AutoSight: Distributed Edge Caching in Short Video Network

Yuchao Zhang, Pengmiao Li, Zhili Zhang, Bo Bai, Gong Zhang, Wendong Wang, Bo Lian, and Ke Xu

Abstract

In recent years there has been a rapid increase of short video traffic in CDN. While the video contributors change from large video studios to distributed ordinary end users, edge computing naturally matches the cache requirements from short video network. But distributed edge caching exposes some unique characteristics: non-stationary user access pattern and temporal and spatial video popularity pattern, which severely challenge the edge caching performance. While the QoE in traditional CDN has been much improved, prior solutions become invalid in solving the above challenges. In this article, we present AutoSight, a distributed edge caching system for short video network, which significantly boosts cache performance. AutoSight consists of two main components, solving the above two challenges respectively: the CoStore predictor, which solves the non-stationary and unpredictability of local access pattern, by analyzing the complex video correlations; and a caching engine *Viewfinder*, which solves the temporal and spatial video popularity problem by automatically adjusting future horizon according to video life span. All these inspirations and experiments are based on the real traces of more than 28 million videos with 100 million accesses from 488 servers located in 33 cities. Experiment results show that AutoSight brings significant boosts on distributed edge caching in short video network.

Introduction

In recent years, many short video platforms are developing at an incredible speed (such as Kuaishou [1], Youtube Go [2], Instagram Stories [3] and so on). These platforms allow users to record and upload very short videos (usually within 15 seconds). As a result of the massive short videos uploaded by distributed users, the caching problem is becoming more challenging compared with the traditional centralized Content Delivery Network (CDN) whose traffic is dominated by some popular items for a long period of time. To handle scalability, fairness [4] and improve users' Quality of Experience (QoE), the emerging edge computing naturally matches the demand for distributed storage. The above mentioned platforms thus have resorted to employ edge caching servers to store and deliver the massive short videos, so as to avoid that all requests have to be fetched from the backend/origin server, which usually introduces extra user-perceived latency.

There have been tremendous efforts toward better caching performance in traditional CDNs, and these caching algorithms can be classified into two categories. The simple but effective reactive caching algorithms, such as First In First Out (FIFO), Least Recently Used (LRU), Least Frequently Used (LFU), k- LRU and their variants, and the proactive caching algorithms, such as DeepCache [5]. These prior solutions work well in traditional centralize-controlled CDNs, but become invalid in the emerging short video networks, where is naturally implemented with distributed edge caching. Here are the two essential differences.

User Access Pattern Is Non-Stationary: The basic assumption of traditional caching policies is the stationary user access pattern, that is, recently requested or frequently requested content in the past should be kept in cache because such policies assume that these contents have greater chance of being visited in the future. A study in [6] shows that in short video networks, popular contents always become expired very quickly (within tens of minutes), indicating that the popularity in the past could not represent that in the future, and this is the root cause of the failure of these reactive policies.

Video Popularity Pattern Has Spatio-Temporal Characteristics: Our study on the workload of Kuaishou shows that video popularity changes in different patterns during different time periods. For example, during peak hours, it takes less than one hour for a popular video to become unpopular, while during late midnight, such invalidation takes more than three hours. Existing caching policies that try to predict future content popularity always focus on a fixed horizon, thus they fail in the edge caching scenarios.

To address the above challenges, this article presents *AutoSight*, a distributed caching mechanism that works in edge caching servers for short video networks. *AutoSight* allows edge servers to retain respective local caching sight to adapt to local video access patterns and video life spans. *AutoSight*'s distributed design is built on two empirical observations:

- Although the historical video access data is non-stationary on individual edge servers (Fig. 1), making it difficult to make popularity predictions, there is sufficient correlations among videos within the same edge server, because users tend to request related videos, contributing much cross visits in edge servers, and thus improves distributed predictions.

Yuchao Zhang, Pengmiao Li, and Wendong Wang are with Beijing University of Posts and Telecommunications; Wendong Wang is also with the State Key Laboratory of Networking and Switching Technology; Zhili Zhang is with the University of Minnesota; Bo Bai and Gong Zhang are with Huawei Company; Bo Lian is with Kuaishou Company; Ke Xu is with Tsinghua University and Beijing National Research Center for Information Science and Technology. Yuchao Zhang and Wendong Wang are the corresponding authors.

Digital Object Identifier:
10.1109/MNET.001.1900345

0890-8044/20/$25.00 © 2020 IEEE

- Temporal and spatial video popularity patterns bring challenges for future sight of caching policy (Fig. 2), but distributed design allows adaptive future sights, enabling edge servers to make decisions according to different video expiration speeds.

We have implemented a prototype of *AutoSight* and evaluate it with Kuaishou data traces. Experiments show that *AutoSight* achieves much higher hit rate compared with both reactive caching policies and the state-of-the-art proactive caching policies.

Our contributions are summarized as followed:

- Characterizing Kuaishou's workload of short video network to motivate the need for distributed edge caching policy.
- Presenting *AutoSight*, a distributed edge caching mechanism working in edge caching servers for short video network, which solves the problem of non-stationary user access pattern and temporal/spatial video popularity pattern.
- Demonstrating the practical benefits of *AutoSight* by real traces.

Related Work

Here we discuss some representative caching policies in traditional CDN and some related edge caching systems.

Existing Caching Policies

The most representative caching policies such as FIFO, LRU, LFU and their variations are simple but effective in traditional CDN, where the frequency of content visits can be modeled as Poisson distribution. Under these policies, the future popularity of a content is represented by the historical popularity. But in short video network, user access pattern is non-stationary and is no longer in Poisson distribution, so these policies become inefficient in short video network. The same problem exists in the TTL-based (Time-To-Live) caching policies. For example, in the caching (feedforward) networks, where the access pattern is Markov arrival process, the authors in [7] gives joint consideration about both TTL and request models, and drives evictions by stopping times. The authors in [8] set an optimal timer to maximize cache hit rate, but it only works in the case of Pareto-distributed access pattern and Zipf distribution of file popularity, thus certainly becomes invalid in short video network. The authors in [9] propose two caches named f-TTL with two timer, so as to filter out non-stationary traffic, but it still relies on locally observed access patterns to change TTL values, regardless of the future popularity.

One of the promising attempts in recent years is learning-based proactive prediction-based policy. The authors in [5] train a characteristics predictor to predict object future popularity and interoperates with traditional LRU and LFU, boosting the number of cache hits. However, it looks a fixed length into the future to predict object popularity (one to three hours, 12–14 hours and 24–26 hours), ignoring the temporal and spatial video popularity pattern, thus cannot handle the varied life spans in short video network. Pensieve [10] trains a neural network model as an adaptive bitrate (ABR) algorithm. It complements our *AutoSight* framework, in the sense that the proposed dynamic control rules can give us a reference when facing the various life span problem, but it ignores temporal pattern information and only works in live streaming. The authors in [11] introduce reinforcement learning for making cache decisions, but it works in the case that user requests comply with the Markov process, while the proposed *AutoSight* in this article can work under arbitrary non-stationary user access patterns.

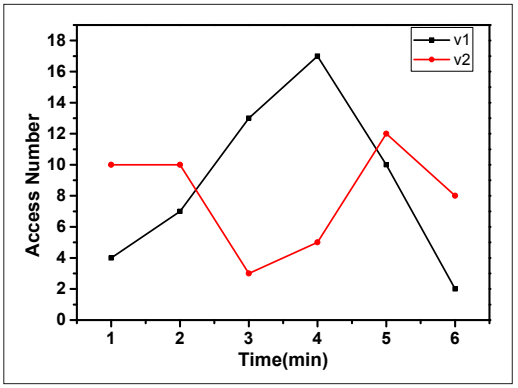

FIGURE 1. Non-stationary video access pattern.

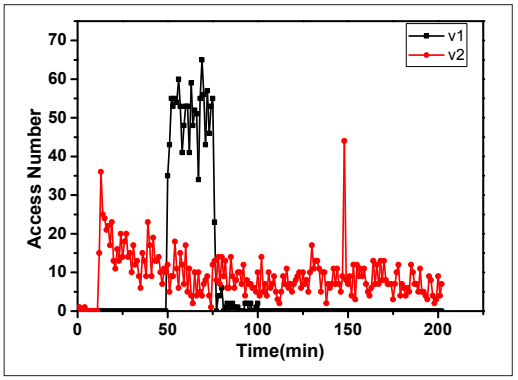

FIGURE 2. Temporal and spatial video popularity pattern.

Edge Caching Systems

Edge computing was proposed to enable offloading of latency-sensitive tasks to edge servers instead of cloud, and has achieved rapid development in many ares such as 5G, wireless, mobile networks [12] and video streaming [13]. The authors in [14] study the content placement problem in edge caching to maximize energy efficiency. The analysis in this work gives us references to design our *AutoSight* network topology, but what we considered under such topology is the caching of short video network rather than energy-saving. The authors in [13] propose a geo-collaborative caching strategy for mobile video network, suggesting that joint caching over multiple edges can improve QoE, which provides strong proof for our *AutoSight* design. While this article tries to reveal the characteristics of different mobile videos, we focus the short video network on edge servers with unique user access pattern and video popularity pattern. The authors in [15] consider a network caching setup, comprising a parent node con-

nected to several leaf nodes to serve end user file requests They propose an efficient caching policy leveraging deep RL, and show capable of learning-and-adapting to dynamic evolutions of file requests, and caching policies of leaf nodes, which provides strong support to our *AutoSight* design.

Background

Before introducing the caching problem, we start by characterizing the short video network, illustrating the essential differences with traditional CDN. Then we show the limitations of existing schemes and draw lessons from real-world traces to inform the design of *AutoSight*. The findings are based on real datasets from Kuaishou's caching network collected during four days, from 9 Oct. 2018 to 12 Oct. 2018.

Characteristics and Challenges

Short video platforms allow users to upload seconds of videos (usually within 15 seconds) to the network. Such convenience on content uploading and accessing finally leads to a revolution in the way network works and the way videos are cached.

Non-Stationary User Access Pattern: In traditional caching system, user access pattern is stationary, in other words, the popular contents at present still have a higher likelihood of receiving more accesses in the next moment. But a study in [6] shows that due to the short life cycle of short videos, popular contents always become expired very quickly (within tens of minutes), indicating that the popularity at present could not represent that in the future. Figure 1 shows the requests number of two particular videos v_1 and v_2, from which we can see the non-stationary of the user access pattern (with sudden increase and decrease). In the first two minutes, v_1 receives less requests while v_2 receives more, but the access pattern reverses in the second minute (request burst on v_1 while valley on v_2). There is also a similar reverse in the fifth minute. Therefore, traditional caching policies become invalid due to the non-stationary access pattern, and a new popularity prediction algorithm in short video network is in urgently needed.

Spatio-Temporal Video Popularity Pattern: In short video network, video popularity changes in various patterns. According to our study on Kuaishou workloads, it takes less than one hour for a popular video to become unpopular during peak hours, while such invalidation takes more than three hours during late midnight. Figure 2 shows the life spans of two videos from an edge server but appear in different time periods, which are significantly different from each other. This figure illustrates the variability of video life span during different time period, and similar situations also occur among different edge servers, which means that in some temporal and spatial, videos get expire at different speeds. This finding motivates us to equip our caching engine with auto-adjusted future horizons, and here we present the definition of Future Horizon. The algorithm *Viewfinder* will be described later.

Definition 1: Future Horizon: It represents how far into the future to plan the caching. It is the length of future time t, during which period of time, the predicted video popularity could represent the popularity of a current video.

Limitations of Existing Solutions

Realizing caching improvement of short video network has some complications. As a first order approximation, we planned to simply borrow existing techniques from traditional CDN. But the above two characteristics results in inefficiency of existing approach that will be described below.

Key Observation 1: The non-stationary access pattern makes the heuristic reactive caching policy invalid.

Explanation: In the traditional stationary network scenario, heuristic reactive caching policies, such as LRU and LFU, can work well by replacing the least used videos. But in short video network with non-stationary access pattern, as the case shown in Fig. 1, those policies will eject v_1 at the second min, because v_1 is less recently used and less frequently used in the first two minutes. However, v_1 got more accesses than v_2 during the third minute, resulting in less caching hit and lower overall hit rate. This is because under the non-stationary access pattern, video popularity in the past could not represent that in the future, and the ejected video at the second minute should be v_2 rather than v_1. This simple example shows the invalidity of heuristic reactive caching policies.

Key Observation 2: Spatio-temporal video popularity pattern on different edge servers and at different time periods make the fixed-horizon proactive caching policy inefficient.

Explanation: When making video popularity prediction, existing learning-based proactive caching policies always look a fixed length into the future, that is, future horizon = Δ_t, which is a fixed length window, and their output is a sequence of k future popularity probabilities, where k represents the number of probabilities to predict during the future Δ_t time. Such policies work well in traditional caching systems, but in the short video network scenario, the average video life span on different edge servers or during different time periods significantly varies from each other; there is no "one size fits all." In the case shown in Fig. 2, if future horizon t is set to two hours, even through those policies can have 100 percent prediction accuracy, the one would be ejected at 50th minute is v_1 rather than v_2, because the predicted popularity of v_1 is less than that of v_2 in the next two hours. But in reality, v_1 is much more popular than v_2 in the near future. Ejecting v_1 will obviously downgrade the performance. This example shows the invalidity of the fixed-horizon proactive caching policy.

AutoSight Design

The core of *AutoSight* is a distributed caching algorithm that makes adaptive caching for edge servers. There are two main components in *AutoSight*: a correlation analyzer named *CoStore*, which solves the non-stationary access pattern problem by analyzing the videos correlations; and a caching engine named *Viewfinder*, which solves the temporal/spatial popularity problem by automatically adjusting horizon to adapt to different edge caching server during different time periods.

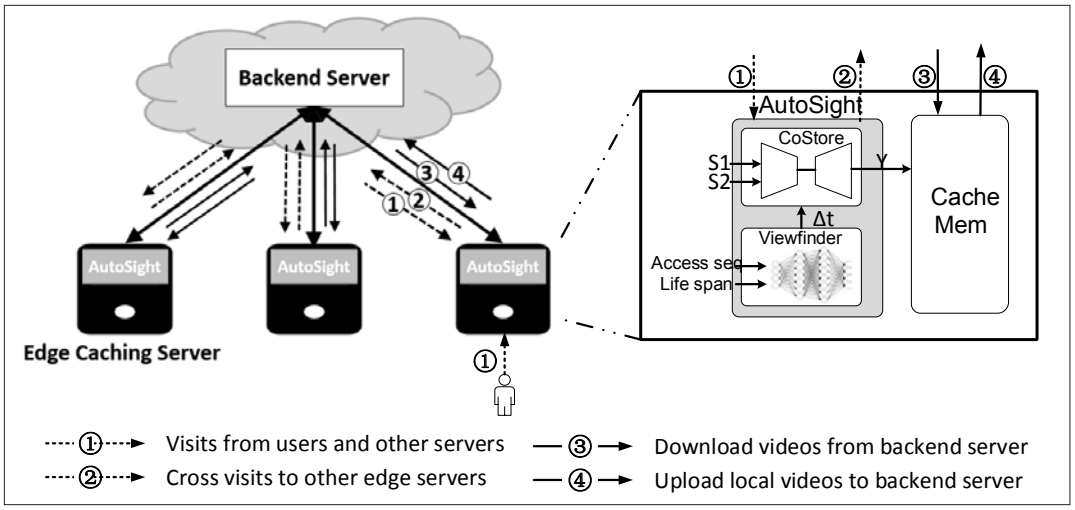

FIGURE 3. The design of distributed edge caching and the framework of *AutoSight*.

System Overview

AutoSight takes an explicit stance that it works the distributed edge caching servers to handle the non-stationary user access pattern and temporal/spatial video popularity pattern, significantly boosting the cache hit rate in the setting of short video network. *AutoSight* uses a correlation-based predictor that predicts the number of times a video will be requested by making real time analysis of videos' cross visits, and uses a caching engine with adaptive future horizon to make caching decisions. The framework of *AutoSight* is shown in Fig. 3. The workflow can be summarized as follows.

- *AutoSight* is implemented in edge servers. It receives both visits from local users and cross visits from other edge servers.
- When the requested videos are stored in local cache memory (called "cache hit"), the edge server sends back the videos directly. When the requested videos are not stored in local cache, the edge server makes a cross visit to other edge servers, and then sends the obtained videos to the requestor.
- When the requested videos are not stored in other edge servers either (called "cache miss"), such videos would be downloaded from backend servers (where all videos are stored).
- To ensure the video completeness of the back-end servers, all of the uploaded videos would be stored in backend servers.

The ultimate goal of *AutoSight* is to increase the number of cache hits and eliminate cache miss, so as to reduce the request response time. We therefore define the hit rate by *cache hit*/(*cache hit* + *cache miss*). In the following *CoStore* and *Viewfinder* design, we try to improve the overall hit rate.

Correlation-Based Predictor: CoStore

Although short video network has non-stationary popularity, that is, video popularity in the past could not represent that in the future, we also found that past video correlations could provide much information to make future popularity prediction [6]. This is because the video correlations in short video network is much informative due to the frequent cross-visits. For a particular video with low popularity, it possibly gets higher popularity if its correlated videos already have high popularity.

Inspired by the long short term memory (LSTM) network that has already shown its dominance in natural language processing (NLP), machine translation and sequence prediction, *CoStore* is built on LSTM, using video correlations as input features, and predicting request numbers within the future horizon. In particular, for video v_i at time t, the input consists of two sets of access sequence: $S_1 = \{r_{v_i}^1, r_{v_i}^2, ..., r_{v_i}^t\}$ and $S_2 = \{r_{v_j}^1, r_{v_j}^2, ..., r_{v_j}^t\}$, where $r_{v_i}^k$ denotes the request number of v_i at time k, and v_j is the most related video to v_i at time t. The output of CoStore is the expected request number of v_i during the future horizon Δ_t.

Caching Engine: Viewfinder

As edge caching servers are experiencing temporal and spatial video popularity pattern, inappropriate future horizon Δ_t (too short-sighted or too long-sighted) always leads to inefficient or even wrong caching decisions. We therefore design *Viewfinder*, which can adjust future horizons automatically. *Viewfinder* aims at finding a suitable horizon for future prediction. In different time frames, it outputs different horizons through on a reinforcement algorithm, and this horizon is the time window for *CoStore* to predict video popularity.

The challenge here is that there are too many options to explore (in the granularity of seconds/minutes), which introduces unacceptable overhead. *Viewfinder* changes it into a classification problem that chooses Δ_t from a predefined set: Δ_t = {60min; 120min; 160min; 180min; 200min; 360min}.

This significantly reduces the computation overhead, and the experiment results show that Viewfinder works well in edge caching servers, disclosing that the quicker videos get expired, the shorter-sight the caching policy should be.

Evaluation

In this section, we evaluate our approach *AutoSight* using real traces, and show the results of applying *AutoSight* on them versus the existing representative policies.

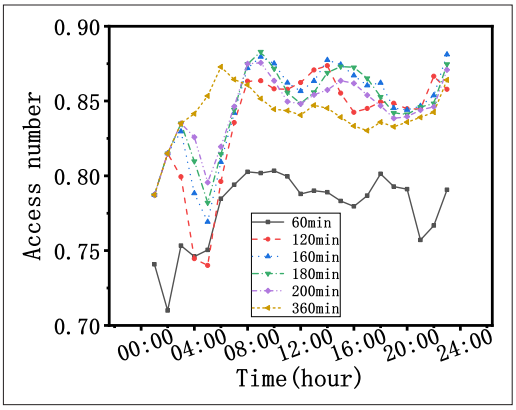

FIGURE 4. The power of *Viewfinder*.

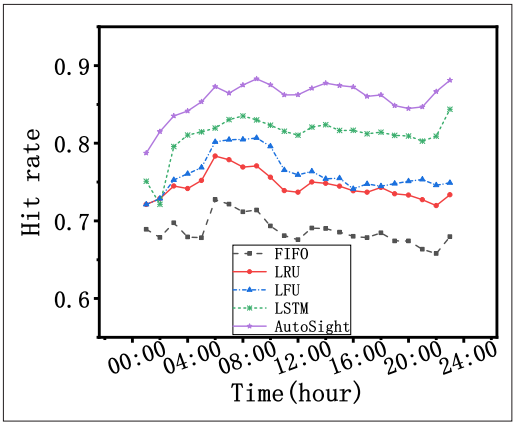

FIGURE 5. Hit rate comparison.

Experiment Setting

Algorithms: We compare *AutoSight* with four existing solutions: FIFO, LRU, LFU and LSTM-based prediction scheme without the auto-adjusted future horizon.

Datasets: We analyze the traces from two cities with 1,128,989 accesses to 132,722 videos in 24 hours. Each trace item contains the timestamp, anonymized source IP, video ID and URL, file size, location, server ID, cache status and consumed time. Thus we can deploy and evaluate different caching policies.

Performance Comparison

We first provide dataset introduction and analysis about these two cities, then we compare the overall hit rate on edge servers among five different caching policies. After that, we look into the proposed *AutoSight* and show the power of *Viewfinder* with auto-adjusted future horizons.

The Power of *Viewfinder*: To evaluate the effect of the caching engine *Viewfinder* with the auto-adjusted future view, we show the cache hit rate with *Viewfinder* set to fixed Δ_t. Fig. 4 shows the corresponding cache hit rate under different future horizons. The optimal value for each period varies with time, that is, during late midnight when video life span is longer, *Viewfinder* tends to be long-sighted, while during the leisure time (e.g., 20:00-21:00pm) when video life span is shorter, *Viewfinder* also tends to be short-sighted. These results further emphasize the necessity of *Viewfinder* with adaptive future horizon.

Overall Cache Hit Rate: As analyzed previously, the non-stationary access pattern would make the reactive caching policy inefficient, and the temporal/spatial video popularity pattern also invalidates learning-based policies with fixed-length future horizon. Figure 5 shows the overall cache hit rate of applying the five policies. *AutoSight* outperforms all the existing algorithms.

Conclusion

In this article, we analyze the Kuaishou dataset and use trace-driven experiments to motivate and investigate edge caching performance for short video network. We first disclose the characteristics on non-stationary user video access pattern and temporal/spatial video popularity pattern, and illustrate the invalidation of existing caching policies by giving two real cases. Then we design *AutoSight*, a distributed edge caching system for short video network, with *CoStore* and *Viewfinder*. Results show that enabling *AutoSight* in edge caching servers could significantly outperform the existing algorithms.

Acknowledgment

The work of Yuchao Zhang was supported in part by the National Natural Science Foundation of China (NSFC) Youth Science Foundation under Grant 61802024 and 61602051; the National Key R&D Program of China under Grant 2019YFB1802603; the Huawei Autonomous and Service 2.0 Project under Grant A2018185; and the CCF-Tencent Rhinoceros Creative Foundation under Grant IAGR20190103. The work of Pengmiao Li was supported in part by the BUPT Excellent Ph.D. Students Foundation under CX2019134. The work of Ke Xu was in part supported by the National Key R&D Program of China with no. 2018YFB0803405; the China National Funds for Distinguished Young Scientists with no. 61825204; and the Beijing Outstanding Young Scientist Program with no. BJJWZYJH01201910003011.

References

[1] Kuaishou, "Kuaishou," https://www.kuaishou.com, 2019.
[2] Y. Go, "Youtube go," https://youtubego.com, 2019.
[3] I. Stories, "Instagram stories," https://storiesig.com, 2019.
[4] Y. Zhang *et al.*, "Going Fast and Fair: Latency Optimization for Cloud-Based Service Chains," *IEEE Network*, vol. 32, no. 2, 2017, pp. 138–43.
[5] A. Narayanan *et al.*, "Deepcache: A Deep Learning Based Framework for Content Caching," *Proc. 2018 Workshop on Network Meets AI & ML. ACM*, 2018, pp. 48–53.
[6] Y. Zhang *et al.*, "Challenges and Chances for the Emerging Short Video Network," *Proc. 2019 IEEE Conf. Computer Commun. (Infocom).*, IEEE, 2019, pp. 1–2.
[7] D. S. Berger *et al.*, "Exact Analysis of TTL Cache Networks: The Case of Caching Policies Driven by Stopping Times," *ACM SIGMETRICS Performance Evaluation Review*, vol. 42, no. 1, 2014, pp. 595–96.
[8] A. Ferragut, I. Rodrguez, and F. Paganini, "Optimizing TTL Caches under Heavy-Tailed Demands," *ACM SIGMETRICS Performance Evaluation Review*, vol. 44, no. 1. ACM, 2016, pp. 101–12.
[9] S. Basu *et al.*, "Adaptive TTL-Based Caching for Content Delivery," *ACM SIGMETRICS Performance Evaluation Review*, vol. 45, no. 1, 2017, pp. 45–46.
[10] H. Mao, R. Netravali, and M. Alizadeh, "Neural Adaptive Video Streaming with Pensieve," *Proc. Conf. ACM Special Interest Group on Data Communication*, ACM, 2017, pp. 197–210.
[11] A. Sadeghi, F. Sheikholeslami, and G. B. Giannakis, "Optimal and Scalable Caching for 5G Using Reinforcement Learning of Space-Time Popularities," *IEEE J. Sel. Topics in Signal Processing*, vol. 12, no. 1, 2018, pp. 180–90.

[12] X. Li *et al.*, "Hierarchical Edge Caching in Device-to-Device Aided Mobile Networks: Modeling, Optimization, and Design," *IEEE JSAC*, vol. 36, no. 8, 2018, pp. 1768–85.
[13] G. Ma *et al.*, "Understanding Performance of Edge Content Caching for Mobile Video Streaming," *IEEE JSAC*, vol. 35, no. 5, 2017, pp. 1076–89.
[14] F. Gabry, V. Bioglio, and I. Land, "On Energy-Efficient Edge Caching in Heterogeneous Networks," *IEEE JSAC*, vol. 34, no. 12, 2016, pp. 3288–98.
[15] G. W. A. Sadeghi and G. B. Giannakis, "Deep Reinforcement Learning for Adaptive Caching in Hierarchical Content Delivery Networks," *IEEE Trans. Cognitive Commun. Networking*, vol. abs/1902.10301, 2019; available: http://arxiv.org/abs/1902.10301.

Biographies

Yuchao Zhang (ORCID: 0000-0002-0135-8915) received her Ph.D. degree from Computer Science Department at Tsinghua University in 2017. Before that, she received the B.S. degree in computer science and technology from Jilin University in 2012. Her research interests include large scale datacenter networks, content delivery networks, data-driven networks and edge computing. She is currently with the Beijing University of Posts and Telecommunications as an associate professor.

Pengmiao Li received M.A. in computer science and technology department, North China University of Science and Technology, Tangshan, China, in 2018. She is currently a Ph.D. candidate in software engineering from Beijing University of Posts and Telecommunications, Beijing, China. Her current research interests include CDN and edge computing.

Zhili Zhang graduated with a B.S. in computer science with highest distinction from Nanjing University, Nanjing, China. He received his M.S. and Ph.D. degrees in computer science from the University of Massachusetts, Amherst in 1992 and 1997, respectively. He joined the Department of Computer Science and Engineering at the University of Minnesota in January 1997, where he is now a full professor. He is a Fellow of IEEE.

Bo Bai [S'09, M11, SM'17] received the B.S. degree (Hons.) from the School of Communication Engineering, Xidian University, Xian, China, in 2004, and the Ph.D. degree from the Department of Electronic Engineering, Tsinghua University, Beijing, China, in 2010. He was a research assistant and a research associate with the Department of Electronic and Computer Engineering, Hong Kong University of Science and Technology, from 2009 to 2010, from 2010 to 2012, respectively. From 2012 to 2017, he was an assistant professor with the Department of Electronic Engineering, Tsinghua University. He is currently a senior researcher with the Theory Lab, 2012 Labs at Huawei Technologies Co., Ltd., Hong Kong.

Gong Zhang is a Principal Member at the Theory Lab at Huawei. His research spans networks, distributed systems and communication system architectures. He has contributed to more than 90 patents. He had been a product development team leader of smart devices in 2002 to pioneer new consumer business for Huawei. He then started to lead the research on future Internet and cooperative communication in 2005. He has been in charge of the Advance Network Technology Research Department, leading research on future networks, distributed computing, database systems, and data analysis since 2009. His recent research focuses on future networks.

Wendong Wang [M'05] received his B.E. and M.E. degrees both from the Beijing University of Posts and Telecommunications, China, in 1985 and 1991, respectively, where he is currently a full professor in the State Key Laboratory of Networking and Switching Technology. He has published over 200 of papers in various journals and conference proceedings. His current research interests are the next generation network architecture, network resources management and QoS, and mobile Internet. He is a member of IEEE.

Bo Lian received his B.S. from Northwestern Polytechnical University, Xi'an, China. He currently works as a senior R&D engineer at the Kuaishou Company. Before that, he worked at Tencent for more than 10 years. His research interests include video streaming, CDN, recommendation, and network infrastructure.

Ke Xu [M'02, SM'09] received his Ph.D. from the Department of Computer Science and Technology at Tsinghua University, where he serves as full professor. He serves as an associate editor for *IEEE Internet of Things Journal* and has guest edited several special issues in IEEE and Springer Journals. His research interests include next generation Internet, P2P systems, Internet of Things, network virtualization, and network economics. He is a member of ACM