Predictive Caching at The Wireless Edge Using Near-Zero Caches

Sherif ElAzzouni*
Dept. of ECE
Ohio State University

Fei Wu*
Dept. of CSE
Ohio State University

Ness Shroff
Dept. of ECE and Dept. of CSE
Ohio State University

Eylem Ekici
Dept. of ECE
Ohio State University

ABSTRACT

In this paper, we study the effect of predictive caching on the delay of wireless networks. We explore the possibility of caching at wireless end-users where caches are typically very small, orders of magnitude smaller than the catalog size. We develop a predictive multicasting and caching scheme, where the Base Station (BS) in a wireless cell proactively multicasts popular content for end-users to cache, and access locally if requested. We analyze the impact of this joint multicasting and caching on the delay performance. Our analysis uses a novel application of Heavy-Traffic theory under the assumption of *vanishing caches* to show that predictive caching fundamentally alters the asymptotic throughput-delay scaling. This in turn translates to a several-fold delay improvement in simulations over the on-demand unicast baseline as the network operates close to the full load. We highlight a fundamental delay-memory trade-off in the system and identify the correct memory scaling to fully benefit from the network multicasting gains.

CCS CONCEPTS

• **Networks** → *Packet scheduling*; **Mobile networks**; **Network management**.

KEYWORDS

Wireless Caching, Multicast, Heavy-Traffic, Delay Analysis

ACM Reference Format:
Sherif ElAzzouni, Fei Wu, Ness Shroff, and Eylem Ekici. 2020. Predictive Caching at The Wireless Edge Using Near-Zero Caches. In *The Twenty-first ACM International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing (Mobihoc '20), October 11–14, 2020, Boston, MA, USA.* , 10 pages. https://doi.org/10.1145/3397166.3409126

*Both authors contributed equally to this research.

Mobihoc '20, October 11–14, 2020, Boston, MA, USA

ACM ISBN 978-1-4503-8015-7/20/10…$15.00
https://doi.org/10.1145/3397166.3409126

1 INTRODUCTION

Caching is poised to play a key role in most proposed future network architectures. The huge increase in mobile traffic, expected to reach 77 exabytes of data per month by 2022 [6], and higher user expectations in terms of high throughput and low latency have pushed the networking community to rely on edge caching as a central tenant of emerging architectures due to its potential to increase network capacity, reduce latency, and alleviate peak-hour congestion among other expected benefits. Recently, there has been a special IEEE JSAC issue that was dedicated to the question of "What role will caching play in future communication systems?" [17]. As an example, the Information Centric Networks (ICN) [1] proposal is an ambitious project to evolve the internet away from the host-centric paradigm to a new content-centric paradigm that decouples senders and receivers. In ICN, the sender requests a certain object, rather than establishing a connection with the object's host, and the network then leverages **in-network caches** to locate that item and deliver it to the user. The reliance on caching has motivated modeling ICN as a "network of caches". Another domain where caching has been gaining significant traction is 5G cellular networks. There have been many works examining the potential of caching in both the core and the RAN edge [24][15].

Central to the recent increased interest in caching is the possibility of caching at the wireless edge [16]. Utilizing Base Stations (BSs)/Access Points (APs) to cache popular content has been proposed [19], which has sometimes been referred to as femtocaching. This enables users to fetch content from the closest Base Station, if possible, decreasing the round-trip delay and reducing network congestion caused by moving content between core servers and edge devices (such as RANs). Although femtocaching can significantly reduce access delay, femtocaching cannot reduce the last mile wireless network delay, thus, we present the following question: "If caching content in last mile edge devices can cause significant delay reduction, can we go one step further and push popular content to end-users devices' caches?". Users can access cached content locally with *zero-delay*. Furthermore, this helps reduce the overall delay by avoiding having to continuously transmit redundant content over the wireless medium, which dissipates expensive wireless resources, that may be the delay bottleneck especially during the busy hour.

The motivating principle behind our work is the "commonality of information", i.e., the same content being requested by a large number of users over a short period of time. Indeed, most content publishers and sharing platforms often have daily *trending* content that is widely requested over a short period of time. This phenomena

has been empirically studied and statistically modeled [22]. From a networking standpoint, serving trending content in an on-demand unicast fashion (as is prevalent in today's networks) unnecessarily strains the network, wasting radio resources on fulfilling redundant requests. Intuitively, the network is better off multicasting trending content to all users that may request it, exploiting the broadcast nature of the wireless channel. Thus, instead of using a single resource (for example the number of 5G-NR resource blocks needed to transmit a video) per each user request, the operator can use a single resource to fulfill *all requests*.

The joint deployment of multicasting and caching has been previously proposed in [18] for energy minimization for transmissions that can tolerate a small amount of delay, for minimization of delay, power, and fetching cost in [25], and maximization of successful transmission probability in [2]. A deep reinforcement learning approach was undertaken in [20] to determine which content to multicast. Perhaps closest to our work is [5], which empirically studied proactively multicasting and caching popular content in 3G/4G networks using real traces, and were able to show that this has the potential to significantly reduce the download volume subject to some practical constraints such as cacheability of popular content. Despite the interest in joint caching and multicasting, a theoretical study of the effect of combining both on delay has been absent. The fundamental idea of multicasting popular content for end users to cache faces two fundamental challenges. The **first challenge** is that users are likely to request the same content at different times. Some works [18] have circumvented that obstacle by waiting for a constant window before multicasting content to all users with outstanding requests, forcing users to wait until the end of the window (on the order of a few minutes) which might be unacceptable for users less tolerant to delay. Conversely, we propose *proactively* multicasting popular content upon generation then exploiting end-users caches to hold popular content. We refer to this scheme as *Predictive Caching*. Predictive Caching consists of two steps:1. Popular content is proactively multicast to all users in the cell. 2. End users cache that content upon receipt in their local caches for a duration equal to the typical requests lifetime, before discarding that content to empty cache space for newer content. The users can then access that content any time from the local cache with zero-delay. The **second challenge** is that end-users have very limited cache sizes. Much of the previous work on wireless caching [19] [9] assumed that the local cache size is on the order of the catalog size. This assumption is not suitable for a variety of wireless networks, where the end devices (e.g., smart phones, tablets, etc.) have limited memory. Thus, we carry out our analysis under the assumption that cache sizes at end users can be very small. More precisely, we show that significant delay reductions are attained even if the end-user cache sizes vanish as the load increases. Our contributions can be summarized as follows:

1. We propose a predictive caching system whereby a BS (or an AP) divides the bandwidth into a load-dependent θ-fraction (constrained by the cache sizes) for predictive caching and a $(1-\theta)$-fraction for traditional on-demand unicast. The BS then uses that θ-fraction to *proactively* multicast popular content for end-users to cache by exploiting the wireless broadcast channel.
2. We model the predictive caching system as a downlink scheduling problem. We introduce the Heavy-Traffic (HT) queuing framework to analyze the delay performance under predictive caching. We use a novel duality framework to simplify the scheduling problem with a load-dependent capacity region into a single dimensional routing problem that is easier to analyze using standard HT tools.
3. We analyze the predictive caching system for vanishing cache sizes vis-a-vis the baseline unicast on-demand system. We show that predictive caching *alters the asymptotic delay scaling in the heavy traffic limit*. This means that the average delay of the predictive caching regime grows slower than the baseline as a function of the network load, leading to significant delay savings as the network approaches full load. We also illustrate via simulations that this delay scaling altering translates to many-fold savings in delay for reasonable cache sizes.
4. We characterize the effect of cache sizes, popularity distribution, number of users in the system, and network load on the delay. We identify and formalize the inherent delay-memory trade-off in the system, which is expected to aid in end-user cache dimensioning. We also characterize the memory scaling as a function of throughput to attain favorable delay scaling.

2 SYSTEM MODEL

2.1 Basline On-demand Unicast System

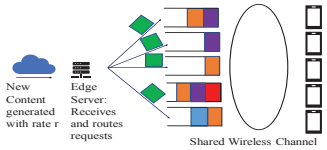


Figure 1: On-Demand Unicast Baseline System Model

The system model is shown in Fig. 1. We assume new content is continuously generated by the network with rate r. Each new content/item has a popularity $p \in [0,1]$ drawn from a prior popularity distribution $f(p)$. Upon content generation, each user will request this new content with probability p. For ease of presentation, we assume the popularity distribution is homogeneous across different users. Nevertheless, the theoretical framework could be extended to the case with heterogeneous popularity distributions. The Base Station (BS) keeps a queue, Q_i, for each user i, to hold their requests until they are served. Each queue has an arrival rate $A_i[t]$ depending on the content requests and a service rate $S_i[t]$ that depends on the BS scheduling algorithm. We assume that the channel between the BS and the end users is a collision channel, where each time slot, the BS can transfer one item to one user. For simplicity, we assume that all items are equal in size, which is justified by the fact that large items can be split up to smaller chunks of equal size. The BS can deploy any scheduling algorithm to serve outstanding requests. However, in the on-demand unicast system, requests have to be fulfilled *individually* and *reactively*.

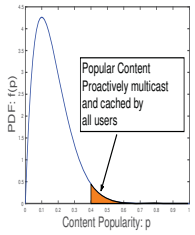
Formally, the on-demand unicast queues evolve as follows:

$$Q_i[t+1] = (Q_i[t] + A_i[t] - S_i[t])^+, \ \forall i = 1, \ldots, N, \text{where} \tag{1}$$

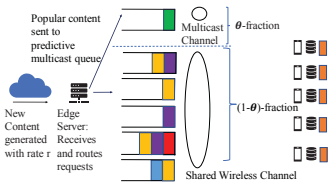
$$S_i[t] \in \{0,1\}, \ \forall i = 1, \ldots, N., \quad \sum_{i=1}^{N} S_i[t] \leq 1, \ \forall i = 1, \ldots, N. \tag{2}$$

$$\lambda = \mathbb{E}[A_i[t]] = r \int_0^1 pf(p)dp \tag{3}$$

$$\epsilon = \mathbb{E}(S_\Sigma[t]) - \mathbb{E}(A_\Sigma[t]) = 1 - N\lambda, \tag{4}$$



(a) Popularity Dist. highlighting $\gamma(\theta, \epsilon)$



(b) Predictive Caching System Model

Figure 2: Predictive Caching Model

where $(x)^+ = \max(0, x)$. $A_\Sigma[t]$ and $S_\Sigma[t]$ are the sum of arrivals and service of all queues at time t, respectively, and ϵ quantifies the network load, so a network with $\epsilon \to 0$ is said to be operating at *full load*. Condition (2) highlights that only one user can be served at a time-slot

We can see from Fig. 1 that popular contents that are requested by multiple users over a short period of time cause a lot of redundancy in the queues, which for crowded cells can cause users to experience large delays. This phenomena was empirically verified in [3], where the traffic from big events was analyzed (in this case the superbowl), and it was reported that popular content (in this case game related content) constituted the majority of traffic requests by users. However, it was reported that simultaneous requests for same content were rare, rendering direct multicast ineffective. This motivates our predictive caching solution that we now present.

2.2 Predictive Caching

In order to reduce the redundancy in the network and exploit the commonality of information and temporal locality in users' content requests, we propose the model in Fig. 2. The main idea is that content that is known to be popular is likely to be widely requested, thus, the BS can proactively multicast those items and have end users cache them and access them locally. In order to do that, we propose the BS divides the bandwidth into a $\theta^{(\epsilon)}$-fraction dedicated to multicasting, and a $(1 - \theta^{(\epsilon)})$-fraction dedicated to on-demand unicast.

The choice of $\theta^{(\epsilon)}$ is determined by two things: The first is the network load ϵ, and the second is the amount of physical memory available at the end users. Physical memory imposes a constraint on how large $\theta^{(\epsilon)}$ can be independent of the load. To see this, assume that all items have a lifetime T, for which they can be requested. This approximates the temporal locality phenomenon of content requests reported in [22]. Assuming a physical cache size of M (with respect to a normalized item size), $\theta^{(\epsilon)}$ can be bounded proportionally to $\frac{M}{T}$ to ensure that items are available in the cache for a duration no shorter than their lifetime. From hereon, we will use the multicast bandwidth fraction, $\theta^{(\epsilon)}$, as representative of the amount of end user cache used to hold content.

We further assume that the multicast channel has a rate of 1 to transmit an item to all users. Once new content is generated in the network, the BS makes a choice on whether to predictively multicast that content and have end-users cache it. The BS makes that decision by simply setting a popularity threshold $\gamma(\theta, \epsilon)$. Any item that has popularity greater than $\gamma(\theta, \epsilon)$ is automatically multicast, where the threshold is chosen to ensure that the multicast queue is stable. Thus, the contents are divided into two sets, as shown in Fig. 2(a), a popular set to be multicast, $C^{(\theta,\epsilon)}$, and a unicast on-demand set, $\overline{C}^{(\theta,\epsilon)}$.

$$C^{(\theta,\epsilon)} = \left\{ c \in \mathit{Contents} \ | p(c) \geq \gamma(\epsilon, \theta) = F^{-1}\left(1 - \frac{\theta^{(\epsilon)}}{r}\right) \right\}, \tag{5}$$

where F^{-1} denotes the inverse CDF of the popularity distribution. This can be equivalently written in terms of the cache arrival rate by picking the threshold to ensure a certain arrival rate of items to the cache size that guarantees that, with high probability, items stay in the cache for a duration no shorter than their request lifetime:

$$r \int_{\gamma(\theta,\epsilon)}^{1} f(p)dp = \theta^{(\epsilon)} \tag{6}$$

Predictive caching reduces both the arrival rates and service rates of unicast queues. We denote those two quantities as $A_i^*[t], S_i^*[t]$, respectively. We can now write down the equations that make up the unicast queues of the Predictive Caching system as follows:

$$Q_i[t+1] = (Q_i[t] + A_i^*[t] - S_i^*[t])^+, \ \forall i = 1, \ldots, N. \tag{7}$$

$$S_i^*[t] \in \{0, 1\}, \ \forall i = 1, \ldots, N, \ \sum_{i=1}^{N} S_i^*[t] = \begin{cases} 0 & \text{w.p. } \theta^{(\epsilon)}, \\ 1 & \text{w.p. } 1 - \theta^{(\epsilon)}. \end{cases} \tag{8}$$

$$\mathbb{E}[A_i^*[t]] = \lambda^* = r \int_0^{\gamma(\theta,\epsilon)} pf(p)dp \tag{9}$$

$$\delta(\epsilon, \theta) = \mathbb{E}[S_\Sigma[t]] - \mathbb{E}[A_\Sigma[t]] = 1 - \theta^{(\epsilon)} - N\lambda^* \tag{10}$$

Note that the predictive caching regime, arrivals to the unicast queues exclude items that belong to the set $C^{(\theta,\epsilon)}$, since those items are multicast, cached, and accessed locally by end-users whenever requested. Similarly, the channel is accessible by unicast queues in (8) only for a fraction of $1 - \theta^{(\epsilon)}$, and for a $\theta^{(\epsilon)}$-fraction, the wireless channel is dedicated to serving the multicast queue. Thus, the load of delivering those requests is now deferred to the multicast queue at the cost of a $\theta^{(\epsilon)}$-fraction of the bandwidth.

The key question in this paper is: Can we achieve delay gains even if the cache available is asymptotically zero as the network approaches the full load, i.e., $\theta^{(\epsilon)} \to 0$ as $\epsilon \to 0$? An affirmative answer to that question means that in practice, even small caches at end users could be leveraged to reduce the overall user delay. We carry out a detailed analysis that shows that predictive caching can improve the asymptotic delay scaling for vanishing caches.

3 ANALYSIS OF UNICAST ON-DEMAND SYSTEM

We start by analyzing the baseline unicast on-demand system to provide a basis for comparison when we analyze the predictive caching scheme. We first derive a lower bound on the sum queue lengths at the BS by utilizing the resource pooling lower bound [4]. In the unicast problem, the BS scheduler makes a decision on which user should be served every time slot depending possibly on the requests' queue lengths (For example the scheduler can give the channel to the user with the longest queue of outstanding requests). It is known that the capacity region is $\mathcal{R} = \text{Convex Hull}(\mathcal{S})$, where $\mathcal{S}$ is the set of feasible schedules. Under our assumption of a collision homogeneous channel, $\mathcal{S}$ could be written as $\mathcal{S} = \{S_i, \ i = 1, \ldots, N, |S_i \in \{0, 1\}, \forall i, \sum_{i=1}^{N} S_i = 1\}$

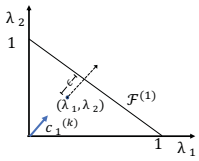


Figure 3: Predictive Caching Model

Since the number of users, N, is finite, the region $\mathcal{S}$ is a polyhedron that can be fully described by as intersection of hyperplanes as follows:

$$\mathcal{R} = \{r \geq 0 : \langle c^{(k)}, r\rangle \leq b^{(k)}, k = 1, \ldots, K\} \tag{11}$$

where K is the number of hyperplanes describing the polyhedron. The notation $\langle ., . \rangle$ indicates an inner product. The $k-$th hyperpane, $\mathcal{H}^{(k)}$, can be described by the pair $(c^{(k)}, b^{(k)})$. For the special case of the collision channel, the capacity region can be described by the single hyperplane $\mathcal{R} = \{\mathbf{r} : \mathbf{r} \geq 0, \frac{1}{\sqrt{N}}\langle \mathbf{1}, \mathbf{r}\rangle \leq \frac{1}{\sqrt{N}}\}$. We plot in Fig. 3 the capacity region for the two user case.

Having defined the capacity region, we can now utilize the "resource pooling" system to derive a lower bound on the steady-state queue lengths in the heavy traffic setting. The queue lengths process $\{Q_i[t]\}_{i=1}^N$ can be modelled as a Markov chain that converges in distribution to steady-state $\{\overline{Q}_i\}_{i=1}^N$ when the system is stable, i.e., the Markov chain is positive Harris recurrent. We are interested in characterizing the steady-state of the sum queue lengths. Intuitively, pooling the resources of all queues into one queue leads to a natural lower bound on the system. We parameterize the system with the network load, $\epsilon = 1 - N\lambda$, and derive the steady-state sum queue lengths at this load $\sum_{i=1}^N \overline{Q}^{(\epsilon)}$, in particular, we are interested in the behavior of the system in the *Heavy-traffic* limit, i.e., when $\epsilon \to 0$ pushing the operating point to the boundary of the capacity region. This idea was introduced and applied in [4] for both the routing and scheduling problems. The next Lemma characterizes the resource pooling lower bound for the on-demand unicast baseline system:

Lemma 3.1. *In the on-demand unicast system described above, where $\epsilon = 1 - N\lambda$, the sum queue lengths in the network is lower-bounded as follows:*

$$\mathbb{E}[\sum_{i=1}^N \overline{Q}_i^{(\epsilon)}] \geq \frac{\zeta^{(\epsilon)}}{2\epsilon} - \frac{1}{2} \tag{12}$$

where $\zeta^{(\epsilon)} = \sqrt{N}(\sigma_A^{(\epsilon)})^2 + \epsilon^2$, and $(\sigma_A^{(\epsilon)})^2$ is the variance of the arrivals for each queue at each time slot. Furthermore, using the conditional variance of the arrivals, we can show that $(\sigma_A^{(\epsilon)})^2 = \mu_p^{(\epsilon)} - (\mu_p^{(\epsilon)})^2$, where μ_p is the mean of the prior popularity distribution. Furthermore, taking the heavy traffic limit as $\epsilon \to 0$, the steady-state sum queue lengths limit is asymptotically lower bounded as follows:

$$\liminf_{\epsilon \to 0} \epsilon \mathbb{E}\left[\sum_{i=1}^N \overline{Q}_i\right] \geq \frac{\zeta}{2} \tag{13}$$

Where $\overline{Q}_i$ is the limit of $\overline{Q}_i^{(\epsilon)}$ as $\epsilon \to 0$. $\zeta = \sqrt{N}\sigma_A^2$, where $\sigma_A^2 = \lim_{\epsilon \to 0}(\sigma_A^{(\epsilon)})^2$. Equivalently, the sum queue lengths in the steady-state asymptotically scales as $\Omega(\frac{1}{\epsilon})$.

The proof of the result can be directly obtained by applying the resource pooling lower bound for the generic single queue in [4] to a queue with an arrival rate of $\langle \mathbf{1}, \mathbf{A}[t]\rangle$, where $\mathbf{1} = [1, 1, \ldots, 1]$ and a deterministic service rate equal to 1.

In Lemma 3.1, it is important to note that the expected steady state sum queue lengths in the on-demand system scales as $\Omega(\frac{1}{\epsilon})$. We will show that the predictive caching fundamentally alters this scaling to a slower scaling leading to arbitrarily large delay saving in the HT limit. In order to do that we introduce the duality framework that maps the scheduling problem into an easier routing problem.

4 DUALITY FRAMEWORK

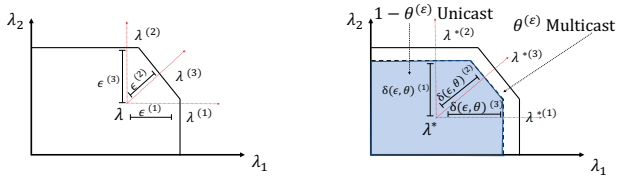
4.1 Capacity of Predictive Caching

In order to motivate our duality framework, it is essential to understand how predictive caching alters the capacity region. Consider a general capacity region, C, for an on-demand system. Recall that the predictive caching reserves a fraction $\theta^{(\epsilon)}$ for multicasting popular content for end users to cache. We make the key design choice of *vanishing cache size*: $\theta^{(\epsilon)} \to 0$ as $\epsilon \to 0$, i.e., as the network load approaches the full load, the multicast bandwidth decreases until this multicast bandwidth vanishes at the *full-load*. The motivation for this choice is two-fold:

1. We aim to show that small memory sizes typical of end-user devices, such as hand-held devices, can still be used to achieve significant delay savings. Having the *vanishing* cache size assumption emphasizes that even diminishing caches can be useful at high network loads, furthermore, a smaller cache at high load can be more useful than a larger cache at lower load. This is crucial to show that our results hold for practical systems and do not make the common assumption in wirless caching previous works [19] [7] [9] [12] that the cache size can hold a significant fraction of the content catalog.
2. As the network approaches full-load, less resources could be dedicated to predictive caching, and the scheduler needs to dedicate all of the resources to fulfill on-demand requests to guarantee stability.

Since $\theta^{(\epsilon)}$ of the the bandwidth is sacrificed to predictive caching, the BS unicast scheduler sees a reduced capacity region $C^{(\epsilon)}$, as shown in Fig. 4. Specifically, the scheduler gets an aggregate $1-\theta^{(\epsilon)}$ capacity to allocate to unicast traffic. However, the average user arrival rate is also reduced from λ_i to λ_i^* due to predictive caching of popular content. Thus, the network load changes from ϵ to a new load: $\delta(\epsilon, \theta)$ (we write the new load δ as a function of ϵ and θ to emphasize the dependence). Ideally, we design our algorithm such that $\delta(\epsilon, \theta) > \epsilon$. If we can show that the average delay of the unicast queues under predictive caching scales as $O(\frac{1}{\delta(\epsilon,\theta)})$, then this establishes that predictive caching alters the *asymptotic scaling* of delay at the heavy-traffic leading to arbitrarily large delay savings as $\epsilon \to 0$.

We can see in Fig. 4 that the capacity region, $C^{(\epsilon)}$ is dependent on the load, ϵ, with the property that $\lim_{\epsilon \to \infty} C^{(\epsilon)} = C$, due to the vanishing caches assumption. The standard analysis for scheduling algorithms in the HT regime [4] [23][13] [11] is carried out under the assumption of a fixed capacity region independent of ϵ. It is unclear how the analysis can be altered to fit the load-dependent regime, since now the Hyperplanes that define the capacity region



(a) On-demand Unicast Capacity Region: C

(b) Predictive Caching Capacity Region: $C^{(\epsilon)}$

Figure 4: Capacity Region of different delivery systems

are dependent on ϵ. To avoid a complicated analysis, we introduce a duality framework that transforms the scheduling problem into a simpler routing problem more amenable to standard HT tools. We note that a version of duality was presented in [21] in finite-buffer systems (between routing queue states and scheduling residual capacities) to derive throughput-optimal policies.

4.2 Duality between Scheduling and Routing

We now present a method to map the scheduling problem into an equivalent routing problem that allows us to extend the existing HT framework to directly analyze the predictive caching problem. Intuitively, scheduling a non-empty queue (denoted as i) over a time-slot leads to reduction of that queue by one request. This can be equivalently viewed as all queues except queue i adding one request if all queues have a constant independent service equal to 1 request/slot, as shown in Fig. 5. Thus, Instead of solving the scheduling problem by determining which queue to schedule at time t, we can equivalently solve a routing problem by determining where to route $N-1$ "artificial arrivals". We denote those artificial arrivals as $\mathbf{B}[t]$. We formalize this intuition in the next Theorem.

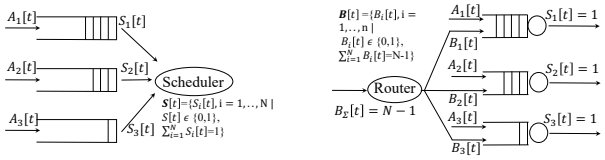
Theorem 4.1. *Duality of Routing and Scheduling Problems: Given two Systems $\mathcal{U}_1$ and $\mathcal{U}_2$. $\mathcal{U}_1$ is a scheduling problem described by equations* (1)-(4) *and some (possibly random) scheduling rule $\mathbf{s}[t] = g(\mathbf{Q}[t])$, that depends on the queue lengths at time t. $\mathcal{U}_2$ is an $(N-1)$-routing problem described by the following equations:*

$$Q_i[t+1] = (Q_i[t] + A_i[t] + B_i[t] - S_i[t])^+, \ \forall i = 1, \dots, N. \quad (14)$$
$$S_i[t] = 1, \qquad \mathbb{E}[A_i[t]] = \lambda, \ \forall i = 1, \dots, N.$$
$$B_i[t] \in \{0,1\}, \qquad \mathbf{B}_\Sigma[t] = \sum_{i=1}^{N} B_i[t] = N-1, \ \forall i = 1, \dots, N.$$
$$\epsilon = \mathbb{E}[S_\Sigma] - \mathbb{E}[A_\Sigma] - \mathbb{E}[B_\Sigma] = 1 - N\lambda \quad (15)$$

Where the router makes routing decisions according to some, possibly random function, $\mathbf{B}[t] = h(\mathbf{Q}[t])$, depending on the queue lengths at time t.
If the scheduling rule in $\mathcal{U}_1$ and the routing rule in $\mathcal{U}_2$ satisfy:

$$P\{g(\mathbf{Q}) \text{ chooses } Q_i \text{for scheduling}\} = P\{h(\mathbf{Q}) \text{ routes requests to all queues except } Q_i\} \ \forall i = 1, \dots, N \quad (16)$$

*Then the systems $\mathcal{U}_2$ and $\mathcal{U}_1$ are **Sample-path equivalent**, i.e., for the same sample path (same realizations of requests arrivals and scheduling/routing random decisions), $\mathbf{Q}[t]$ are equal in $\mathcal{U}_1$ and $\mathcal{U}_2$ with probability 1 at all times t, assuming the same initial state $\mathbf{Q}[0]$.*



(a) Single Resource Scheduling Problem

(b) $N-1$-Routing problem

Figure 5: Duality between routing and scheduling problems

The Proof is straightforward by induction. We give two examples to further illustrate the duality condition (16). The first example is the Longest-Queue-First (LQF) scheduling algorithm, breaking ties uniformly at random. Thus $g(\mathbf{Q}[t]) = \text{RAND}\{\text{argmax}(\mathbf{Q}[t])\}$. This scheduling rule can be mapped to the Join-the-Shortest $N-1$ Queues (JS$(N-1)$Q), as we can express the routing rule that routes to $N-1$ shortest queues as follows $g(\mathbf{Q}[t]) = \mathbf{Q} \setminus \text{RAND}\{\text{argmax}(\mathbf{Q}[t])\}$. where '\' is the set difference notation. It is straightforward to see that LQF and JS$(N-1)$Q satisfy (16). Another example is Random Scheduling (RS) and $(N-1)$–Random Routing $(N-1)$-RR which respectively make the routing and scheduling decisions uniformly at random (where each queue can get at most one request in the routing system). It is easy to see that RS and $(N-1)$RR satisfy (16).

5 PERFORMANCE OF PREDICTIVE CACHING

5.1 Main Result

We now analyze a predictive caching algorithm that we call Predictive Caching Longest-Queue First (PC-LQF) that follows the outline of Section II. PC-LQF multicasts items that have a popularity p higher than threshold $\gamma(\epsilon, \theta)$ for end users to cache, and unicasts all other items. PC-LQF serves the multicast queue with probability $\theta^{(\epsilon)}$, and the unicast queue with probability $1-\theta^{(\epsilon)}$. In the unicast mode, the BS schedules the longest queue for unicast transmissions breaking ties randomly. PC-LQF is summarized in Algorithm 1.

Algorithm 1: Predictive Caching-Longest Queue First (PC-LQF)

```
for time slot t do
    Receive new content items generated by the network
    if new item c has popularity p_c ≥ γ(ε, θ) then
        Send item c to Multicast Queue Q_M to be sent to all users to cache
    else
        Only forward c to Q_i when requested by user i
    w.p. θ^(ε)
        Serve Multicast Queue, Q_M
    w.p. 1 − θ^(ε)
        Choose the longest unicast queue to serve, i.e.,
                    s[t] = RAND{argmax_i Q_i[t]}
```

Having described the PC-LQF algorithm. We are now interested in the delay scaling in the HT limit as $\epsilon \to 0$, under the vanishing caches assumption.

Theorem 5.1. *Main Result: Consider the Predictive Caching System shown in Fig. 2 with homogeneous arrivals equal to* $\lambda_i = r\int_0^1 pf(p)dp$, *satisfying* $\epsilon = 1 - N\lambda > 0$. *If the BS applies Algorithm 1, then, the system is stable as long as* $\gamma(\theta,\epsilon) > \frac{1}{N}$, *and the limiting steady-state queue length vector* $\mathbf{Q}^{(\epsilon)}$ *satisfies the following:*

$$\mathbb{E}\Big[\sum_{i=1}^{N}\overline{Q}_i^{(\epsilon)}\Big] \le \frac{\zeta^{*(\epsilon)}}{2\delta(\epsilon,\theta)} + \overline{B}^{*(\epsilon)} \tag{17}$$

where $\zeta^{*(\epsilon)} = (N(\sigma_A^{(\epsilon)})^2 + \theta(1-\theta) + \delta(\epsilon,\theta)^2)$, *and* $\overline{B}^{*(\epsilon)} = o(\frac{1}{\delta(\epsilon,\theta)})$, *with* $(\sigma_A^{(\epsilon)})^2$ *being the variance of a single queue arrival at any time slot. Furthermore, the scaling factor* $\delta(\epsilon,\theta)$ *can be bounded as follows*

$$\epsilon + \theta(N\gamma(\theta,\epsilon) - 1) \le \delta(\epsilon,\theta) \le \epsilon + \theta(N-1) \tag{18}$$

Additionally, in the ***Heavy-traffic limit*** *as* $\epsilon \to 0$, *which implies* $Nr\lambda \to 1$, $(\sigma_A^{(\epsilon)})^2 \to \sigma_A^2$, *and* $\theta^{(\epsilon)} \to 0$, *by the vanishing caches assumption, the asymptotic limit becomes:*

$$\limsup_{\epsilon\to 0}\ \delta(\epsilon,\theta)\mathbb{E}\Big[\sum_{i=1}^{N}\overline{Q}_i\Big] \le \frac{\zeta}{2} \tag{19}$$

where $\zeta = N\sigma_A^2$.

Finally, PC-LQF is Heavy-Traffic optimal within all algorithms with predictive caching capability.

Before proving our main result, we highlight a few key observations from the main theorem:

1. The most important observation is that predictive caching alters the asymptotic delay scaling, namely, it significantly "slows down" the delay build-up as ϵ vanishes. To see that it is useful to contrast the scaling in Theorem 5.1 with Lemma 3.1. We see that for the baseline unicast system, the sum queue lengths in the steady state is lower bounded by a $\Omega(\frac{1}{\epsilon})$ scaling, whereas the PC-LQF system is upper bounded by $O(\frac{1}{\delta(\epsilon,\theta)})$ scaling. Under the assumption that $\delta(\epsilon,\theta) > \epsilon$ (we will show the mild conditions for this assumption to be true), the average queue lengths under PC-LQF are arbitrarily smaller than the unicast system as $\epsilon \to 0$. This leads to many-fold delay savings in practical heavily-loaded systems as seen in simulations.
2. To get a sufficient condition for delay scaling improvement, it is useful to note that having $\gamma^{(\epsilon,\theta)} > \frac{1}{N}$, guarantees $\delta(\epsilon,\theta) > \epsilon$. This means that the BS should use the rule in (5) for caching *as long as the item's popularity* $p > \frac{1}{N}$. This is expected since this rule guarantees that an item that is multicast and cached is expected to be requested more than once, giving multicasting gains over the unicast system. Thus, this rule can lead the BS to be more conservative in multicasting in cases where end users have no commonality of information to be exploited.
3. The main result answers the question of "How should cache sizes scale with respect to the network load to maintain asymptotic reduction in delay?". This can be seen by inspecting (18): To obtain asymptotic delay improvement, we need the second term in the bound to be on the order of $\Omega(\epsilon)$. The next corollary characterizes the expected improvement in delay according to the cache size scaling with ϵ.

Corollary 5.2. *Under a mild condition that* $\lim_{\epsilon\to 0}\gamma(\epsilon,\theta) > c$, *for some constant c, the cache-size scaling with the network load* ϵ *determines the improvement in delay as follows:*

1. Case I: $\theta^{(\epsilon)} = o(\epsilon)$*: No improvement in delay asymptotics can be achieved since* $\delta(\epsilon,\theta)$ *and* ϵ *are on the same order.*
2. Case II: $\theta^{(\epsilon)} = \Theta(\epsilon)$*: Improves the constant in the delay asymptotics.*
3. Case III: $\theta^{(\epsilon)} = \omega(\epsilon)$*: Improves the scaling in the delay asymptotics.*

This corollary introduces the fundamental requirement for delay improvement in terms of cache scaling. This requirement can translate practically by having end-users allocate the appropriate device memory for content caching at high network loads when instructed by the BS. Thus at congestion, the end-users can still decrease their cache sizes to zero as long as the decrease is slower than ϵ.
4. From Corollary 5.2 , we see that scaling memory as $\Omega(\epsilon)$ alters the asymptotic delay scaling as $O\big(\frac{1}{\epsilon+N\Omega(\epsilon)}\big)$, reducing the average delay *linearly* in the number of users N. This is the *Multicasting Gain* that the predictive caching offers over the baseline system.
5. Finally, the theorem points to the effect of "commonality of information". Given any value of ϵ, we see from (5), that a heavier tail of the popularity distribution means a higher value for the threshold $\gamma(\epsilon,\theta)$ which in turn further slows down the delay scaling lower bound as ϵ grows. This is expected since a heavier tail means the existence of more high popularity items that are useful to cache. Equivalently, a heavier tail implies that users are more likely to request similar contents from the distribution tail which increases the multicasting gains.

5.2 Proof of Main Result

The proof of the main result utilizes the main idea used in HT analysis of routing and scheduling algorithms presented in [4]. The outline of the proof can be broken down into four steps.

(1) We find the appropriate dual system (as in Fig. 5) that satisfies Theorem 4.1. The analysis is carried out for the dual system.
(2) We derive the resource pooling lower bound to be used to show Heavy-traffic optimality of PC-LQF.
(3) We show that under PC-LQF, the queue lengths are close to each other at high network loads. This is formally known as the State-space collapse.
(4) We use the results from state-space collapse to derive the main result in Theorem 5.1. We omit proving the stability condition for space limitations.

5.2.1 Deriving the Dual System. We follow the guidelines implied by Theorem 4.1 to derive an equivalent dual system to the PC-LQF system. We refer to the dual system as Predictve Caching-Join the Shortest $(N-1)$ Queues (PC-JS$(N-1)$Q). The new system retains the structure of PC-LQF by forwarding the most popular content to a multicast queue, and splitting the bandwidth between on-demand unicast and multicast. However, the key difference is that in the new system model, the queues are not contending for the wireless shared channel. The dual system equations can be derived from Theorem 4.1 as follows:

$$Q_i[t+1] = Q_i[t] + A_i^*[t] + B_i^*[t] - S_i^*[t] + U_i[t],\ \forall i = 1,\dots,N.$$

$$B_\Sigma[t] = \begin{cases} 0 \text{ w.p. } \theta^{(\epsilon)}, \\ N-1 \text{ w.p. } 1-\theta^{(\epsilon)}. \end{cases}, \quad S_i[t] = \begin{cases} 0 \text{ w.p. } \theta^{(\epsilon)}, \\ 1 \text{ w.p. } 1-\theta^{(\epsilon)}. \end{cases}$$

$$\mathbb{E}[A_i^*[t]] = \lambda^* = \int_0^{\gamma(\theta,\epsilon)} pf(p)dp, \quad \delta(\epsilon,\theta) = 1 - \theta^{(\epsilon)} - rN\lambda^*$$

Where $B_\Sigma[t]$ is the sum of "artificial" arrivals to be routed in the new equivalent dual problem. Also $S_i[t]$ and $B_\Sigma[t]$ are coupled, meaning they follow the same "coin flip" to decide the value they take at time t. Finally, $U_i[t]$ denotes the unused service of queue i at time t, namely, $U_i[t] = \max(0, S_i[t] - A_i[t] - B_i[t] - Q_i[t])$.

5.2.2 Resource Pooling Lower Bound. We start by providing a lower bound for the queue lengths in the next lemma

Lemma 5.3. *For any predictive caching system, the steady-state sum queue lengths for a load $1 - N\lambda = \epsilon$ can be lower bounded as follows:*

$$\mathbb{E}[\sum_{i=1}^{N} \overline{Q}_i^{(\epsilon)}] \geq \frac{\zeta'^{(\epsilon)}}{2\delta(\epsilon,\theta)} - \frac{N}{2} \tag{20}$$

where $\zeta'^{(\epsilon)} = N(\sigma_A^{(\epsilon)})^2 + \theta(1-\theta) + \delta(\epsilon,\theta)^2$

The proof is omitted for space limitations but the bound can be obtained by applying a Lyapunov analysis to the resource pooling lower bound.

5.2.3 State-Space Collapse. In order to prove state-space collapse, we use the result in [8] for bounding the moments of a Markov Chain defined on a countable state-space:

Lemma 5.4. *[8] For an irreducible and aperiodic Markov chain $\{Q[t]\}_{t\geq 0}$ over a countable state space χ, suppose $Z : \chi \to \mathbb{R}_+$ is a nonnegative-valued Lyapunov function. We define the drift of Z at Q as:*

$$\Delta(Z) \triangleq [Z(Q[t+1]) - Z(Q[t])]\mathcal{I}(Q[t] = Q), \tag{21}$$

where $\mathcal{I}(.)$ is the indicator function. If the following conditions are satisfied:

(1) The exists $\eta > 0$ and $B < \infty$ such that:

$$\mathbb{E}[\Delta Z(Q)|Q[t] = Q] \leq -\eta, \ \forall Q \in \chi \ \text{ with } Z(Q) \geq B. \tag{22}$$

(2) There exists a $D < \infty$ such that

$$\mathbb{P}(|\Delta Z(Q)| \leq D) = 1 \ \text{ for all } Q \in \chi \tag{23}$$

(3) $\{Q[t]\}_t$ is positive recurrent.

Then $Z(Q[t])$ converges in distribution to a random variable that satisfies

$$\mathbb{E}[e^{\theta^* \overline{Z}}] \leq C^* \tag{24}$$

which implies all moments of $\overline{Z}$ exist and are finite.

A key step of proving our main result is showing that as $\epsilon \to 0$, under PC-JS$(N-1)$Q, all user queue lengths are close to each other in size. This enables us to show that at the steady state, the system behaves as a single reource pooling queue **that scales slower than the unicast regime**.

We parametrize the model by the unicast $\epsilon = 1 - N\lambda^{(\epsilon)}$, where $\lambda = r\int_0^1 pf(p)dp$. We are interested in the queue-length process $\{\mathbf{Q}^{(\epsilon)}[t]\}_t$ and its steady state $\overline{\mathbf{Q}}^{(\epsilon)}$ under the PC-JS$(N-1)$Q policy. This is done by decomposing the queue lengths vector into two components: a parallel component that averages all queue lengths, i.e., a projection of the $\mathbf{Q}$ onto the vector $c = \frac{1}{\sqrt{N}}\mathbf{1}$, and a perpendicular component that quantifies the differences in queue lengths:

$$\mathbf{Q}_\parallel^{(\epsilon)} \triangleq \frac{\sum_{i=1}^N Q_i^{(\epsilon)}}{N}\mathbf{1} \qquad \mathbf{Q}_\perp^{(\epsilon)} \triangleq \left[Q_i - \frac{1}{N}\sum_{i=1}^N Q_i^{(\epsilon)}\right]_{i=1}^N$$

From the continuous mapping theorem, we know that the convergence of $\{\mathbf{Q}^{(\epsilon)}[t]\}_t$ implies the convergence of $\{\mathbf{Q}_\parallel^{(\epsilon)}[t]\}_t$ and $\{\mathbf{Q}_\perp^{(\epsilon)}[t]\}_t$. Following the approach of [4], we are interested in showing that $\overline{\mathbf{Q}}_\perp^{(\epsilon)}$ is uniformly bounded for all $\epsilon > 0$.

Proposition 5.5. *Under the dual system considered parametrized by $\epsilon = 1 - N\lambda$, applying the JS$(N-1)$Q routing to the arrivals $\{B_\Sigma[t]\}_t$. Then for any feasible arrival rate, i.e., $N\lambda < 1$, there exists a sequence of finite numbers $\{N_r\}_{r=1,2,\ldots}$ such that $\mathbb{E}\left[\left\|\overline{\mathbf{Q}}_\perp^{(\epsilon)}\right\|^r\right] < N_r$, for all $\epsilon > 0$, and for all $r = 1, 2, \ldots$.*

Before proving the preposition, we state an important Lemma from [4] that bounds the Lyapunov function of $\mathbf{Q}_\perp$ in terms of other functions that are easier to manipulate.

Lemma 5.6. *[4] For any queuing system where the arrival and service processes of each queue are bounded every time slot by $A_{\max}$ and $S_{\max}$, respectively. Let $\Delta L(X) = (L(X[t+1]) - L(X[t]))\mathbf{1}(X[t] = X)$, denote the single-step drift for any appropriate Lyapunov function, L, and any state, X. Then the following bounds hold for $V_\perp(\mathbf{Q})$:*

$$\Delta V_\perp(\mathbf{Q}) \leq \frac{1}{2\|\mathbf{Q}_\perp\|}(\Delta W(\mathbf{Q}) - \Delta W_\parallel(\mathbf{Q})), \ \forall \mathbf{Q} \in \mathbf{R}_+^N \tag{25}$$

$$|\Delta V_\perp(\mathbf{Q})| \leq 2\sqrt{N}\max(A_{\max}, S_{\max}) \, \forall \mathbf{Q} \in \mathbf{R}_+^N \tag{26}$$

We are now ready to prove the main state-space collapse result:

Proof. We begin the proof by analyzing the Lyapunov drift of the function $W(\mathbf{Q})$. For convenience we drop the ϵ superscript notation and the time index t.

$$\mathbb{E}[\Delta W[\mathbf{Q}]] = \mathbb{E}[\|\mathbf{Q}[t+1]\|^2 - \|\mathbf{Q}[t]\|^2 \,|\mathbf{Q}] \tag{27}$$

$$= \mathbb{E}[\|\mathbf{Q} + \mathbf{A} + \mathbf{B} - \mathbf{S}\|^2 + 2\langle \mathbf{Q} + \mathbf{A} + \mathbf{B} - \mathbf{S}, \mathbf{U}\rangle + \|\mathbf{U}\|^2 - \|\mathbf{Q}\|^2 \,|\mathbf{Q}]$$

$$\overset{(a)}{\leq} \mathbb{E}[\|\mathbf{Q} + \mathbf{A} + \mathbf{B} - \mathbf{S}\|^2 - \|\mathbf{Q}\|^2 \,|\mathbf{Q}] \overset{(b)}{\leq} 2\mathbb{E}[\langle \mathbf{Q}, \mathbf{A} + \mathbf{B} - \mathbf{S}\rangle|\mathbf{Q}] + 2N \tag{28}$$

where (a) follows form $(Q_i + A_i + B_i - S_i)U_i < 0)$, and (b) is since A_i, B_i, and S_i are all bounded by 1.
We proceed to bound the first term in LHS in (28) by defining a hypothetical arrival rate $\lambda_B = \frac{(1-\theta)(N-1)}{N}\mathbf{1}$. Denote the expectation of the service rate as $\mathbb{E}[S_i[t]] = \mu$. The first term in the RHS in (28) can be then bounded as follows:

$$\begin{aligned}
&\mathbb{E}[\langle \mathbf{Q}, \mathbf{A} + \mathbf{B} - \mathbf{S}\rangle|\mathbf{Q}] = \mathbb{E}[\langle \mathbf{Q}, \mathbf{B} - \lambda_B\rangle|\mathbf{Q}] + \mathbb{E}[\langle \mathbf{Q}, \lambda_B + \mathbf{A} - \mathbf{S}\rangle|\mathbf{Q}] \\
&= \langle \mathbf{Q}, \mathbb{E}[\mathbf{B}|\mathbf{Q}]\rangle - \langle \mathbf{Q}, \lambda_B\rangle + \langle \mathbf{Q}, \lambda_A^* + \lambda_B - \mu\rangle \\
&\overset{(a)}{\leq} (1-\theta)\Big(\sum_{i=1}^N Q_i - Q_{\max}\Big) - \sum_{i=1}^N \frac{(1-\theta)(N-1)Q_i}{N} - \frac{\delta(\epsilon,\theta)}{\sqrt{N}}\left\|\mathbf{Q}_\parallel\right\| \\
&= \frac{-(1-\theta)}{N}\sum_{i=1}^N (Q_{\max}\mathbf{1} - Q_i) - \frac{\delta(\epsilon,\theta)}{\sqrt{N}}\left\|\mathbf{Q}_\parallel\right\| \\
&= \frac{-(1-\theta)}{N}\|Q_{\max}\mathbf{1} - Q\|_1 - \frac{\delta(\epsilon,\theta)}{\sqrt{N}}\left\|\mathbf{Q}_\parallel\right\| \\
&\overset{(b)}{\leq} \frac{-(1-\theta)}{N}\|Q_{\max}\mathbf{1} - \mathbf{Q}\| - \frac{\delta(\epsilon,\theta)}{\sqrt{N}}\left\|\mathbf{Q}_\parallel\right\| \\
&\overset{(c)}{\leq} \frac{-(1-\theta)}{N}\left\|\frac{\mathbf{1}\sum_{i=1}^N Q_i}{N} - \mathbf{Q}\right\| - \frac{\delta(\epsilon,\theta)}{\sqrt{N}}\left\|\mathbf{Q}_\parallel\right\| \\
&\overset{(d)}{=} -\frac{(1-\theta)}{N}\|\mathbf{Q}_\perp\| - \frac{\delta(\epsilon,\theta)}{\sqrt{N}}\left\|\mathbf{Q}_\parallel\right\|
\end{aligned} \tag{29}$$

where the first term in (a) follows from the fact that the JS$(N-1)$Q routing policy increases the lengths of all queues by 1 except for one queue having the maximal length whenever $N-1$ requests arrive to the router which happens with probability $(1-\theta)$, the second term is by direct computation, and the third term is by the fact that $\lambda_A^* + \lambda_B - \mu = -[\frac{(1-\theta)}{N} - r\int_0^{\tau(\theta,\epsilon)} pf(p)dp]\mathbf{1} = -\frac{\delta(\epsilon,\theta)}{N}\mathbf{1}$.

(b) follows from the fact that for any vector $x \in R^N$, $\|x\|_1 \geq \|x\|$, i.e., the L^1 norm of any vector is always greater than or equal the L^2 norm. (c) follows from the fact that the average is less than or equal to the maximum. (d) is by definition of $Q_\perp$.
The next step in the proof is finding a lower bound for $\mathbb{E}[W_\parallel(Q)|Q]$. It is straightforward to show the following holds:

$$\begin{aligned}\mathbb{E}[\Delta W_\parallel(Q)|Q] =&\mathbb{E}[\langle c, Q+A+B-S+U\rangle^2 - \langle c, Q\rangle^2|Q]\\ \geq&2\langle c, Q\rangle\langle c, \lambda^* + \mathbb{E}[B|Q] - \mu\rangle - 2\mathbb{E}[\langle c, S\rangle\langle c, U\rangle]\\ \geq& -2\frac{\delta(\epsilon,\theta)}{\sqrt{N}}\left\|Q_\parallel\right\| - 2\delta(\epsilon,\theta) \end{aligned} \tag{30}$$

We can plug the bounds in (28), (29), and (30) in (25) to get:

$$\mathbb{E}[\Delta V_\perp(Q)|Q] \leq \frac{-(1-\theta)}{N} + \frac{N+1}{\|Q_\perp\|} \tag{31}$$

This inequality establishes the first condition (22) in Lemma 5.4. The second and third conditions are satisfied by boundedness assumption of arrivals and services and stability of the system, respectively. Thus, applying the conclusion of the Lemma 5.4 in (24) to $V_\perp(Q)$ concludes the proof.

5.2.4 Deriving Upper Bounds on queues. The next step in our proof is utilizing Proposition 5.5 to obtain the sum-queue lengths bound in the main Theorem. We state a Lemma from [4] applied to our predictive caching system which enables us to bound the sum queue lengths in the steady-state.

Lemma 5.7. *[4] Given the dual predictive caching routing system where with arrival* $\mathbf{A}[t]$*, artificial arrival* $\mathbf{B}[t]$*, and service* $\mathbf{S}[t]$ *vectors at time* t*, where the artificial arrivals depend on the queue lengths. Suppose* $\{\mathbf{Q}[t]\}_t$ *converges in distribution to a random vector* $\overline{\mathbf{Q}}$ *with all bounded moments, then for any positive vector* $\mathbf{c} \in \mathbb{R}_+^N$*, the following applies*

$$\mathbb{E}[\langle \mathbf{c}, \overline{\mathbf{Q}}\rangle\langle \mathbf{c}, \overline{\mathbf{S}} - \overline{\mathbf{A}} - \mathbf{B}(\overline{\mathbf{Q}})\rangle] = \frac{\mathbb{E}[\langle \mathbf{c}, \overline{\mathbf{S}} - \overline{\mathbf{A}} - \mathbf{B}(\overline{\mathbf{Q}})\rangle^2]}{2} + \frac{\mathbb{E}[\langle \mathbf{c}, \mathbf{U}(\overline{\mathbf{Q}})\rangle^2]}{2} \tag{32}$$

$$+\mathbb{E}[\langle \mathbf{c}, \overline{\mathbf{S}} - \overline{\mathbf{A}} - \mathbf{B}(\overline{\mathbf{Q}})\rangle\langle \mathbf{c}, \mathbf{U}(\overline{Q})\rangle], \tag{33}$$

where the term in (33) *can be further bounded as follows*

$$(33) \leq \sqrt{\mathbb{E}[\|\overline{\mathbf{Q}}_\perp\|^2]\mathbb{E}[U(\|\overline{\mathbf{Q}}\|^2]} \tag{34}$$

The proof of Lemma is a straightforward application of Lemma 8 and Lemma 9 (which utilizes Cauchy-Schwartz) in [4] into our system.
We can now bound the expression (32) (33) to conclude the main Theorem. We start by rewriting the LHS in (32) as follows

$$\mathbb{E}[\langle \mathbf{c}, \overline{\mathbf{Q}}\rangle\langle \mathbf{c}, \overline{\mathbf{S}} - \overline{\mathbf{A}} - \mathbf{B}(\overline{\mathbf{Q}})\rangle] = \frac{\delta(\epsilon,\theta)}{N}\mathbb{E}\left[\sum_{i=1}^N \overline{Q}_i^{(\epsilon)}\right] \tag{35}$$

Denote the first term in the RHS as $\frac{\zeta^{(\epsilon)}}{2}$. This can be calculated directly as

$$\zeta^{(\epsilon)} \triangleq \mathbb{E}[\langle \mathbf{c}, \overline{\mathbf{S}} - \overline{\mathbf{A}} - \mathbf{B}(\overline{\mathbf{Q}})\rangle^2] = \frac{1}{N}(N(\sigma_A^{(\epsilon)})^2 + \theta(1-\theta) + \delta(\epsilon,\theta)^2) \tag{36}$$

The second term in the RHS of (32) can be bounded as follows:

$$\mathbb{E}[\langle \mathbf{c}, \mathbf{U}(\overline{\mathbf{Q}})\rangle^2] \leq \langle \mathbf{c}, \mathbf{1}\rangle\mathbb{E}[\langle \mathbf{c}, \mathbf{U}(\overline{\mathbf{Q}})\rangle] \leq \delta(\epsilon,\theta) \tag{37}$$

Similarly, we can bound (33) by applying Proposition 5.5 into the bound (34) as follows:

$$\begin{aligned}(33) \leq& \sqrt{\mathbb{E}[\|\overline{\mathbf{Q}}_\perp\|^2]\mathbb{E}[\|\mathbf{U}(\overline{\mathbf{Q}})\|^2]}\\ \leq&\sqrt{\mathbb{E}[\|\overline{\mathbf{Q}}_\perp\|^2]\mathbb{E}[\langle \mathbf{1}, U(\overline{\mathbf{Q}})\rangle]} \leq \sqrt{N_2\delta(\epsilon,\theta)}\end{aligned} \tag{38}$$

Substituting (35), (36), (37), and (38) into Lemma 5.7:

$$\frac{\delta(\epsilon,\theta)}{N}\mathbb{E}\left[\sum_{i=1}^N \overline{Q}_i^{(\epsilon)}\right] \leq \frac{\zeta^{(\epsilon)}}{2} + \frac{\delta(\epsilon,\theta)}{2} + \sqrt{N_2\delta(\epsilon,\theta)} \tag{39}$$

This proves (17). Taking the limit as $\epsilon \to 0$ leads to to the expression in (19). Also taking the limit and comparing to the lower bound in Lemma 5.3 establishes HT optimality since the lower and the upper bound match asymptotically.

5.2.5 Deriving $\delta(\epsilon,\theta)$. To conclude the proof of Theorem 5.1, it remains to find the bounds characterizing $\delta(\epsilon,\theta)$. This could be obtained by rewriting $\delta(\epsilon,\theta)$ as follows:

$$\delta(\epsilon,\theta) = \mathbb{E}[S_\Sigma^*] - \mathbb{E}[A_\Sigma^*] = 1 - \theta^{(\epsilon)} - rN\int_0^{\gamma(\theta,\epsilon)} pf(p)dp \tag{40}$$

$$=1-\lambda + rN\int_{\gamma(\theta,\epsilon)}^1 pf(p)dp - \theta = \epsilon + rN\int_{\gamma(\theta,\epsilon)}^1 pf(p)dp - \theta \tag{41}$$

We use the fact that $\int_{\gamma(\theta,\epsilon)}^1 pf(p)dp \geq \int_{\gamma(\theta,\epsilon)}^1 \gamma(\theta,\epsilon)f(p)dp$ to obtain the following lower bound:

$$\delta(\epsilon,\theta) \geq \epsilon + rN\gamma(\theta,\epsilon)\int_{\gamma(\theta,\epsilon)}^1 f(p)dp - \theta \overset{(a)}{=} \epsilon + \theta(N\gamma(\theta,\epsilon) - 1) \tag{42}$$

Where (a) follows the substitution in (6). Similarly, an upper bound can be obtained for $\delta(\epsilon,\theta)$ by using the inequality $\int_{\gamma(\theta,\epsilon)}^1 pf(p)dp \leq \int_{\gamma(\theta,\epsilon)}^1 1f(p)dp$.

5.3 Closed-Form Delay-Memory Trade-off for the approximate model

The result in Theorem 5.1 illustrates the fundamental delay-memory trade-off in the predictive caching system. Intuitively, larger cache sizes at the users mean more bandwidth can be used to multicasting, which in turn implies more items can be served locally which leads to lower delay. We further illustrate that by proposing a specific approximate model.
We use the Poisson Shot Noise model that was found to empirically fit content requests well in [22] and used in the caching literature [10] to obtain an approximate request model as follows:
1. The aggregate requests from all users for a single item follows a Poisson($Nr\lambda$) distribution.
2. The parameter λ is random for every item, sampled from a Pareto distribution, i.e., $\lambda \sim \frac{\beta\alpha^\beta}{(\lambda+\alpha)^\beta}, \lambda > 0$, where α, β are the scale and shape parameter, respectively. The Pareto distribution approximates the well known Zipf distribution [14] used to model content requests in the infinite catalog regime. We use this model for its practical utility and analytic tractability, as the scaling term $\delta(\epsilon,\theta)$ in (17) can be obtained exactly in closed-form in the next corollary:

Corollary 5.8. *For the approximate Poisson-Pareto model, the PC-LQF algorithm multicasts the items with a Poisson Parameter $\lambda > \gamma(\theta, \epsilon)$, where the threshold can be quantified as follows:*

$$\gamma(\theta, \epsilon) = \alpha\left(\left(\frac{rN}{\theta}\right)^{\frac{1}{\beta}} - 1\right) \tag{43}$$

then, PC-LQF achieves the an asymptotic $(\frac{1}{\delta(\epsilon,\theta)}, \theta^{(\epsilon)})$-Queue length scaling-Memory trade-off, where $\delta(\epsilon, \theta)$, is quantified as follows

$$\delta(\epsilon, \theta) = \epsilon + Nr\alpha\left[\frac{1}{(\beta-1)(\frac{rN}{\theta})^{1-\frac{1}{\beta}}} + \frac{\alpha\left((\frac{rN}{\theta})^{\frac{1}{\beta}} - 1\right)}{\frac{rN}{\theta}}\right] - \theta \tag{44}$$

The Corollary is a straightforward application of Theorem 5.1 applied to the approximate model. Although the theorem was derived for a different model, an identical proof can be carried out with the exception that now the Poisson arrivals every slot cannot be bounded by a number $A_{\max}$ as required by Lemma 5.6, however, it was shown in [26] that the boundedness condition could be relaxed to a bound on the Moment Generating Function, which is satisfied for our Poisson arrivals, since the parameter γ clips the Poisson parameter of the arrivals at any slot.

We proceed to plot the Queue length Scaling-Memory trade-off of the approximate model in Fig. 6, with a Pareto(1,3.5) popularity distribution, and a cell with 100 users, for different values of network utilization $\rho = \frac{\mathbb{E}[A_\Sigma]}{E[S_\Sigma]}$, to further illustrate the essential dynamic in our system. We note two important observations: 1. Reduction in scaling is more significant at higher network utilization confirming the utility of our proposal in congested networks. 2. The relationship between the queue length scaling and the cache size is concave and decreasing. The decreasing part highlights that intelligent caching indeed causes continuous decrease in scaling as the cache sizes increases (as long as items being cached have expected requests higher than 1, and the unicast regime is stable). The concave part highlights diminishing returns of increasing cache sizes: A relatively small cache that can hold most of the "trending content" can offer great savings by eliminating most redundancy. Once the cache sizes increase beyond that, the BS starts multicasting less popular items that are otherwise, not widely requested by the users causing the savings to slow down. This further confirms our main message that small practical cache sizes can be very beneficial in reducing delay.

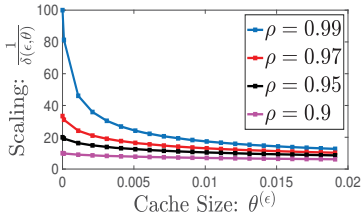
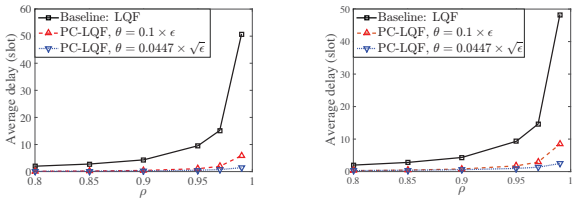


Figure 6: Scaling of (44)

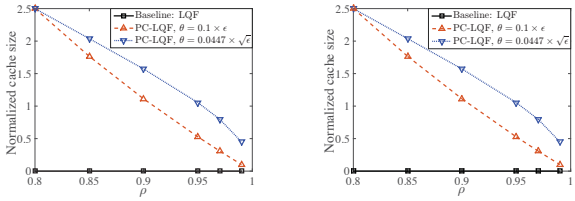
6 SIMULATIONS

We simulate a cellular downlink channel with 100 users, following our request system model. We are interested in the effect of predictive caching at the "busy hour", i.e., in a congested network. Thus



(a) Average Delay vs ρ, $f(p)$: Beta(1, 4) **(b) Average Delay vs ρ, $f(p)$: Beta(1, 9)**

Figure 7: Effect of Predictive Caching on Delay



(a) Normalized Cache Size vs ρ, $f(p)$: Beta(1, 4) **(b) Normalized Cache Size vs ρ, $f(p)$: Beta(1, 9)**

Figure 8: Normalized Cache Size supporting Predictive Caching

we define the network utilization, $\rho = \frac{\mathbb{E}[A_\Sigma]}{E[S_\Sigma]}$, and simulate cells with varying values of ρ. Note that $\rho \to 1$ is equivalent to $\epsilon \to 0$. We simulate three scenarios, the baseline unicast on-demand system, a predictive caching system with cache sizes that scale as $c_1\epsilon$, and a predictive caching system with cache sizes that scale as $c_2\sqrt{\epsilon}$, i.e., a scaling of $\omega(\epsilon)$ (as ϵ decays to 0).

In Fig. 7, we plot the average delay by varying network utilization, ρ, for two scenarios: Fig. 7 (a) for contents sampled from the popularity distribution Beta(1,4). and (b) sampled from Beta(1,9). We plot the corresponding normalized cache sizes (with respect to the user request rate, r) in Fig. 8. Fig 8 constitutes the price we pay to get delay savings in terms of cache size. Following the vanishing cache sizes assumption, the normalized cache sizes decay to zero for both the $\theta(\epsilon)$ and the $\theta(\sqrt{\epsilon})$ scaling. The first thing to note in both figures, is the vast delay reduction for predictive caching over the baseline as $\rho \to 1$, for example, as $\rho = 0.99$, predictive caching offers 10 times delay reduction for $\theta(\epsilon)$ cache sizes, and 30 times delay reduction for $\theta(\sqrt{\epsilon})$ cache sizes, which indicates the benefits of predictive caching. This comes at the cost of a normalized cache sizes of 0.1 and 0.5, roughly meaning a cache size equal to $10\% - 50\%$ of user request rate per content lifetime (often on the order of a day/few days[22]), respectively, indicating the power of a small cache to offer great delay reduction at a congested network. The second thing to notice is that the figures further solidify our intuition gained from Corollary 5.2, since the $\theta(\sqrt{\epsilon})$ offers favorable scaling that empirically alters delay asymptotics as the delay build-up is very slow compared to the other case. Finally, the discrepancy between the average delay in Fig. 7 (a) and Fig. 7 (b) points out to the effect of popularity distributions on the average delay. The reason case (a) has better delay performance is because the Beta(1,4) distribution has a **heavier tail** than the Beta(1,9) distribution. Heavier

tails imply more items with high popularity suggesting more homogeneity in user requests which increases the multicasting gains in delay saving. Thus, predictive caching exploits that commonality information which might offer a guiding principle in design that users with similar tastes should be grouped in physical or virtual cells to fully realize the benefits of predictive caching.

In Fig. 9, we plot the empirical delay-memory tradeoff by directly plotting the delay vs. the cache size for various values of ρ. The average delay value for the on-demand unicast system (equivalently when the cache sizes are zero) appears on the Y-axis. and the predictive caching average delay appear on the X-axis. The figure further highlights that a small cache can offer very significant delay reductions especially in congested networks.

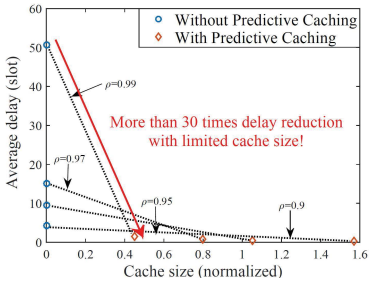


Figure 9: Empirical Delay-Cache size Trade-off

7 CONCLUSION AND FUTURE WORK

We have studied the potential of predictive caching to reduce delay at wireless cells especially at high traffic. We introduced a novel duality framework between routing and scheduling problems that we expect to be of independent interest in simplifying analysis of scheduling algorithms. We have shown that under a vanishing cache size assumption, predictive caching that utilizes multicasting alters the delay-throughput scaling as the network approaches full-load, which translates to many-fold reduction in average delay in simulations. We highlighted a fundamental delay memory trade-off in the system and characterized the correct delay scaling to obtain linear multicasting gains in the number of users. Future works include the treatment of personalized predictions where multicasting and caching can be done taking into account some information of user tastes. This combines the advances in recommender systems and online learning with the delivery problem that aims to build efficient multicasting trees. Furthermore, we aim to develop the PC-LQF scheduling algorithm to operate under non-ideal radio conditions, such as fading, where achievable rates can vary for various receivers. Choosing the correct multicasting rate becomes a non-trivial problem. We also plan to test that practical algorithm with real-life data traces using testbeds to accurately quantify the empirical effect of memory usage on delay.

Acknowledgment: This work has been supported in part by NSF grants Award CNS-1717060, CNS-1731698, CNS-1814923 and CNS-1901057, ONR grant N00014-17-1-2417, and by the IITP grant (MSIT), (2017-0-00692, Transport-aware Streaming Technique Enabling Ultra Low-Latency AR/VR Services).

REFERENCES

[1] Bengt Ahlgren, Christian Dannewitz, Claudio Imbrenda, Dirk Kutscher, and Borje Ohlman. 2012. A survey of information-centric networking. *IEEE Communications Magazine* (2012).
[2] Ying Cui and Dongdong Jiang. 2016. Analysis and optimization of caching and multicasting in large-scale cache-enabled heterogeneous wireless networks. *IEEE transactions on Wireless Communications* (2016).
[3] Jeffrey Erman and Kadangode K Ramakrishnan. 2013. Understanding the super-sized traffic of the super bowl. In *Proceedings of the 2013 conference on Internet measurement conference*. ACM.
[4] Atilla Eryilmaz and Rayadurgam Srikant. 2012. Asymptotically tight steady-state queue length bounds implied by drift conditions. *Queueing Systems* (2012).
[5] Alessandro Finamore, Marco Mellia, Zafar Gilani, Konstantina Papagiannaki, Vijay Erramilli, and Yan Grunenberger. 2013. Is there a case for mobile phone content pre-staging?. In *Proceedings of the ninth ACM conference on Emerging networking experiments and technologies*. 321–326.
[6] Cisco VNI Forecast. 2019. Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2017–2022 White Paper.
[7] Savvas Gitzenis, Georgios S Paschos, and Leandros Tassiulas. 2012. Asymptotic laws for joint content replication and delivery in wireless networks. *IEEE Transactions on Information Theory* (2012).
[8] Bruce Hajek. 1982. Hitting-time and occupation-time bounds implied by drift analysis with applications. *Advances in Applied probability* 14 (1982).
[9] Mingyue Ji, Giuseppe Caire, and Andreas F Molisch. 2015. Wireless device-to-device caching networks: Basic principles and system performance. *IEEE Journal on Selected Areas in Communications* (2015).
[10] Mathieu Leconte, Georgios Paschos, Lazaros Gkatzikis, Moez Draief, Spyridon Vassilaras, and Symeon Chouvardas. 2016. Placing dynamic content in caches with small population. In *IEEE INFOCOM*.
[11] Bin Li, Ruogu Li, and Atilla Eryilmaz. 2015. Wireless scheduling design for optimizing both service regularity and mean delay in heavy-traffic regimes. *IEEE/ACM Transactions on Networking* 24, 3 (2015), 1867–1880.
[12] Mohammad Ali Maddah-Ali and Urs Niesen. 2014. Fundamental limits of caching. *IEEE Transactions on Information Theory* (2014).
[13] Siva Theja Maguluri and R Srikant. 2016. Heavy traffic queue length behavior in a switch under the MaxWeight algorithm. *Stochastic Systems* 6, 1 (2016), 211–250.
[14] Mark EJ Newman. 2005. Power laws, Pareto distributions and Zipf's law. *Contemporary physics* (2005).
[15] Imtiaz Parvez, Ali Rahmati, Ismail Guvenc, Arif I Sarwat, and Huaiyu Dai. 2018. A survey on low latency towards 5G: RAN, core network and caching solutions. *IEEE Communications Surveys & Tutorials* (2018).
[16] Georgios Paschos, Ejder Bastug, Ingmar Land, Giuseppe Caire, and Mérouane Debbah. 2016. Wireless caching: Technical misconceptions and business barriers. *IEEE Communications Magazine* (2016).
[17] Georgios S Paschos, George Iosifidis, Meixia Tao, Don Towsley, and Giuseppe Caire. 2018. The role of caching in future communication systems and networks. *IEEE Journal on Selected Areas in Communications* (2018).
[18] Konstantinos Poularakis, George Iosifidis, Vasilis Sourlas, and Leandros Tassiulas. 2016. Exploiting caching and multicast for 5G wireless networks. *IEEE Transactions on Wireless Communications* (2016).
[19] Karthikeyan Shanmugam, Negin Golrezaei, Alexandros G Dimakis, Andreas F Molisch, and Giuseppe Caire. 2013. Femtocaching: Wireless content delivery through distributed caching helpers. *IEEE Transactions on Information Theory* (2013).
[20] Samuel O Somuyiwa, András György, and Deniz Gündüz. 2019. Multicast-Aware Proactive Caching in Wireless Networks with Deep Reinforcement Learning. In *IEEE SPAWC*.
[21] Panayotis D Sparaggis, Christos G Cassandras, and Don Towsley. 1993. On the duality between routing and scheduling systems with finite buffer space. *IEEE Trans. Automat. Control* 38, 9 (1993), 1440–1446.
[22] Stefano Traverso, Mohamed Ahmed, Michele Garetto, Paolo Giaccone, Emilio Leonardi, and Saverio Niccolini. 2013. Temporal locality in today's content caching: why it matters and how to model it. *ACM SIGCOMM Computer Communication Review* (2013).
[23] Weina Wang, Kai Zhu, Lei Ying, Jian Tan, and Li Zhang. 2016. Maptask scheduling in mapreduce with data locality: Throughput and heavy-traffic optimality. *IEEE/ACM Transactions on Networking (TON)* 24, 1 (2016), 190–203.
[24] Xiaofei Wang, Min Chen, Tarik Taleb, Adlen Ksentini, and Victor CM Leung. 2014. Cache in the air: Exploiting content caching and delivery techniques for 5G systems. *IEEE Communications Magazine* (2014).
[25] Bo Zhou, Ying Cui, and Meixia Tao. 2017. Optimal dynamic multicast scheduling for cache-enabled content-centric wireless networks. *IEEE Transactions on Communications* (2017).
[26] Xingyu Zhou, Fei Wu, Jian Tan, Kannan Srinivasan, and Ness Shroff. 2018. Degree of queue imbalance: Overcoming the limitation of heavy-traffic delay optimality in load balancing systems. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* (2018).