StreamCache: Popularity-based Caching for Adaptive Streaming over Information-Centric Networks

Wenjie Li, Sharief M.A. Oteafy, Hossam S. Hassanein
School of Computing
Queen's University, Kingston, ON, Canada
Email: {liwenjie, oteafy, hossam}@cs.queensu.ca

***Abstract*—The growing demand for video streaming is straining the current Internet, and mandating a novel approach to future Internet paradigms. The advent of Information-Centric Networks (ICN) promises a novel architecture for addressing this exponential growth in data traffic, with ubiquitous caching to facilitate video delivery. In this paper, we present a novel in-network video caching policy in ICN, named *StreamCache*, catering to variable video contents with different sizes and bit rates. Our objective is improving the average throughput of users which consequently enhances the Quality of Experience (QoE), under the heterogeneity of users' devices and network conditions. *StreamCache* is a popularity-based policy, which operates distributively at routers, designed for the online processing in order to narrow the gap between the offline theoretical optimal solution and the real-world application. *StreamCache* operates in rounds, making caching decisions based on video request statistics and minimal cache coordination. We show that, *StreamCache* achieves near-optimal performance compared with the offline benchmark scheme, *DASCache* and outperforms current state-of-the-art protocols, such as *ProbCache* by presenting an elaborate evaluation carried out on ndnSIM, over NS-3.**

***Keywords*—Dynamic adaptive streaming; Content-centric network; In-network caching; Heuristic design; Popularity index.**

I. Introduction

The dynamics of Internet, which have scaled adequately over the past twenty years, are currently faltering under the growth of data demand. According to Cisco's Visual Networking Index (VNI) [1], by 2019, it is projected that the global Internet traffic would surpass 89 thousand petabytes per month. More importantly, among all types of data traffic, Internet video amounts to a considerable percentage: around 80% in conservative estimates [1]. This trend presents challenges for both video content providers (such as YouTube and Netflix) and Internet service providers on how to deliver multimedia to the end users in an efficient way.

This problem led to the standardization of MPEG Dynamic Adaptive Streaming [2] which currently operates on HTTP protocol (DASH). This dynamic approach provides a time-shift control on media requests according to varying network conditions experienced by each user and adapts automatically between versions of each video in order to serve the most suitable bit rate and reduce stalls in the playback.

While in recent years, a rising concern for the scalability and efficiency of the Internet has resulted in the emergence of Information-Centric Networks (ICN) [3]. They are proposed as the next generation of the Internet, designed to intrinsically handle large content and distribute it via network layer primitives. In ICN, the core premise is adapting to content, and catering to both consumers and producers, rather than adopting the client-server approach of the current host-centric Internet. Considering ICN must handle even more amount of multimedia data than the current Internet, to facilitate efficient download, we argue that the dynamic adaptive streaming thus is a native and essential component in the future Internet [4].

As ICN exploits ubiquitous caching as a networking primitive to enhance information delivery, the challenge of catering to video content can be coped with by applying a caching scheme which is aware of the adaptive requests. We therefore argue that one of the key challenges of dynamic adaptive streaming over ICN is to decide which and where versions (bit rates) of video contents should be cached, with the ultimate goal of improving users' Quality of Experience (QoE). The QoE metrics of adaptive streaming are specified by Oyman and Singh's work [5], which include average throughput, representation switches, playout delay, and so on. To address the issue of heterogeneous video cache placement in ICN, our previous work [6] presented an offline benchmark which demonstrated the maximal expected throughput of requested video segments to facilitate rapid streaming over ICN. The benchmark was solved by optimization which was formulated as an NP-Complete problem.

In this paper, we propose a novel video caching policy, *StreamCache*, which presents a near-optimal solution for real-time cache placement decisions. Our contribution lies not only in the near-optimality of this solution, but also in establishing a distributed and low-overhead protocol that scales to heterogeneous scenarios. In each router, *StreamCache* makes local decisions on what video content should be cached based on minimal coordination and aggregated video statistics, avoiding global knowledge of the network as we required in [6]. We show that, under various network settings (*e.g.,* available cache storage and popularity distribution), *StreamCache* enables users to achieve near-optimal average throughput compared with the results in [6] and outperforms current state-of-the-art caching placement policies in the literature.

The remainder of this paper is organized as follows. Section II overviews related work including recent research on caching schemes in ICN and predecessor work on dynamic adaptive streaming. Section III elaborates the design principals of *StreamCache* and explains the steps of our greedy selection algorithm, detailing the approach of calculating the caching utility of video contents. Our experiment setup and performance evaluation results are then presented in Section IV, and we conclude in Section V with our final remarks and propositions for future work in this crucial direction.

II. Related Work

While many recent proposals have presented various ICN architectures, Content-Centric Networks (CCN) proposed by Jacobson *et al.* [7] stand out as one of the prominent architectures. In the realm of CCN, *Interest* and *Data* are two types

978-1-4799-6664-6/16/$31.00 ©2016 IEEE

of packets. As their names imply, *Interest* is dispatched by subscribers (users) for information and *Data* encompasses the requested contents which could be cached anywhere in the network.

There is already a wealth of research related to the caching policy design where those works differ by the level of caching decision coordination and information exchange. Li *et al.* [8] proposes saving as much data traffic as possible based on the popularity of contents. Their approach requires collecting statistics of contents request frequency and caching decision coordination among routers along the entire routing paths. While Li *et al.* [9] requires only neighbourhood coordination, utilizing a hash function to eliminate the caching redundancy. Psaras *et al.* presents *ProbCache* model [10] which assigns each router a caching probability based on its own information and requires no coordination. Our proposed policy reduces the statistics exchange between routers and specify the caching utility compared with [8], particularly for video streaming application and achieves much better performance than [10] with limited cache decision coordination.

The problem of caching for dynamic adaptive bit rates of video contents has also been studied in the domain of CDN, viz the work done by Lee *et al.* [11] and Liu *et al.* [12]. However, these attempts are mainly devoted to the influence of cached contents on the rate adaptation algorithm for smooth switch among bit rates and fail to address the problem of deciding what and where to cache with intrinsic caching capabilities in CCN. Liu *et.al.* [13] studied caching behavior over CCN when dynamic adaptive streaming is applied. Their observation that clients could be served with bit rates even higher than actual bandwidth discloses the important role of caching on improving user's QoE. However, the applied cache decision policies, which tackle a general caching scenario, are not designed for dynamic adaptive streaming. Thus, in order to most benefit the users, it is critical to consider a caching policy particularly for video streaming applications.

III. Distributed Video Caching Solution over Information-Centric Networks

We build our video caching system upon the prominent ICN architecture, CCN, because of the wide support by the research community and tools available for benchmarking and simulation. In this section, we first detail our *StreamCache* system, explaining the network topology and routing protocol we apply. Next, the caching policy design principals and the operation of our *StreamCache* approach in the real-world application will be elaborated.

A. System Description

Our system primarily targets video delivery in CCN. The requested videos by all users are assumed to reside in **one** video server without considering any replica. This assumption is derived from the current settings of CCN architecture because each server would add a unique postfix to the names of contents. Thus, this feature distinguishes each content from all its replicas. The system contains two different types of routers: edge and intermediate routers. All clients are served by edge routers while intermediate routers never connect with clients directly.

We apply the shortest-path routing scheme, generating a single *Interest* forwarding path from each edge router to the video server, which is depicted in Figure 1. Although CCN supports multi-path forwarding which increases the resilience and possibly reduces load at the repository, Rossini *et al.* present experimental results in their study [14] which show that the single-path strategy still achieves the lowest network load and smallest request hop counts compared with other multi-path strategies. Therefore, to reduce the system load which thereby benefits the video downloading, we adhere to CCN's single-path forwarding strategy.

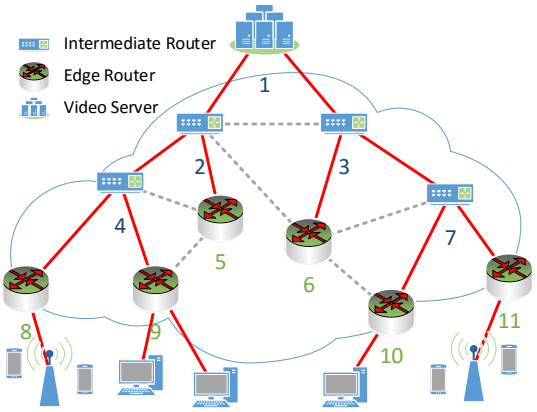


Fig. 1. Network Topology. The solid lines indicate the forwarding paths based on the shortest-path routing scheme.

B. StreamCache *Design principals*

The objective of *StreamCache's* caching policy is to improve users' QoE by maximizing average throughput when they utilize a dynamic rate adaptation control over ICN. To reach this objective, several principals must be followed and met.

First, *StreamCache* is inclined to cache those video segments which are frequently requested. Caching popular video contents would increase the probability of cache hits which ultimately leads to a better cache utilization and higher throughput. Secondly, in order to scale over a large-scale network, *StreamCache* should be able to make distributed caching decisions without relying on a centralized controller to synthesize information from the entire network. Thirdly, as video streaming is a delay-sensitive application which incurs high-volume and continuous data transmission, large amount of coordination signals and information exchange would interfere with the normal video delivery which sometimes may even significantly degrade users' experience. Thus, frequent control information exchange is not desirable for *StreamCache*.

In order to cache videos which are frequently requested, statistics collection is thus necessary. Thanks to the naming scheme of ICN, such statistics is feasible to obtain by recording the 'Name' of each *Interest* packet without the need for parsing the data packet. *StreamCache* then operates in rounds. Each round begins with the statistics collection on edge routers. At the end of each round, *StreamCache* then makes video caching decision based on the collected information and starts the next round. Each ICN/CCN router in our system are only allowed to cache the designated video contents by *StreamCache* which means there is no cache replacement on routers in response to the data flows during the period of each round.

The size of records of video requests could be considerably large after a long run. In order not to generate large amount

of coordination data, these records are stored locally on edge routers and shall never be passed or aggregated with records of other routers over the network directly. Admitting that limited coordination is also necessary to efficiently utilize the cache capacity, at the end of each round, the summarized version of statistics tables are first created by each edge router and delivered along the forwarding path as well as its local caching decision to the parent node; when the parent node receives the summarized statistics and caching decisions from all the children nodes, it then creates its own tables and passes forward. This procedure continues until all routers in the network finish updating their own caching decisions by local *StreamCache* protocol, from the edge routers toward the root of routing topology.

In the remainder of this section, the protocol of constructing the summarized tables and recovering the statistical information are detailed in Section III-C. Section III-D then elaborates the algorithm based on statistical information to make independent caching decision at each router.

C. Deriving Cache Utility

The purpose of calculating cache utility for each video segment is to rank those chunks which have more effect on improving users' average throughput. As analyzed in Section III-B, a more popular video segment shall intuitively have a higher chance to be cached. Therefore, cache utility is correlated with video statistics at each router. We start with modelling user's requests for dynamic adaptive video streaming and then explain how to summarize the statistics tables for cache utility calculation on edge routers, followed by similar processing on intermediate routers.

1) Characteristics of dynamic adaptive streaming: Under dynamic adaptive streaming, each video request record could be abstracted by a tri-tuple: (VideoFile(f), VideoChunk(k), Bitrate(b)), which represents the request for a video segment of kth chunk in the file f, encoded with bitrate b.

The adaptive bit rate (b) contained in the video *Interest* is adjusted according to user's instantaneous link condition and is not correlated with the video contents that the users are requesting for. This provides an opportunity to make statistical analysis on bit rate distribution for users who are under the service of the same edge routers since those users share a similar link condition.

We assume that the process of video requests made by a certain user corresponds to automatically playing a video by the media player or browser, such that the sequence of requests made by the same user for video chunks are generated following the exact video playback. Two consecutive requests for video chunks exist obvious correlation since as to any request for a certain video chunk, we know exactly what request the network is expected to receive (*i.e.,* the next chunk in the playback, if retransmission is not considered). Therefore, to quantify such correlation, we use p to describe the probability for a user to continue watching the next chunk so that the probability of requesting totally k video chunks in a certain file follows the geometric distribution, where $P(X = k) = (1 - p)p^{k-1}$. This modelling corresponds to the characteristics of most people when watch the video that start playing from the beginning; keep watching for a period of time; close the media player when the video gets boring.

2) Cache utility calculation on edge routers: The caching utility of video segments is to represent the importance of caching on improving the average throughput. Let us use $\mathbb{U}$ to denote this cache utility, RTT_{avg} to denote the average round trip delay of a certain video request, and S to denote the size of video segments. Thus, the utility function is defined as

$$\mathbb{U} = PopScore \times \frac{RTT_{avg}}{S} \tag{1}$$

where $PopScore$ represents the popularity score of a certain video segment. As our policy aims at improving users' QoE in terms of average throughput, video requests for those segments which would result in low throughput need to be cached in order to reach a better overall performance. Therefore, the inverse of throughput (RTT_{avg}/S), together with the popularity score, indicate what video segments need to be cached: the higher this utility value, the more importance of caching.

As video *Interest*s are indexed by the tri-tuple (f, k, b), $PopScore$ of a certain video segment, $PopScore(f, k, b)$, has the following relationship

$$PopScore(f, k, b) \propto q(f)p^{k-1}\pi(b) \tag{2}$$

where $q(f)$ and $\pi(b)$ represent the probability of request for video file f and bitrate b respectively. As we described the common behaviour of users watching media contents, p denotes the probability of continuing watching next video chunk. Thus, in order to watch the kth chunks, previous $k-1$ chunks must have already been requested.

In dynamic adaptive streaming, video contents are chopped into segments while each one contains the same length of playback time. Therefore, the size of *Data* is determined only by its encoded bitrate and the average RTT of video packets should be measured based on different bitrates as well. If we consider the ratio of proportional relation in (2) is simply one, the caching utility of video segment indexed by (f, k, b) thus could be written as

$$\mathbb{U}(f, k, b) = \frac{q(f)p^{k-1}\pi(b)RTT_{avg}(b)}{S(b)} \tag{3}$$

As mentioned in Section III-B, edge routers in our system are responsible for recording users' video requests where each entry is indexed by the tri-tuple, (f, k, b), and generating summarized statistics tables for further reference by its parent node. There are two such summarized tables: **Video File** Table and **Bitrate** Table, as shown in Figure 2. The Video File Table shows the times of requests for different video files by counting the record for the first chunk ($k = 1$) regardless of what bitrates are encoded with, while we also count the number of video requests by different bitrates and keep the result in the Bitrate Table.

The purpose of constructing these two summarized tables is to limit the size of coordination information exchanged among routers. Suppose we have a record of video requests, containing 10,000 video files where each one is chopped into 1,000 chunks, encoded with 10 different bitrates and we also need a four-byte storage to keep each count number. The size of this original statistics record would be roughly 400MB which will certainly influence the normal video delivery when this information needs to be exchanged. With our method, the size of Bitrate Table will be only 40Bytes and the size of Video File Table would be roughly 40KB, which are much smaller compared with the original record.

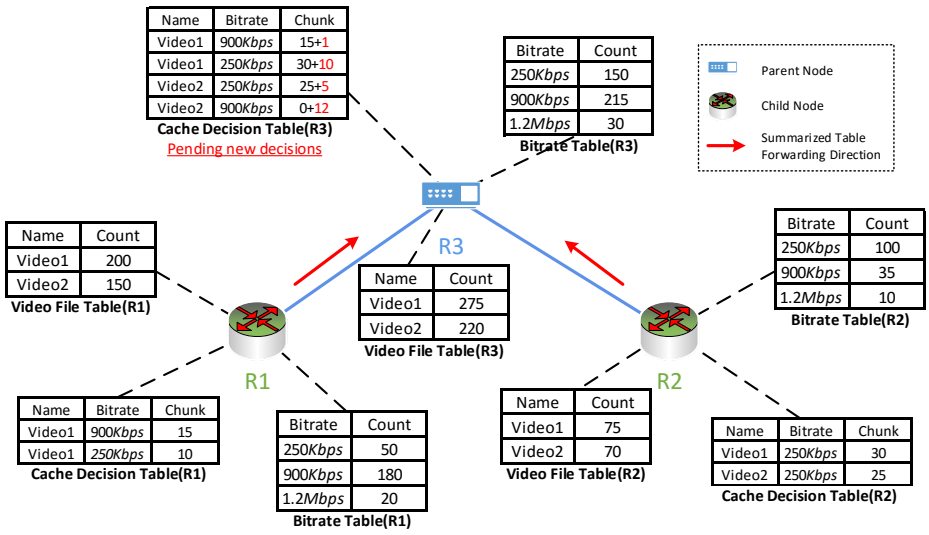


Fig. 2. Summarized Statistics and Cache Decision Table. Information is first gathered from child nodes and then parent node updates its local decision.

Using this two tables, we can achieve the parameters, $q(f)$ and $\pi(b)$ in Equation (3). We do not keep the third table to represent the popularity of different video chunks. Instead, as we claim that the requests for chunks follow a geometric distribution, we apply the maximum likelihood approach to estimate the probability of continuing watching the next video chunk (p). We assume that the size of video chunks encoded with different bitrates ($S(b)$) are known by the system and we also require that each edge router sampling the delay of video requests to get $RTT_{avg}(b)$. Therefore, after achieving all parameters in Equation (3), we can calculate the caching utility on edge routers.

3) Cache utility calculation on Intermediate routers: The utility function for intermediate routers to calculate $\mathbb{U}(f, k, b)$ is the same as Equation (3) for edge routers. However, the problem now comes down to deriving corresponding parameters since intermediate routers own no complete statistical records to calculate the parameters as we do for edge routers. We thus require each intermediate router to collect the Video File and Bitrate Table from its downstream nodes following the *Interest* forwarding path to build its own summarized tables.

As shown in Figure 2, router R3 builds the Video File and Bitrate tables by joining the corresponding tables from R2 and R1. We update the probability of continuing watching the next chunk (p) by simply choose the largest value among its child nodes (*i.e.,* $p_{R3} = \max(p_{R1}, p_{R2})$, in this example). $RTT_{avg}(b)$ for intermediate routers still needs sampling video requests since the queuing delay is dependent on the network load and changes hop-by-hop. It is worthwhile to note that, this utility calculation (for both edge and intermediate) routers should be done prior to running our distributed caching policy in order to provide the up-to-date information to facilitate decision.

D. StreamCache *Greedy Selection Algorithm*

Our *StreamCache* policy relies on the summarized statistics to calculate the caching utility and **Cache Decision** Tables from downstream nodes to make greedy choice.

Observe that, as to a certain video file (f) encoded with bitrate (b), if the α^{th} video chunk is chosen to be cached on a certain router, any video chunk k in the same file, where $1 \leq k < \alpha$, must have already been cached as well. It is because the caching utility for those video chunks is larger than the α^{th} chunk and based on the Equation (3), they should be cached prior to the α^{th} chunk if there is enough cache capacity.

This observation simplifies our Cache Decision table design. As shown in Figure 2, tuple $(Name, Bitrate)$ is the key for the cache Decision table where column named with 'Chunk' records the sequence number to indicate which video chunks have already been cached. For example, the entry of (*Video1*, *900kbps*, 15) in the table of router R1 means that the first 15 chunks of *Video1* encoded with *900kbps* have been cached.

Before running the *StreamCache* policy, the Cache Decision tables from the downstream nodes still need to be aggregated as we did for summarized statistics and then the *StreamCache* policy would update the chunk sequence, by appending numbers in this table which represents the local caching decisions at this parent router (as marked in red colour in Figure 2).

Let us use W to denote the whole video chunks space and $V_j(i)$ to denote the cached video chunks on router i which is the downstream node of router j. In order to cache those video contents which are commonly interesting to users, the available caching space for router j is thus $W - \bigcup_i V_j(i)$. This idea is reflected on the Cache Decision table update rule: the chunk sequence of each entry in the aggregated decision table is the maximum number among tables of downstream nodes. For example, in Figure 2, the decision entry for (*Video1*, *250kbps*) exist both in R1 and R2 which cache the first 10 and 30 chunks respectively. However, the chunk sequence number in corresponding entry of the aggregated table at R3 is 30 which means the R3 would consider caching the same video file starting from the 31^{st} chunk. The detailed *StreamCache* algorithm is listed in Figure 3.

Lines 1-5 calculate the caching utility on router j based on the summarized statistics while lines 6-14 aggregate the caching decision table on router j in order to satisfy the common interests from downstream nodes. Line 15 sorts the

INPUT: $\forall i \in$ Downstream(j), Video File Table (VF_i), Bitrate Table (BR_i), Cache Decision Table (CD_i), Cache Capacity at router j (C_j)
OUTPUT: The cache decision update, ΔCD_j

1: **for** $\forall i \in$ Downstream(j) **do**
2: $VF_j := VF_j \cup VF_i$
3: $BR_j := BR_j \cup BR_i$
4: **end for**
5: $U_j :=$ CalculateUtility(VF_j, BR_j)
6: **for** $\forall i \in$ Downstream(j) **do**
7: **for** $\forall e \in CD_i$ **do**
8: **if** $e \notin CD_j$ **then**
9: Insert e in CD_j
10: **else if** $CD_j(e)$.ChunkSeq $< e$.ChunkSeq **then**
11: $CD_j(e)$.ChunkSeq $= e$.ChunkSeq
12: **end if**
13: **end for**
14: **end for**
15: $U_j :=$ Sort(U_j)
16: **for** $\forall u \in U_j$ **do**
17: **if** $C_j - Size(u.b) \geq 0$ **then**
18: **if** $(u.f, u.b) \notin CD_j \vee u.k > CD_j(u)$.ChunkSeq **then**
19: $\Delta CD_j := \Delta CD_j \cup \{u\}$
20: $C_j = C_j - Size(u.b)$
21: **end if**
22: **else**
23: break
24: **end if**
25: **end for**

Fig. 3. Distributed *StreamCache* Algorithm

caching utility for all video chunks from the largest to smallest and lines 16-25 utilize the greedy selection to fill the cache capacity of router j. The complexity of *StreamCache* algorithm is dominated by *Sort* operation on Line 15, which could be $O(nlogn)$ on average, where n is the number of video chunks considered in the system.

IV. Performance evaluation

In this section, our proposed distributed *StreamCache* policy is evaluated under an environment which mimics the real scenario when dynamic adaptive streaming is applied. We show that *StreamCache* achieves significant improvement over popular caching schemes.

A. Simulation Setup

We build our simulation environment over ns-3 based simulator, ndnSIM [15]. In the simulation, each video file is segmented into chunks with a duration of 2 seconds and we consider four common available bit rates, which are: 250Kbps, 400Kbps, 600Kbps and 900Kbps. We compare the performance of our *StreamCache* policy with the optimal benchmark, *DASCache*, presented in [6] and two other popular cache placement schemes in the literature: *Cache Everything Everywhere (CE2)* [7] and *ProbCache* [10].

As highlighted in Section III-B, *StreamCache* policy operates in rounds. Thus, one test in the simulation contains four rounds where in the first three rounds, routers collect statistics and refresh the cached video contents and we only measure the performance of *StreamCache* on the fourth round. As to *CE2* or *ProbCache* policy, we apply it over the first three rounds (and use LRU approach if replacement is warranted) and disable the cache replacement in the fourth round to measure the performance. As to the optimal *DASCache*, it is an offline approach such that the cached contents are not refreshed in the entire simulation.

TABLE I. Default Simulation Parameters

Parameter	Value
Number of video files	100
Number of video chunks per file	15
Number of Nodes	20
Available bit rates	4
Size of video chunk (KB)	{62.5, 100, 150, 225}
Skewness factor (α)	0.8
Cache capacity percentage (ω)	5%
Cache allocation ratio (ϵ)	1
The probability of continuing watching (p)	0.9
Bandwidth	2Mbps

As *StreamCache* targets optimizing users' QoE, we chose *Average Throughput* of all users as a core performance metric. It is measured between two timestamps when an *Interest* packet is sent and the corresponding *Data* packet arrives at the user's device.

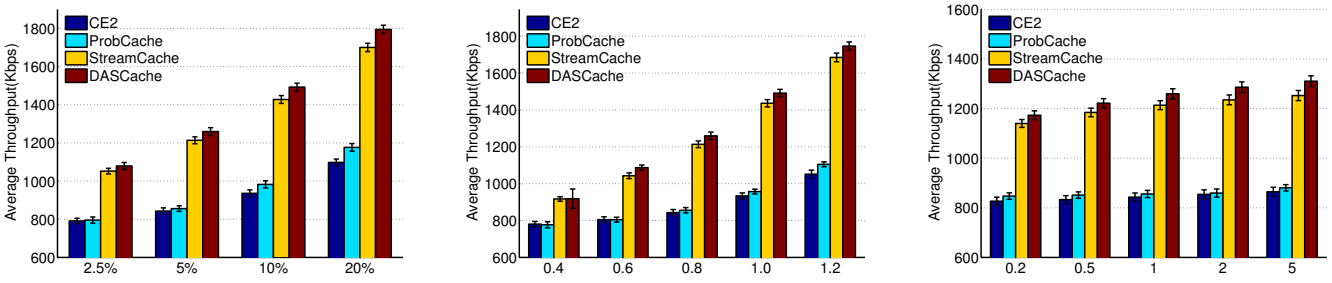
In the simulation, without loss of generality, we use a tree topology of 20 nodes (12 edge nodes, 7 intermediate nodes, 1 root node). The average rate of video requests made by users in the simulation is generated randomly with the only constraint of not exceeding the load that each link bandwidth could support. However, as to which video file should be requested, we simply choose Zipf distribution where the probability of requesting fth file is $q_f = \beta/f^\alpha$, $\beta = (\sum_{f=1}^{F} 1/f^\alpha)^{-1}$. The requests for video chunks in the same file is made in sequence and we control the probability of continuing watching the next chunk, p. The default parameters used in the simulation are listed in Table I.

B. Performance Analysis

We test several parameters which may influence the performance of caching policy, including the cache capacity percentage (ω), the popularity skewness (α) and the cache allocation ratio (ϵ). ω indicates the total amount of cache capacity proportional to the size of all video contents. α controls the shape of Zipf distribution while ϵ is the ratio of cache capacity of edge routers over intermediate routers. All experimental results show confidence intervals at a 90% confidence level.

1) The Impact of Cache Capacity Percentage: To investigate the impact of available cache capacity on the performance of caching policy, Figure 4a shows the results of average throughput with increased amount of cache capacity (while we maintain $\epsilon = 1$). Our *StreamCache* policy outperforms *CE2* and *ProbCache* among all test cases. For example, at $\omega = 10\%$, the average throughput of *StreamCache* is improved by 52.4% and 45.3% over *CE2* and *ProbCache* respectively. The reason is that neither *CE2* nor *ProbCache* is a popularity-based scheme which cannot distinguish between requests from users which are mixed with popular and unpopular contents. However, *StreamCache* records statistics so that it could utilize cache storage more efficiently.

Figure 4a also presents that our proposed policy achieves close performance compared with our previous optimal benchmark. For example, at $\omega = 10\%$, *StreamCache* is only 4.4% worse than the optimal result. Therefore, our designed distributed caching protocol captures the dynamics of video



(a) Cache Capacity Percentage (ω) (b) Popluarity Skewness Factor (α) (c) Cache Allocation Ratio (ϵ)

Fig. 4. Simulation results

caching system and could be applied in the online processing application.

2) The Impact of Popularity Skewness: In the simulation, we suppose that users' request for video files follows the Zipf distribution while we control the parameter α to alter the popularity score for each chunk. Figure 4b presents the results across different skewness values. For example, at $\alpha = 0.8$, *StreamCache* improves by 44.0% and 41.8% over *CE2* and *ProbCache*. When α changes from 0.4 to 1.2, *StreamCache* improve significantly on average throughput. The reason is that, at large α, requests concentrate more on a small set of popular contents which thereby increases the cache utilization. *CE2* and *ProbCache* receive less impact on performance since they are less sensitive to popularity changes.

3) The Impact of Cache Allocation Ratio: In addition to the total amount of cache capacity, the allocation of cache storage among routers will also influence the performance of video caching. As shown in Figure 4c, our *StreamCache* caching policy outperforms *CE2* and *ProbCache* among all test cases. By moving more cache storage closer to users, we can expect that all caching policies achieve better performance in terms of average throughput.

As ϵ changes from 0.2 to 5, the cache storage size on edge routers is around three times larger and the average throughput of *StreamCache* policy improves by 11.7%. However, compared with the result form Figure 4a that *StreamCache* improves by 18.5% as cache capacity is only one times larger (ω from 5% to 10%), result in Figure 4c indicates the importantance of caching on intermediate routers and shows the effect of cache redundancy. Since the total amount of available cache capacity is determined, a large ϵ means more cache storage allocated on edge routers, caching similar video contents which consequently, increase the cache redundancy. This less efficient cache utilization thus offsets the benefit achieved from moving cache closer to the users.

V. Conclusion

In this paper, we proposed a distributed caching policy, *StreamCache*, for dynamic adaptive streaming to facilitate video delivery over ICN. The simulation presents that *StreamCache* improves the QoE of users by achieving close-to-the-optimal average throughput of users under variable network settings. In future work, instead of requiring collecting statistics and responding to the video requests periodically, we intend to design a model which could dynamically adjust the decision on the cached bitrates of video contents, inherently adapting to changing link conditions.

References

[1] Cisco, "Cisco visual networking index: Forecast and methodology, 2014–2019," 2014.

[2] I. Sodagar, "The mpeg-dash standard for multimedia streaming over the internet," *IEEE MultiMedia*, no. 4, pp. 62–67, 2011.

[3] G. Xylomenos, C. N. Ververidis, V. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. V. Katsaros, G. C. Polyzos *et al.*, "A survey of information-centric networking research," *Communications Surveys & Tutorials*, vol. 16, no. 2, pp. 1024–1049, 2014.

[4] S. Lederer, C. Mueller, C. Timmerer, and H. Hellwagner, "Adaptive multimedia streaming in information-centric networks," *Network, IEEE*, vol. 28, no. 6, pp. 91–96, 2014.

[5] O. Oyman and S. Singh, "Quality of experience for http adaptive streaming services," *Communications Magazine, IEEE*, vol. 50, no. 4, pp. 20–27, 2012.

[6] W. Li, S. Oteafy, and H. S. Hassanein, "Dynamic adaptive streaming over popularity-driven caching in information-centric networks," in *Communications (ICC), 2015 IEEE International Conference on*, pp. 5747–5752.

[7] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proceedings of the 5th international conference on Emerging networking experiments and technologies*. ACM, 2009, pp. 1–12.

[8] J. Li, H. Wu, B. Liu, J. Lu, Y. Wang, X. Wang, Y. Zhang, and L. Dong, "Popularity-driven coordinated caching in named data networking," in *Proceedings of the eighth ACM/IEEE symposium on Architectures for networking and communications systems*, 2012, pp. 15–26.

[9] Z. Li and G. Simon, "Time-shifted tv in content centric networks: The case for cooperative in-network caching," in *Communications (ICC), 2011 IEEE International Conference on*. IEEE, 2011, pp. 1–6.

[10] I. Psaras, W. K. Chai, and G. Pavlou, "Probabilistic in-network caching for information-centric networks," in *Proceedings of the second edition of the ICN workshop on Information-centric networking*. ACM, 2012, pp. 55–60.

[11] D. H. Lee, C. Dovrolis, and A. C. Begen, "Caching in http adaptive streaming: Friend or foe?" in *Proceedings of Network and Operating System Support on Digital Audio and Video Workshop*. ACM, 2014, p. 31.

[12] C. Liu, I. Bouazizi, M. M. Hannuksela, and M. Gabbouj, "Rate adaptation for dynamic adaptive streaming over http in content distribution network," *Signal Processing: Image Communication*, vol. 27, no. 4, pp. 288–311, 2012.

[13] Y. Liu, J. Geurts, J.-C. Point, S. Lederer, B. Rainer, C. Muller, C. Timmerer, and H. Hellwagner, "Dynamic adaptive streaming over ccn: a caching and overhead analysis," in *Communications (ICC), 2013 IEEE International Conference on*, 2013, pp. 3629–3633.

[14] G. Rossini and D. Rossi, "Evaluating ccn multi-path interest forwarding strategies," *Computer Communications*, vol. 36, no. 7, pp. 771–778, 2013.

[15] A. Afanasyev, I. Moiseenko, L. Zhang *et al.*, "ndnsim: Ndn simulator for ns-3," *University of California, Los Angeles, Tech. Rep*, 2012.