.tudelft.nl/datasets/t1103) [39, 40] captured from live P2P networks to simulate the arrival of peers as well as the file requests. The network is assumed to be constituted of 100 servers, each one having a cache size of the average size of the input files, multiplied by 5. The traces used have been captured from different P2P communities. Trace 1 constitutes of data collected from P2P community distributing recorded events consisting of 50 files; similarly, Trace 2,3 and 4 respectively constitute of data collected from P2P file sharing community consisting of 127, 16 and 32 files; while Trace 5 constitutes of data collected from P2P community distributing gaming demos with 20 files. The duration of the simulation was kept the same as the duration for which the traces were collected, i.e., at 524.58, 3553.38, 2552.03, 3551.78 and 2518.03 h for Traces 1,2,3,4 and 5 respectively. The popularity of files vary across the traces with the 5 most popular files constituting 36.87, 79.29, 97.16, 72.72 and 62.73 % of the total requests in Trace 1,2,3,4 and 5 respectively. The total number of hits and average service time observed for each trace file are depicted in Figs. 8 and 9.

Based on earlier discussion, since the top 5 files constitute the majority of the requests, the popularity-based cache delivers equivalent amount of hits as the requests. On the other hand, using the LRU and LFU caching schemes, the contents of the cache get replaced based on local history even though the file being replaced may be more popular than the file replacing it, thus resulting in more cache miss than the popularity-based scheme. Hence, the LRU and LFU schemes deliver a higher service time than the popularity-based caching scheme. As the popularity of top 5 files increases, the probability of the files remaining in the cache also increases and thus, the popularity-based scheme offers comparatively less improvement with respect to the LRU and LFU schemes.

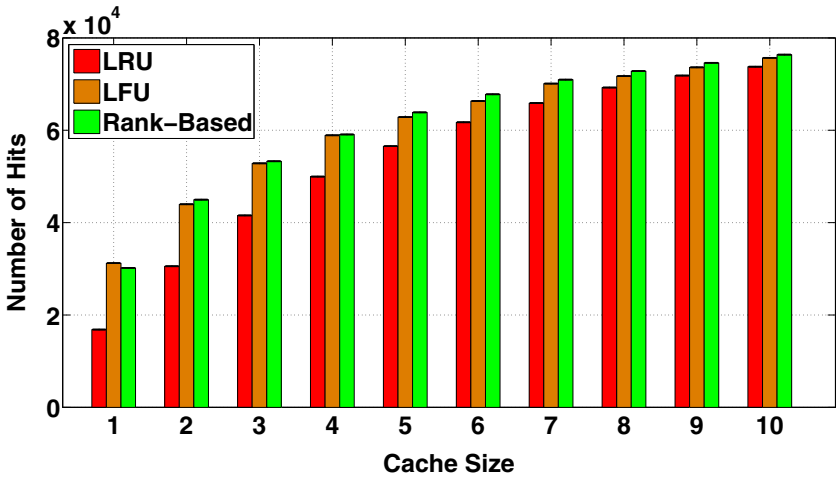

Fig. 12 Comparison of number of hits of the popularity-based and the LRU and LFU schemes with varying cache size in trace 2

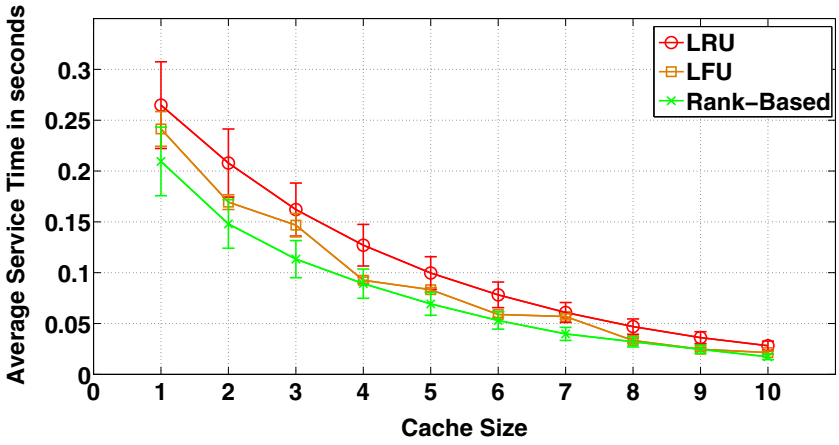

Fig. 13 Comparison of average service time of the popularity-based and the LRU and LFU schemes with varying cache size in trace 2

We further analyzed the effect of cache size on the three caching schemes based on real-world data of P2P network, using Traces 1 and 2, as shown in Figs. 10, 11, 12 and 13 respectively. The number of servers was assumed to be 100 and the simulation time was kept the same as the capture time of Traces 1 and 2, respectively. As the cache size increases, the probability of the file being requested after being replaced in the cache decreases, hence the performance of the LRU and LFU caching schemes improve. Nevertheless, as depicted in Figs. 10, 11, 12, and 13, the popularity-based scheme still outperforms both the LRU and LFU schemes even for large cache size. However, the performance improvement of the popularity-based scheme over the LRU and LFU schemes decreases with the cache size and with the total popularity of the S' most popular files. As depicted in Figs. 10–13, the popularity-based scheme has a higher number of hits and lower service time than the LRU and LFU schemes. This validates our analysis and simulation results discussed earlier and establishes the higher efficiency of the popularity-based caching scheme.

5 Concluding remarks

In this study we propose popularity-based caching for IPTV services over P2P networks. Though our main goal is to apply the method for IPTV services, it can be applied to any content distribution network. The only single feature typical for IPTV that we used in this study is the observation that most of the video requests are for only very few items [19]. Any content distribution network that shares this property can potentially be a good candidate to use a popularity-based caching method. We show by simulation that the popularity-based caching outperforms the LRU and LFU caching in terms of hit ratio and service time. This observation was also verified with real-world data taken from traces of live P2P networks (http://p2pta.ewi.tudelft.nl/datasets/t505small, http://p2pta.ewi.tudelft.nl/datasets/t1103) [39, 40]. The superiority of the proposed method increases as the cache size decreases. This feature is important, since it was shown in [6] that most of the popular information should be cached nearby the clients. These caches are by nature relatively small, thus the incentive to use popularity-based caching at this level of cache hierarchy is strong. In the future we intend to modify our popularity-based caching scheme and to adapt this scheme for other networks than P2P networks, such as cellular networks, and to investigate its performance for these networks.

References

1. Baumgart I, Heep B, Krause S (2007) OverSim: a flexible overlay network simulation framework. In: IEEE global internet symposium, 2007, pp 79–84
2. Abrahamsson H, Bjorkman M (2010) Simulation of IPTV caching strategies. In: SPECTS, pp 187–193
3. Applegate D, Archer A, Gopalakrishnan V, Lee S, Ramakrishnan KK (2010) Optimal content placement for large-scale VoD systems. In: Proceedings of CoNext
4. Bernardini C, Silverston T, Feswtor O (2013) MPC: popularity-based caching strategy for content centric networks. In: Proceeding of IEEE ICC, pp 3619–3623
5. Bjorkman M, Abrahamsson H (2013) Caching for IPTV distribution with time-shift. In: ICNC 2013, pp 916–921
6. Borst S, Gupta V, Walid A (2010) Distributed caching algorithms for content distribution networks. In: Proceedings IEEE INFOCOM
7. Ying C, Zhan C, Wallapak T (2008) Caching collaboration and cache allocation in peer-to-peer video systems. In Journal of Multimedia Tools Appl 37(2):117–134
8. Cha M, Kwak H, Rodriguez P, Ahn Y, Moon S (2007) I tube, you tube, everybody tubes: analyzing the world's largest user generated content video system. In: Proceedings of ACM IMC

9. Chen S, Shen B, Wee S, Zhang X (2003) Adaptive and lazy segmentation based proxy caching for streaming media delivery. In: Proceedings 13th ACM NOSSDAV, pp 22–31
10. Cho K, Lee M, Park K (2012) WAVE: popularity-based and collaborative in-network caching for content-oriented networks. In: Proceeding of IEEE INFOCOM 2012 WKSHPS, pp 316–321
11. Sajal DK, Mayank R, Zohar N (2012) Popularity-based caching for IPTV services. In: IEEE globecom 2012 conference
12. De Vleeschauwer D, Koenraad L (2009) Performance of caching algorithms for IPTV on-demand services. IEEE Trans Broadcast 55(2):491–501
13. Jen-Wen D, Shi-Yuan L (2009) Quality-Adaptive proxy caching for peer-to-peer video streaming using multiple description coding. J Inf Sci Eng 25(3):687–701
14. Garey MR, Johnson DS (1979) Computers and intractability: a guide to the theory of NP-completeness. W H Freeman, New York
15. Shahram G, Shahin S (2007) Greedy cache management techniques for mobile devices. In: Proceedings of IEEE ICDEW, pp 39–48
16. Griwodz C, Bar M, Wolf LC (1997) Long-term movie popularity models in video-on-demnad systems or the life of an on-demand movie. In: Proceedings ACM multimedia, vol 97, pp 349–357
17. Guo L, Chen S, Ren S, Chen X, Jiang S (2004) PROP: a scalable and reliable P2P assisted proxy streaming system. In: Proceedings of the ICDCS
18. Guo L, Tan E, Chen S, Xiao Z, Zhang X (2007) Does Internet media traffic really follow zipf-like distribution? In: Proceedings of ACM SIGMETRIX
19. Huang C, Li J, Rose KW (2007) Can internet video on demand be profitable? In: Proceeding of ACM SIGCOMM
20. Huang Y, Xias Z, Chen Y, Jana R, Rabinovitch M, Wei B (2007) When is P2P technology beneficial to IPTV service? In: Proceedings of the ACM NOSSDAV
21. Hun KA, Cai Y, Sheu S (1998) Patching: a multicast technique for true video-on-demand services. In: Proceeding of ACM multimedia, pp 191–200
22. Leibowitz N, Bergman A, Ben-shaul R (2002) Are file swapping networks cacheable? characterizing p2p traffic. In: Proceedings of the 7th Int. WWW caching workshop
23. Liu F, Shen S, Li B, Li B, Yin H, Li S (2011) Novasky: cinematic-quality vod in a p2p storage cloud. In: IEEE Infocom, pp 936–944
24. Ni J, Tsang DHK (2005) Large-Scale cooperative caching and application-level multicast in multimedia content delivery networks. IEEE Comm Mag 43:98–105
25. OMNET++ Network Simulator, http://www.omnetpp.org/
26. O'Neil E, O'Neil P, Weikum G (1993) The LRU-k page replacement algorithm for database disk buffering. In: Proceedings of international conference on management of data
27. Ong MD, Chen M, Taleb T, Wang X, Leung Victor CM (2014) FGPC: fine-grained popularity-based caching design for content centric networking. In: Proceeding of ACM MSWiM, pp 295–302
28. Qiu T, Ge Z, Lee S, Wang J, Zhao Q, Xu J (2009) Modeling channel popularity dynamics in a large IPTV system. In: SIGMETRICS, pp 275–286
29. Silberschatz A, Peterson J, Galvin P (1992) Operating system concepts. Addison Wesley, Reading
30. Saleh O, Hefeeda M (2006) Modeling and caching of peer-to-peer traffic. In: IEEE ICNP, pp 249–258
31. Silverston T, Fourmaux O (2007) Measuring P2P IPTV systems. In: Proceedings of the ACM NOSSDAV
32. Stoica I, Morris R, Karger D, Kaashoek MF, Balakrishnan H (2001) Chord: a scalable peer-to-peer lookup service for internet applications. In: SIGCOMM, pp 149–160
33. Tan E, Guo L, Chen S, Zhang X (2007) Scap: smart caching in wireless access points to improve P2P streaming. In: Proceeding of IEEE ICDCS
34. Verhoeyen M, De Vleeschauwer D, Robinson D (2008) Content storage architectures for boosted IPTV service. Bell Lab Tech J 13(3):29–43
35. Wu K, Yu PS, Wolf J (2001) Segment based proxy caching of multimedia streams. In: Proceedings of 10th international conference on world wide web, Hong Kong, pp 36–44
36. Ying L, Basu A (2005) pcVOD: internet peer-to-peer video-on-demand with storage caching on peers. In: 11th international conference on distributed multimedia systems DMS, pp 218–223
37. Zipf GK (1949) Human behaviour and the principle of leas-effort. Eddison-Wesley
38. Zhang X, Liu J, Li B, Yum Y-SP (2005) CoolStreaming/DONet: a data-driven overlay network for peer-to-peer live media streaming. In: IEEE INFOCOM, pp 2102–2111
39. Zhang B, Iosup A, Garbacki P, Pouwelse J (2009) A unified format for traces of peer-to-peer systems. In: Proceedings of the 1st ACM workshop on Large-Scale system and application performance, pp 27–34
40. Zhang B, Iosup A, Epems D (2010) The peer-to-peer trace archive: design and comparative trace analysis. Technical Report, Delft University of Technology

Sajal K. Das is the Chair of Computer Science Department and Daniel St. Clair Endowed Chair in Computer Science at the Missouri University of Science and Technology, Rolla, USA. During 2008-2011, he served the US National Science Foundation as a Program Director in the Division of Computer Networks and Systems. His current research interests include theory and practice of wireless and sensor networks, mobile and pervasive computing, cyber-physical systems and smart environments including smart grid and smart healthcare, distributed and cloud computing, security and privacy, biological and social networks, applied graph theory and game theory. He has published more than 600 research articles in high quality journals and refereed conference proceedings, 51 invited book chapters, and coauthored 4 books. He holds 5 US patents and received 10 Best Paper Awards in prestigious conferences such as ACM MobiCom'99, IEEE PerCom'06 and IEEE SmrtGridComm'12. He is also a recipient of numerous awards including the IEEE Computer Society's Technical Achievement Award for pioneering contributions to sensor networks and mobile computing, Lockheed Martin Teaching Excellence Award, and Graduate Dean's Award of Excellence. He is the founding Editor-in-Chief of the Pervasive and Mobile Computing journal, and an Associate Editor of IEEE Transactions on Mobile Computing, ACM Transactions on Sensor Networks, Journal of Parallel and Distributed Computing, and Journal of Peer to Peer Networking and Applications. He co-founded international conferences like IEEE WoWMoM, IEEE PerCom, and ICDCN, and served on numerous conference committees as General Chair, Program Chair, or Program Committee member. He is a IEEE Fellow.

Zohar Naor received the Ph.D. degree in Computer Science from Tel Aviv University, Tel Aviv, Israel, in 2000. Since 2003 he is with the University of Haifa, Israel. His areas of interests include video streaming, P2P networks, wireless networks, resource management of computer networks, mobility management, search strategies, and multiple access protocols

Mayank Raj (mryb8@mst.edu) received his Ph.D. degree in Computer Science from Missouri University of Science and Technology, Rolla, USA in 2015. He holds M.Tech. in Information Technology from International Institute of Information Technology - Bangalore, India (2007) and B.E. in Electronics and Communication from Visvesvaraya Technological University, India (2005). He is currently working on database security as part of IBM Analytics group at Lenexa, USA. Prior to joining IBM, he served as Adjunct Assistant Professor at Computer Science department at Missouri University of Science and Technology. His current research interests include big data analytics, energy efficient computing, wireless sensor network, Internet of Things (IoT), Wireless networks, Peer-to-peer networks (P2P), and cloud computing.