# Distributed Packet Forwarding and Caching Based on Stochastic Network Utility Maximization

Yitu Wang, *Student Member, IEEE*, Wei Wang, *Senior Member, IEEE*, Ying Cui, *Member, IEEE*, Kang G. Shin, *Life Fellow, IEEE, Fellow, ACM*, and Zhaoyang Zhang, *Member, IEEE*

***Abstract*— Cache-enabled network architecture has great potential for enhancing the efficiency of content distribution as well as reducing the network congestion. This, in turn, has called for joint optimization of traffic engineering and caching strategies while considering both network congestion and content demands. In this paper, we present a distributed framework for joint request/data forwarding and dynamic cache placement in cache-enabled networks. Specifically, to retrieve the information about content demands and network congestion over the network, we establish a dual queue system for both requests and data, and define a dynamic mapping between the two queues with the help of dummy data such that the nodes can determine packet forwarding and caching strategies based only on local information. As the local objective function associated with Lyapunov optimization is time-varying due to the stochastic evolution of request/data queues, we develop a low-complexity distributed forwarding and caching algorithm via stochastic network utility maximization. We also prove the proposed algorithm achieves queue stability, and derive its region stability property for time-varying local optimization to demonstrate the convergence behavior. The simulation results verify queue stability and shows the proposed algorithm outperforms the existing ones.**

***Index Terms*— Wireless communications, resource allocation, content caching, Lyapunov optimization, network utility maximization.**

## I. Introduction

THE rapid expansion of wireless services and mobile devices has led to a rapid increase of traffic load in wireless networks. To alleviate this increasing network load, the popular contents can be cached at base stations (BSs), relay stations and mobile devices, which allows users to request their desired contents from the nearby cache-enabled nodes for avoiding duplicate content transmission and reducing over-the-air traffic [1], [2]. As one of the emerging techniques for 5G communications, cache-enabled infrastructures have also been drawing considerable interest from industry [3].

Significant theoretical and experimental research has been done to show that content caching can significantly enhance the system performance. A large wireless caching network is investigated in [4] with a hierarchical tree structure of transmissions to derive the scaling results on the capacity region. Distributed caching at macro BSs is introduced in [5] to improve the network capacity. In [6], caching is adopted at BSs and mobile devices to reduce the traffic. The access delay performance is improved by caching at relay stations in cellular networks [7]. In [8], the energy consumption is minimized by appropriately caching popular contents in a proactive manner. In [9], the content caching is studied to balance the tradeoff between content dissemination delay and energy consumption. However, most of the existing work designs caching strategies according to the known content demands only without considering data-forwarding issues, such as link sharing by multiple content objects and network congestion due to heavy traffic.

Data forwarding can significantly affect the performance gain with content caching. Consider a motivating example: if a serious network congestion occurs in the transmission path of a content object from its server to its destination node, it is important to cache this content at the nodes near the destination node. Therefore, it is necessary to jointly optimize forwarding and caching, which are intrinsically coupled.

In this paper, we would like to explore distributed forwarding and caching for cache-enabled networks. Considering *mutual coupling between forwarding and caching*, there are two technical challenges.

1) **How to retrieve the network state information?** To efficiently utilize the limited link capacity and the limited cache space at the nodes, we need to retrieve the network state information, including content demands and network congestion. The content demands reveal the degree of necessity of forwarding and caching, and the network congestion implies the transmission condition. The excessive cost of flooding the information about content demands and network congestion over the entire network makes it difficult for nodes to quantitatively retrieve the information directly.
2) **How to locally optimize the performance?** The coupling between forwarding and caching makes the local optimization problem non-convex.

Manuscript received July 24, 2017; revised February 4, 2018; accepted March 26, 2018; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor C. Joo. Date of publication April 27, 2018; date of current version June 14, 2018. This work was supported in part by the National Natural Science Foundation of China under Grant 61571396 and Grant 61725104, in part by the Zhejiang Provincial Natural Science Foundation of China under Grant LR17F010001, in part by the Young Elite Scientist Sponsorship Program by CAST under Grant 2016QNRC001, and in part by the Talent Project of ZJAST under Grant 2017YCGC011. *(Corresponding author: Wei Wang.)*

Y. Wang, W. Wang, and Z. Zhang are with the Zhejiang Provincial Key Laboratory of Information Processing, Communication and Networking, College of Information Science and Electronic Engineering, Zhejiang University, Hangzhou 310027, China (e-mail: wangw@zju.edu.cn).

Y. Cui is with the Department of Electronic Engineering, Shanghai Jiao Tong University, Shanghai 200240, China.

K. G. Shin is with the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109-2121 USA.

Digital Object Identifier 10.1109/TNET.2018.2825460

Moreover, obtaining distributed solutions for the coupling cases involves iterative update and explicit message passing, making it unrealistic to assume that the wireless channel remains unchanged over the iterations. It is nontrivial to develop efficient forwarding and caching algorithms and investigate the convergence behavior of such iterative algorithms.

To meet the above challenges, we develop a distributed framework for joint request/data forwarding and dynamic caching in cache-enabled networks. The main contributions of this paper are two-fold:

1) **Dual Queue System with Dynamic Mapping.** To indirectly extract the information about content demands and network congestion over the network, we establish a dual queue system including both request/data queue sub-systems, and define a *dynamic mapping* between the request/data queues such that the nodes can retrieve the combined effect of content demands and network congestion from local queues only. Dummy data is designed to guarantee the mapping when the data is not many enough. We prove that the amount of dummy data in the network is stochastically upper-bounded, implying that the dummy data will not affect the stability of the proposed algorithm.
2) **Distributed Forwarding and Caching with Explicit Message Passing.** The stochastic evolution of request/data queues makes the local objective function associated with *Lyapunov optimization* time-varying. The distributed algorithms involve iterative solutions with explicit message passing among nodes, meaning that the objective function may change before the convergence. By *stochastic network utility maximization*, we develop a low-complexity distributed forwarding and caching algorithm. The proposed algorithm updates the request/data forwarding and cache placement strategies iteratively. We prove that the proposed algorithm achieves the queue stability and derive its region stability property in stochastic environments to demonstrate the convergence behavior.

The rest of this paper is organized as follows. Section II discusses the related work and Section III presents the system model. In Section IV, we establish a dual queue system by allowing neighbor nodes to exchange the request and data packets. In Section V, we propose the joint forwarding and caching algorithm, which can be implemented distributively with a low complexity. Following this, the performance of the proposed algorithm is evaluated in Sections VI. Finally, this paper concludes with Section VII.

## II. Related Work

This paper proposes a distributed forwarding and caching algorithm with a dual queue system to address the two technical challenges mentioned above. We briefly review the existing research on these challenges.

### A. Network State Retrieval From Local Information

Instead of retrieving the network state information directly, one commonly adopted approach is the back-pressure algorithm [11], in which the local queue information is utilized and the forwarding over a link is driven by the differences of queue lengths at transmit and receive nodes. It is demonstrated that by embracing the back-pressure algorithm, the throughput performance can be optimized such that the stability of general multi-hop queueing networks can be ensured for any arrival rate vector within the network stability region [12].

Significant research has been done on back-pressure algorithms for data forwarding, but most of it adopts the back-pressure algorithm operating on data queues. A distributed resource allocation algorithm is proposed in [13] to meet end-to-end throughput demands for multiple sessions using back-pressure data forwarding in multi-hop networks. More efficient routes are selected in [14] according to back-pressure-based forwarding to achieve desirable delay performance and avoid the waste of network resources.

In contrast with the above efforts where the data transmission is initiated by source nodes, in the cache-enabled networks, the data transmission is usually initiated by data requesters (content consumers), and the nodes which cache the contents do not maintain a long queue to provide the pressure for data forwarding. As a result, the back-pressure algorithm operating on data queues cannot account for dynamic content demands. To overcome this problem, a back-pressure algorithm operating on request queues is proposed in [15], where the local request queue information is adopted for providing the demand information for data forwarding and content caching. However, our problem is different due to the consideration of the mutual coupling between forwarding and caching, where the information about network congestion is implied in the data queues. To this end, we construct a quantitative dynamic mapping between data and request queues, such that the combined effect of network congestion and content demands can be captured from the local queue information.

### B. Joint Forwarding and Caching Optimization

In the literature, the joint routing and caching optimization in cache enabled networks has recently drawn extensive attention. In [16], a centralized joint routing and caching algorithm is proposed, where the needed content can be accessed either in one hop or two hops. In [17], collaborative caching among small-cells is mainly considered, which exploits heterogeneous cache diversity that accounts for the superiority over the non-cooperative paradigm. Different to the above two works, we address the issue of the mutual coupling between routing and caching in a multi-hop network and in a distributed manner, in which we retrieve the network state information by establishing a dual queue system with dynamic mapping, such that we are able to extract the combined effects of both content demands and network congestion locally to optimize the performance. In [18], joint caching and routing schemes with optimality guarantees for arbitrary network topologies are investigated in view of minimizing the routing cost, and takes the assumption that data follows the same path as its corresponding request, which may result in traffic congestion due to the asynchronism of the source routing.

It is very challenging to jointly optimize forwarding and caching due to the requirement on iterative explicit message passing in distributed implementation. A throughput-optimal caching and forwarding algorithm is proposed in [15] for named data networks using Lyapunov optimization, where caching and forwarding are decoupled by local optimization. In [19], taking one more step, the authors further improve the performance of the existing VIP algorithms in [15] by reflecting more accurately the actual interest packet traffic in the NDN network under interest suppression. However, the cost of this decoupling is loosening the bound of the Lyapunov drift which is not the global optimization objective, reducing the improvement of performance. A potential-based forwarding algorithm is proposed in [20] for information-centric networks, while the random caching is adopted. In [21], cooperative caching algorithms are heuristically designed without optimizing the forwarding strategy jointly.

These existing studies utilize heuristic algorithms or technical tricks to avoid the coupling between forwarding and caching, such that forwarding and caching can be optimized separately, which significantly simplifies the problem. We address this issue head-on by embracing both Lyapunov optimization and stochastic network utility maximization.

### C. Stability-Based Considerations

There are several common approaches to handle stability-based delay-aware resource allocation [22]. Large deviation [23] is an approach to convert the delay constraint into an equivalent rate constraint. However, this method achieves good delay performance only for a large delay regime. Stochastic majorization [24] provides a way to minimize the delay for the cases with symmetric arrivals. Markov decision process (MDP) [28] can minimize the delay for general cases but leads to the curse of dimensionality by brute-force value iteration.

*Lyapunov optimization* [10] is an effective approach on queue stability, which ensures that the queue system is stable as long as the average arrival rates are within the system stability region. In addition, the Lyapunov optimization has two benefits for solving the problem in our work,

1) The scheduling decision is make purely based on local information, i.e., channel conditions with the neighbor nodes and local queue backlogs. The Lyapunov drift approach that we adopt divides the queue stability into the effort of minimizing the drift in each time slot. Therefore, we are able to optimize the performance in a distributed manner.
2) Lyapunov optimization approach has a lower computational complexity to achieve the queue stability, which makes it possible to apply the proposed algorithm to very general scenarios with different request models.

By embracing Lyapunov optimization, we are trying to optimize both the delay performance (by the average queue backlog) and the power consumption, where we can adjust the Lagrangian multiplier $\nu_{ij}$ (i.e., the price of power) to optimally balance delay performance and power consumption.

TABLE I
LIST OF NOTATIONS

| | |
|---|---|
| $\mathcal{G} = (\mathcal{N}, \mathcal{L})$ | Network graph, with nodes $\mathcal{N}$ and edges $\mathcal{L}$ |
| $Z(i)$ | Neighbor nodes of node $i$ |
| $\mathbf{C}(t)$ | Time-varying channel capacity |
| $\mathcal{K}$ | Data objects |
| $S_i$ | Cache capacity at node $i$ |
| $\mathbf{b}(t)$ | Cache placements |
| $\mathbf{r}(t)$ | Data forwarding rates |
| $\mathbf{A}(t)$ | Random request arrivals |
| $\boldsymbol{\lambda}$ | Average request arrivals |
| $\Lambda$ | The capacity region |
| $\mathbf{W}(t)$ | Request queues without request forwarding |
| $\mathcal{V}(t)$ | The availability to cache the data objects |
| $\boldsymbol{\mu}(t)$ | Request forwarding rates |
| $\mathbf{Q}(t)$ | Request queues with request forwarding |
| $\mathbf{D}(t)$ | Data queues |
| $\mathbf{H}(t)$ | Dummy data queues |

## III. System Model

We introduce the architecture of cache-enabled networks and outline the key resource-allocation variables. After discussing the queue dynamics, we formulate the forwarding and caching problem from a queue stability perspective. We have summarized the notations of main symbols in Table I.

### A. Network Architecture

Consider a multi-hop cache-enabled network modeled by a directed graph $\mathcal{G} = (\mathcal{N}, \mathcal{L})$, where $\mathcal{N}$ and $\mathcal{L}$ are the sets of $N$ nodes and $L$ links, respectively. Assume that $(i, j) \in \mathcal{L}$ whenever $(j, i) \in \mathcal{L}$ due to the channel reciprocity. Let $Z(i)$ be the set of the neighbor nodes of node $i$, i.e., for any $j \in Z(i)$, $(i, j) \in \mathcal{L}$. Time is slotted and the duration of each time slot is one unit of time. The channels remain constant within a time slot and is i.i.d. over time slots. Let $C_{ij}(t)$ be the time-varying channel capacity of link $(i, j)$ at time slot $t$, and $\mathbf{C}(t) = \{C_{ij}(t), \forall (i, j) \in \mathcal{L}\}$ as the global channel capacity information[1] at time slot $t$.

The storage space at a node is consisted of two parts, namely, a CDN server[2] and a buffer. One of the typical functionalities of a CDN server is content outsourcing and distributions, i.e., to replicate and cache contents. Let $S_i \geq 0$ be the size of cache at node $i \in \mathcal{N}$. The contents in the network are represented by $K$ data objects, denoted as $\mathcal{K}$. For simplicity, we assume that all content objects have the same size[3] [16], [18], [19]. We consider the cases with limited cache sizes, i.e., $S_i < K, \forall i \in \mathcal{N}$, where no node can cache all content objects.[4] To make sure the system is stabilizable, there is a designated server node that always caches all the data objects [18]. This server node permanently stores all data objects in excess memory outside its cache.

[1]The effect of MAC can be included in the link capacity where the MAC scheme is given.

[2]In the rest of the paper, the CDN server is referred as cache for simplicity.

[3]Our work can be extended to more general cases with different content size by partitioning data objects with different size into fine-grained mini data objects with the same size.

[4]There is usually much more data than a node's cache can hold. Otherwise, each node caches all the data, making the cache placement problem trivial.

At the beginning of each slot, a node decides on two actions, including data forwarding and cache placement. The associated control variables are defined as:

- *Cache placement* $\mathbf{b}(t)$: Define $\mathbf{b}(t) = \{b_i^k(t), \forall i \in \mathcal{N}, \forall k \in \mathcal{K}\}$, where $b_i^k(t) \in \{0,1\}$ and $b_i^k(t) = 1$ represents that node $i$ caches data object $k$ in slot $t$.
- *Data forwarding rate* $\mathbf{r}(t)$ [5]: Define $\mathbf{r}(t) = \{r_{ij}^k(t), \forall i,j \in \mathcal{N}, \forall k \in \mathcal{K}\}$, where $r_{ij}^k(t)$ is the rate of data object $k$ from node $i$ to node $j$ in slot $t$.

Note that $r_{ij}^k(t) = -r_{ji}^k(t)$, where the negative rate indicates the transmission along the reverse direction of the link.

## B. Queue Dynamics and Stability

There is a bursty request arrival at each node.[6] Let $\mathbf{A}(t) = \{A_i^k(t), \forall i \in \mathcal{N}, \forall k \in \mathcal{K}\}$ be the random request arrival from the application layer. Assume that $\mathbf{A}(t)$ is i.i.d. over time slots, with $\mathbb{E}[A_i^k(t)] = \lambda_i^k$, where $\lambda_i^k$ is the average request arrival rate for data object $k$ at node $i$.

Each node has a request queue recording the unsatisfied requests, whose length is denoted as $W_i^k(t)$ for data object $k$ at user $i$ in slot $t$. Let $\mathbf{W}(t) = \{W_i^k(t), \forall i \in \mathcal{N}, \forall k \in \mathcal{K}\}$ be the global request queue information[7]. A request is met when and only when receiving a copy of the requested data object. Especially, if node $i$ determines to cache data object $k$, it generates as many copies as necessary to satisfy the requests until all the requests are satisfied, i.e., the queue $W_i^k(t)$ reaches zero. The queue dynamics of $W_i^k(t)$is

$$W_i^k(t+1) = \left(1 - b_i^k(t)\right)\left(W_i^k(t) - \sum_{j \in Z(i)} r_{ji}^k(t)\right)^+ + A_i^k(t), \tag{1}$$

where $(x)^+$ represents $\max\{x, 0\}$.

It is important to study the problem from the perspective of queue stability. Unstable queues result in infinite delay for the nodes requesting the data. It also builds up network congestion. According to [10], we define the queue stability as:

*Definition 1 (Queue Stability):* A queue $W_i^k(t)$ is strongly stable if

$$\lim_{T\to\infty} \frac{1}{T}\left(\sum_{t=0}^{T} \mathbb{E}[W_i^k(t)]\right) < \infty. \tag{2}$$

The system is said to be stable if all the queues in the system are strongly stable. ∎

Furthermore, to ensure that the system is stabilizable, we define the capacity region following [10] as:

*Definition 2 (Capacity Region):* The capacity region $\Lambda$ is defined as the closure of the set of all input rate vectors $\boldsymbol{\lambda}$ stabilizable under some rate allocation algorithm that conforms to the transmission capability constraint $\sum_{k\in\mathcal{K}} r_{ij}^k(t) \le C_{ij}(t)$. ∎

Throughout this paper, we assume that the input rate vector $\boldsymbol{\lambda}$ is strictly interior to the capacity region, such that the system is stabilizable.

## C. Problem Formulation

Our goal is to stabilize the system by optimizing forwarding and caching strategies. For any $\boldsymbol{\lambda} \in \Lambda$, the cache placement $\mathbf{b}(t)$ and data forwarding rate $\mathbf{r}(t)$ should satisfy the following conditions:

$$W_i^k(t), \quad \forall i \in \mathcal{N},\ \forall k \in \mathcal{K}, \text{are strongly stable}, \tag{3}$$
$$\sum_{k\in\mathcal{K}} r_{ij}^k(t) \le C_{ij}(t), \quad \forall (i,j) \in \mathcal{L}, \tag{4}$$
$$\sum_{k\in\mathcal{K}} b_i^k(t) \le S_i, \quad \forall i \in \mathcal{N}, \tag{5}$$
$$b_i^k(t) \in \mathcal{V}_i^k(t), \quad \forall i \in \mathcal{N},\ \forall k \in \mathcal{K}, \tag{6}$$

where (3) implies the stability constraint, (4) implies the limitation on link capability, (5) implies the limitation on cache size and (6) implies that the caching decision of a node is based on an action set $\mathcal{V}(t)$, which is determined from the currently received data objects, i.e., $\mathcal{V}_i^k(t) = \{0,1\}$ if data object $k$ is available to cache at node $i$, and $\mathcal{V}_i^k(t) = \{0\}$ if data object $k$ is not available.

Without prior knowledge of traffic arrival rates, which are generally not known to the controller or users, it is difficult to directly judge whether the stability constraints (3) are satisfied. By rewriting (3) as the queue stability condition in Definition 1, we will design a distributed forwarding and caching algorithm, observing the local queue information to fulfill the stability constraint and the above limitations by establishing a dual queue system and optimizing the forwarding and caching strategies locally.

## IV. Dual Queue System With Dynamic Mapping

The current request queue $\mathbf{W}(t)$ cannot capture the network state information because it contains only the requests of the node itself. To retrieve the network state information, including content demands and network congestion, we allow the nodes to forward the requests such that the local queue information can capture this network state information.

In this section, we will address the first technical challenge by establishing a dual queue system including request/data queue sub-systems. Furthermore, we will define the dynamic mapping between both request/data queues and present the queue dynamics in this new dual queue system.

## A. Request Forwarding

To make the content demands extractable from the local queue information, we allow the nodes to forward the requests, such that the request queue of a node contains not only the requests generated by itself but also those from other nodes in the network. We introduce a new control variable for this:

- *Request forwarding rate* $\boldsymbol{\mu}(t)$: Define $\boldsymbol{\mu}(t) = \{\mu_{ij}^k(t), \forall i,j \in \mathcal{N}, \forall k \in \mathcal{K}\}$, where $\mu_{ij}^k(t)$ is the

[5]Since buffers only temporarily cache the data without the capability of data replication, if a node have several requests for the content that is not cached in the CDN server, it needs to receive the same amount of copies to satisfy the requests due to the heterogeneity of the storage space.

[6]Similarly to [15], node $i$ is considered as a point of aggregation which combines many network users. Such an aggregation point is likely to submit many requests for a given data object over time.

[7]$\mathbf{W}(t)$ cannot capture the network state information since it contains only the requests of the node itself. In section IV, we propose a dual queue system to address this issue.

request forwarding rate of data object $k$ from node $i$ to node $j$ in slot $t$.

*Remark 1 (Physical Meaning of Request Forwarding): The request queues can be interpreted as the potential. For any data object, there is a downward gradient from the nodes with longer queues to those with shorter ones, which is similar to load balancing with the back-pressure algorithm.* ■

With requests forwarding, we establish a dual queue system, which includes request queues $\mathbf{Q}(t)$ and data queues $\mathbf{D}(t)$, where $Q_i^k(t)$ and $D_i^k(t)$ are the request and data queues, respectively, for data object $k$ at user $i$ in slot $t$.

## B. Dynamic Mapping Between Data and Request Queues

Besides the content demands, the network congestion information is also important. It is possible to extract the network congestion information from data queues, while the content demands are captured by request queues. For extraction of the combined effect of both content demands and network congestion, we define a dynamic mapping between data and request queues.

The concept of this dynamic mapping is, when the requests are forwarded from node $i$ to its neighbor node $j$, the same amount of data should be transmitted over the reverse link, i.e., from node $j$ to node $i$. However, it is nontrivial to achieve this from time to time, because the data queue of node $j$ may not always be long enough for transmission. To render this dynamic mapping valid, we introduce the concept of dummy data. Let $\mathbf{H}(t)$ denote the dummy data queue, where $H_i^k(t)$ is the dummy data queue for data object $k$ at user $i$ in slot $t$.

With the dynamic mapping between data and request queues[8], we can determine data forwarding rate $r_{ij}^k(t)$ indirectly by controlling the request forwarding rate $\mu_{ij}^k(t)$. However, $r_{ij}^k(t)$ cannot be determined simply as $r_{ij}^k(t) = -\mu_{ij}^k(t)$, because the data queue is not always long enough for the transmission on the reverse link. Let $\gamma_{ij}^k(t)$ denote the dummy data rate for data object $k$ on link $(i, j)$. We need to discuss the values of $\mu_{ij}(t)$, $Q_i^k(t)$ and $D_j^k(t)$ before determining $r_{ij}^k(t)$ and $\gamma_{ij}^k(t)$. Here, we assume that the requests are sent from node $i$ to node $j$, i.e., $\mu_{ij}^k(t) > 0$ and $\mu_{ij}^k(t) \leq Q_i^k(t)$,

- **The data queue is long enough,** $\mu_{ij}^k(t) \leq D_j^k(t)$: The request rate equals the data rate over the reverse link $r_{ij}^k(t) = -\mu_{ij}^k(t)$ without transmitting the dummy data, i.e., $\gamma_{ij}^k(t) = 0$.
- **The data queue is not long enough,** $\mu_{ij}^k(t) > D_j^k(t)$: The request rate equals the data rate plus the dummy data rate $r_{ij}^k(t) + \gamma_{ij}^k(t) = -\mu_{ij}^k(t)$ with node $j$ generates $\gamma_{ji}^k(t) = \mu_{ij}^k(t) - D_j^k(t)$ dummy data.

Note that the dummy data are generated to balance the ideal mapping, which are not actually transmitted over-the-air.[9]

[8]It is not necessary for the dynamic mapping that the data forwarding rate is equal to the request forwarding rate, since some systems may have the capability to copy the data packets. Our proposed dual queue system can be extended to the cases where the relationship between the data forwarding rate and the queue length is determinate by adjusting the dynamic mapping between data and request queues.

[9]We will later prove that the number of dummy data is stochastically upper-bounded, which will not affect the stability of the queuing system.

Besides the transmission between nodes, the dynamic mapping between data and request queues also exists when the packets enter and exit the network. The requests are removed from the system when they are forwarded to a node which caches the requested data object, while the same amount of data is generated by this node. The requests are added into the system while the data or dummy data are removed from the system with a total rate $A_i^k(t)$. One major difference between actual and dummy data is that when dummy data are received by the node requesting the data, extra requests will be added to the system because the requests are not actually met.

## C. Queue Dynamics in Dual Queue System

*1) Dynamics of the Request Queues:* In the request queue sub-system, using the request forwarding rate $\mu_{ij}^k(t)$ instead of the data forwarding rate $r_{ij}^k(t)$ in (1) due to the mapping between $\mathbf{r}(t)$ and $\boldsymbol{\mu}(t)$, we obtain the queue dynamics of the request queue $Q_i^k(t)$ as

$$Q_i^k(t+1) = \left(1 - b_i^k(t)\right)\left(Q_i^k(t) - \sum_{j\in Z(i)} \mu_{ij}^k(t)\right)^+ + A_i^k(t) + \min\{H_i^k(t), A_i^k(t)\}. \quad (7)$$

where $\left(Q_i^k(t) - \sum_{j\in Z(i)} \mu_{ij}^k(t)\right)^+$ in the first term represents request forwarding with neighbor nodes, $\left(1 - b_i^k(t)\right)$ in the first term represents if node $i$ determines to cache data object $k$, it generates as many copies as necessary to satisfy the requests until all the requests are satisfied, and the last term $\min\{H_i^k(t), A_i^k(t)\}$ is the extra requests which are added due to the receipt of dummy data.

The data queue sub-system consists of two closely related queues, namely the data queue and the dummy data queue. The dummy data are generated to balance the dynamic mapping between data and request queues when the data queue is not long enough.

*2) Dynamics of the Data Queues:* The dynamics of the data queue $D_i^k(t)$ satisfies

$$D_i^k(t+1) = \left(D_i^k(t) - \sum_{j\in Z(i)} r_{ij}^k(t) - \left(A_i^k(t) - H_i^k(t)\right)^+ + b_i^k(t) \sum_{j\in Z(i)} \left(r_{ij}^k(t)\right)^+\right)^+. \quad (8)$$

Data generation: Data can only be generated at the node caching the specific data, hence the request queue of this node is empty. While the request queues of the neighbor nodes are non-empty, the neighbor nodes will forward requests to this node. Then, the received requests move out of the system, and the same amount of data is generated and transmitted over the reverse link. This procedure accounts for the second term and the last term.

Data consumption: Upon data arrival, the data are used to satisfy the local request first, i.e., requests and data cancel out when meeting. The rest of the data are then stored in the data queue waiting forwarding. This procedure accounts for the third term.

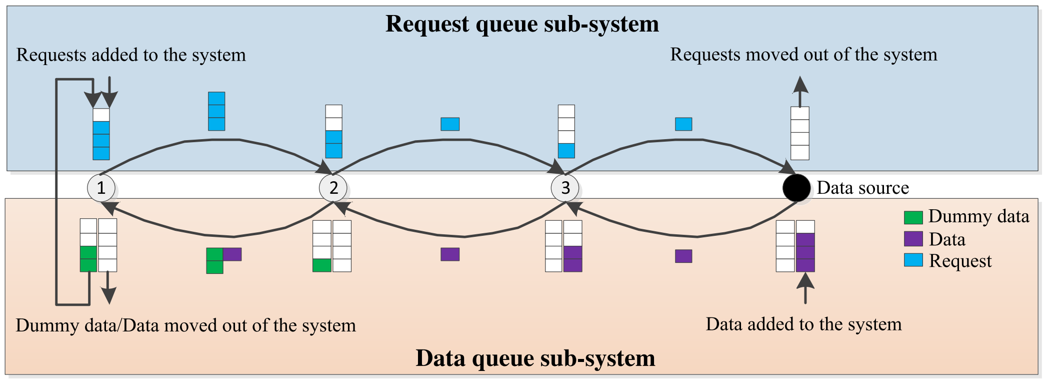


Fig. 1. Illustration of the dual queue system.

*3) Dynamics of the Dummy Data Queues:*

$$H_i^k(t+1) = \left(H_i^k(t) - \sum_{j\in Z(i)} \gamma_{ij}^k(t) - A_i^k(t) + \left(1-b_i^k(t)\right)\left(\sum_{j\in Z(i)} \mu_{ji}^k(t) - H_i^k(t) - D_i^k(t)\right)^+\right)^+. \quad (9)$$

Dummy data generation: Dummy data are generated at a node when the sum of request rates from its neighbors is larger than the sum of the dummy data queue length and the data queue length. This procedure accounts for the second term and the last term.

Dummy data consumption: This procedure is similar to that of data consumption. Upon the arrival of dummy data, the dummy data are used to satisfy the local request first and the remaining dummy data are put into the dummy data queue. Extra requests will be added if dummy data are consumed, which accounts for the third term.

Fig. 1 illustrates how the dual queue system works. The dual queue system consists of two sub-systems, namely the request queue sub-system and the data queue sub-system. There is a downward gradient from node 1 to the data source, which is similar to load balancing through the back-pressure algorithm. Unlike the conventional back-pressure algorithm operating on data queues, we operate on request queues to consider dynamic content demands. The longer the request queue becomes, the more urgent the node needs the data. The growth of node 1's request queue leads to the growth of those of nodes 2 and 3, which results in a larger amount of data transmitted over the reverse link due to the dynamic mapping between data and request queues. The dummy data is generated to balance the ideal mapping and is not actually transmitted over-the-air. The satisfied dummy data needs to be added to the system again because the requests are not actually met. The dummy data are generated to establish the system, and the correspondent excessive data stay in the system for the dynamic mapping, which are the cornerstone of the dual queue system.

## V. Distributed Forwarding and Caching Algorithm

In the dual queue system, nodes forward the requests and are equipped with a quantitative relationship with data queues such that they can obtain network state information for the forwarding and caching optimization based on local request queues $Q_i^k(t)$. The key obstacle in deriving a distributed forwarding and caching algorithm is the iteration with explicit message passing in a stochastic environment, which is due to the coupling between forwarding and caching.

Here we address the second technical challenge by decomposing the Lyapunov drift optimization problem via stochastic network utility maximization. Two decomposed subproblems, including forwarding and caching optimizations, are solved iteratively. Furthermore, we analyze the queue stability property and the region stability property for time-varying local optimization to demonstrate the convergence behavior.

### *A. Stability-Driven Optimization*

To meet the stability constraints, we adopt the Lyapunov optimization method with a commonly-used quadratic Lyapunov function [10], which increases quadratically with the queue length and can provide a large enough penalty to stabilize the system. The Lyapunov function is

$$L\left(Q_i^k(t)\right) = \sum_{i\in\mathcal{N}} \sum_{k\in\mathcal{K}} \left(Q_i^k(t)\right)^2. \quad (10)$$

According to Definition 1, all queues should be stabilized for system stability. Thus, we formulate the associated Lyapunov drift optimization problem as

$$\max_{\boldsymbol{\mu}(t),\mathbf{b}(t)} Y(\boldsymbol{\mu}(t),\mathbf{b}(t)) = \sum_{i\in\mathcal{N}} \sum_{k\in\mathcal{K}} \left(2\left(1-b_i^k(t)\right)Q_i^k(t) \sum_{j\in Z(i)} \mu_{ij}^k(t) + b_i^k(t)\left(Q_i^k(t)\right)^2 + b_i^k(t)\left(\sum_{j\in Z(i)} \mu_{ij}^k(t)\right)^2\right). \quad (11)$$

See Appendix A for the detailed derivation of the above Lyapunov drift optimization problem.

Note that the optimization problem (11) is a non-convex and non-monotonic problem even after relaxing the feasible region of $\mathbf{b}(t)$ to $[0,1]$. It is very difficult to obtain a closed-form solution and extract insights on the design of forwarding and caching. This is because the queues are coupled with each other due to the coupling constraints of the request forwarding rate $\boldsymbol{\mu}(t)$ and the caching placement $\mathbf{b}(t)$, which stem from the limitations of link capacity and cache size.

## B. Decomposed Forwarding and Caching Optimization

To derive a distributed algorithm, we decompose the optimization problem in (11) into subproblems for each node and each link by using a similar approach to network utility maximization. However, unlike conventional network utility maximization, we consider its iterative solution in a stochastic environment. Specifically, the stochastic evolution of the request queue $Q_i^k(t)$ and the link capacity $C_{ij}(t)$ yield the time-varying property of the optimal value.

By rewriting the optimization problem in (11) in the view of nodes, we decompose it into the caching subproblems for each node to optimize $\mathbf{b}(t)$ with given $\boldsymbol{\mu}(t)$. Similarly, by rewriting this optimization problem in the view of links, we decompose it into the forwarding subproblems for each link to optimize $\boldsymbol{\mu}(t)$ with given $\mathbf{b}(t)$. Following the above discussion, next we design and present a distributed algorithm.

*1) Caching Optimization:* Rewrite (11) in a form that can be decomposed by nodes:

$$\begin{aligned}\max_{\mathbf{b}(t)} \ & \sum_{i\in\mathcal{N}}\sum_{k\in\mathcal{K}} 2Q_i^k(t)\sum_{j\in Z(i)}\mu_{ij}^k(t) \\ & + \sum_{i\in\mathcal{N}}\sum_{k\in\mathcal{K}} b_i^k(t)\left(Q_i^k(t)-\sum_{j\in Z(i)}\mu_{ij}^k(t)\right)^2. \\ s.t. \ & \sum_{k\in\mathcal{K}} b_i^k(t)\le S_i, \quad \forall i\in\mathcal{N}. \end{aligned} \tag{12}$$

*Theorem 1 (Optimal Cache Replacement Algorithm): Given request forwarding rate $\boldsymbol{\mu}(t)$, for the optimality of the caching objective* (12)*, the caching priority function is*

$$f_i^k(t)=\left(Q_i^k(t)-\sum_{j\in Z(i)}\mu_{ij}^k(t)\right)^2. \tag{13}$$

*In the optimal caching strategy, the data objects with the highest $S_i$ caching priorities are stored at node $i$.*

*Proof:* By taking a partial derivative of (12) with respect to $b_i^k(t)$ for each $i$ and $k$, we obtain the caching priority function (13).

Since (12) is linear in $b_i^k(t)$, considering the limitation on cache size in (5), the data objects with the highest $S_i$ caching priorities should be stored for optimality. ■

*Remark 2: (Interpretation of Theorem 1): The request queue length represents the urgency of requiring the data object. The longer the request queue length is, the more urgent the data object should be cached. When the requests are forwarded to neighbor nodes, the data are sent back on the reverse path at the same rate, thus lowering the urgency.*

*2) Forwarding Optimization:* Rewrite (11) in a form that is decomposable by links,

$$\begin{aligned}\max_{\boldsymbol{\mu}(t)} \ & \sum_{(i,j)\in\mathcal{L}}\sum_{k\in\mathcal{K}} 2\mu_{ij}^k(t)\big(Q_i^k(t)\big(1-b_i^k(t)\big)-Q_j^k(t)\big(1-b_j^k(t)\big)\big) \\ & + \sum_{i\in\mathcal{N}}\sum_{k\in\mathcal{K}}\left(b_i^k(t)\big(Q_i^k(t)\big)^2\right) \\ s.t. \ & \sum_{k\in\mathcal{K}}\mu_{ij}^k\le C_{ij}(t), \quad \forall (i,j)\in\mathcal{L}. \end{aligned} \tag{14}$$

*Theorem 2 (Optimal Request Forwarding Algorithm): Given a caching placement vector $\mathbf{b}(t)$, for the optimality of the forwarding objective* (14)*, the request rate allocated for data object $k$ over link $(i,j)$ is*

$$\mu_{ij}^k(t)=\frac{1}{2}\big(\omega_{ij}^k(t)-\omega_{ij}^*(t)\big)^+, \tag{15}$$

*where $\omega_{ij}^*(t)$ satisfies*

$$\sum_{k\in\mathcal{K}}\frac{1}{2}\big(\omega_{ij}^k(t)-\omega_{ij}^*(t)\big)^+=C_{ij}(t), \tag{16}$$

*and $\omega_{ij}^k(t)=Q_i^k(t)\big(1-b_i^k(t)\big)-Q_j^k(t)\big(1-b_j^k(t)\big)$ is a weight depending on the difference between the request queues of the two end nodes of link $(i,j)$ for data object $k$.*

*Proof:* Taking partial the derivative of (14) with respect to $\mu_{ij}^k(t)$ for each link $(i,j)$ and object $k$ using the Lagrange method, we obtain

$$\frac{\partial Y(\boldsymbol{\mu}(t))}{\partial \mu_{ij}^k(t)}=2\omega_{ij}^k(t)-\nu_{ij}, \tag{17}$$

where $\nu_{ij}$ is the Lagrangian multiplier which is adjusted to satisfy the link capacity constraint (14).

Instead of scheduling the requests for only one data object as the conventional back-pressure algorithm, we schedule request packets at a finer granularity for multiple data objects for transmission in the same time slot. This is to achieve better performance. One request for the data object with the largest $\omega_{ij}^k(t)$ will be scheduled according to (17) and $\omega_{ij}^k(t)$ decreases with the transmission. Executing such an algorithm iteratively finds the optimal solution to (14), where the requests for the data objects which satisfy $\omega_{ij}^k(t)>\omega_{ij}^*(t)$ are scheduled with a request rate that is a half of the backlog difference, i.e., $\mu_{ij}^k(t)=\frac{1}{2}\big(\omega_{ij}^k(t)-\omega_{ij}^*(t)\big)^+$. The total transmission rate is limited by the constraint (14), which provides a convenient way of determining $\omega_{ij}^*(t)$ by substituting (15) into (14). ■

*Remark 3 (Interpretation of Theorem 2): This back-pressure-based forwarding algorithm allocates a request rate for each data object and each link, which is different from the back-pressure scheduling that schedules only one user. Specifically, the larger $\omega_{ij}^k(t)$ is, the larger request rate is allocated. $\omega_{ij}^k(t)$ is related to the queue difference, the back-pressure approach maximally balances the queues to avoid too long request queues in some part of the network.* ■

Note that both the back-pressure-based forwarding algorithm and the max-weight caching algorithm are distributed algorithms. To implement such a forwarding algorithm, neighbor nodes need to exchange their queue length information periodically via explicit message passing. The implementation of such a caching algorithm only needs local decisions without any information exchange.

## C. Algorithm Design

Based on the decomposed forwarding and caching optimization, we propose an algorithm to solve the two types of subproblems. We adopt a modified order of updates: Gauss-Seidel algorithm [28], which iteratively adjusts only one variable to its optimal value while keeping the other variables intact, i.e.,

$$x_i(t+1) = \arg\max_{x_i} f\bigg(x_1(t+1), \cdots, x_{i-1}(t+1), x_i, x_{i+1}(t), \cdots, x_n(t)\bigg),$$

where $x_1, \cdots, x_n$ represent the optimization variables.

Note that the variables can be updated at only one iteration in each time slot due to explicit message passing, while the objective function will change due to the stochastic evolution of $Q_i^k(t)$ and $C_{ij}(t)$ in the next time slot.[10]

Algorithm 1 provides the details of this using pseudo codes, which is launched at the beginning of each time slot.

**Algorithm 1** Distributed Forwarding and Caching Algorithm

```
1: loop
2:     Each node $i$ observes channel capacity $C_{ij}(t), j \in Z(i)$.
3:     Each node $i$ receives the information about the request queue length $Q_j^k(t)$ and the cache status $b_j^k(t)$ of its neighbors $\forall j \in Z(i), \forall k \in \mathcal{K}$.
4:     Allocate the request rate $\boldsymbol{\mu}(t)$ according to (15) in Theorem 2.
5:     The data rate $\mathbf{r}(t)$ and the dummy data rate $\boldsymbol{\gamma}(t)$ are obtained according to $\boldsymbol{\mu}(t)$, $\mathbf{D}(t)$ and $\mathbf{C}(t)$.
6:     Each node $i$ observes current request rates of connected links $\mu_{ij}^k(t), j \in Z(i)$ at $t$.
7:     Compute the caching priority function according to (12).
8:     The data objects with the largest $S_i$ caching priorities are stored at node $i$ at $t$ according to Theorem 1.
9:     The queues are updated according to (7), (8), and (9).
10: end loop
```

In Algorithm 1, lines 3–5 optimize the request rate $\boldsymbol{\mu}(t)$ and then obtain the data/dummy data rate, and lines 6–8 optimize the caching strategy $\mathbf{b}(t)$. The request forwarding rate $\boldsymbol{\mu}(t)$ and cache placement $\mathbf{b}(t)$ are updated online and iteratively.

*Remark 4 (Computational Complexity of Algorithm 1): The most time-consuming part of Algorithm 1 is the ordering in line 8, which could be solved using a quick sort with the running time $O(K \log_2 K)$. The optimal request forwarding rates in Line 5 can be found using the gradient method, whose complexity is low, and the other parts can be calculated in closed-form, whose computational complexity are low.* ■

## D. Performance Evaluation

In this subsection, we first demonstrate the convergence behavior of the proposed algorithm. Then, we prove that the proposed algorithm achieves queue stability, such that the system is stabilized. Finally, the performance overhead brought by allowing the nodes to forward the requests is analysed, including the queue system overhead and the communication overhead.

[10]Although the result after one iteration is suboptimal, such a solution actually track the trajectory of optimal values in a stochastic evolutionary environment, which will be proved later.

The stochastic evolution of request/data queues makes the local objective function associated with Lyapunov optimization time-varying. The distributed algorithms involve iterative solutions with explicit message passing among nodes, meaning that the objective function may change before the iterative algorithms converge, which indicates that the optimal point is also changing and hence, the algorithm trajectory will not converge to a single point but rather a limit region. Therefore, to analyse the convergence property of the proposed algorithm in time-varying environments, we invest the region stability property of the proposed algorithm alternatively, which is a widely adopted performance measure of iterative algorithms in time-varying environments, especially for hybrid systems [26], [27]. The region stability is defined as follows.

*Definition 3 (Region Stability):* A discrete-time system with state vector $\mathbf{x}(t)$ is said to be *stable* w.r.t. a limit region $\mathcal{X}$, if for every trajectory $\mathbf{x}(t, \mathbf{x}(0))$, there exists a time instant $T(\mathbf{x}(0))$ such that from then on, the trajectory is always within the limit region $\mathcal{X}$. Mathematically, $\forall \mathbf{x}(t, \mathbf{x}(0))$, $\exists T(\mathbf{x}(0))$, such that $\mathbf{x}(t, \mathbf{x}(0)) \in \mathcal{X}$, $\forall t \geq T(\mathbf{x}(0))$. ■

To analyze the tracking behavior of the proposed algorithm, we first consider the case when the request queue lengths and the channel capacity remain unchanged within a time slot. Let $Y^*(t)$ denote the optimal value of (11) in slot $t$. In slot $t$, the proposed iterative algorithm executes the iteration once. We introduce the contraction property of the proposed algorithm,

$$Y^*(t) - Y^1(t) \leq \beta(t)(Y^*(t) - Y^0(t)), \tag{18}$$

where $\beta(t)$ is referred to as the *contraction modulus* of an iteration in slot $t$, $Y^0(t)$ ($Y^1(t)$) represents the value before (after) performing the iteration in slot $t$. Note that the contraction modulus satisfies $\beta(t) < 1$, because the proposed algorithms try to reduce the difference among the request queues at the nodes, and hence the quadratic Lyapunov drift decreases, i.e., the objective function increases $Y^1(t) > Y^0(t)$.

Based on the above analysis, we derive the region stability of the proposed algorithm in the following theorem.

*Theorem 3 (Region Stability):* There exists a time instant $T$ such that for $t > T$, the trajectory of the objective function (11) is always within a limit region

$$\mathcal{X} = \left\{ x | x \leq \max_{t>T} Y^*(t) + \delta \frac{\beta}{1-\beta} + \Delta \right\}, \tag{19}$$

where $\Delta$ can be chosen arbitrarily small, $\delta = \max_{m>T} \delta_{m-1,m} = \max_{m>T} |Y^*(m) - Y^*(m-1)|$ denotes the maximum distance between the target optimal solution of one slot and that of the next slot, $\beta = \max_{m>T} \beta(m)$ represents the minimum performance improvement per time slot.

*Proof:* Please refer to Appendix B. ■

*Remark 5 (Interpretation of Theorem 3):* The trajectory of the objective function is always within a limit region $\mathcal{X}$, which indicates a bounded tracking error. ■

Next, we prove that the proposed algorithm achieves queue stability in the following theorem, such that the system is stabilized.

*Theorem 4 (Queue Stability):* If the performance under the proposed algorithm is $\frac{1}{1+\alpha}$ of that under the optimal algorithm, then the capacity region will shrink by $\frac{1}{1+\alpha}\lambda_{\max}\mathbf{I}$, and the corresponding average queue lengths satisfy

$$\lim_{T\to\infty}\frac{1}{T}\left(\sum_{t=0}^{T}\sum_{i\in\mathcal{N}}\sum_{k\in\mathcal{K}}D_i^k(t)\right)\le\lim_{T\to\infty}\frac{1}{T}\left(\sum_{t=0}^{T}\sum_{i\in\mathcal{N}}\sum_{k\in\mathcal{K}}Q_i^k(t)\right)\le\frac{(1+\alpha)B}{2(\epsilon-\frac{1}{1+\alpha}\lambda_{\max})},\tag{20}$$

$$\lim_{T\to\infty}\frac{1}{T}\left(\sum_{t=0}^{T}\sum_{i\in\mathcal{N}}\sum_{k\in\mathcal{K}}W_i^k(t)\right)\le\frac{(1+\alpha)B}{2(\epsilon-\frac{1}{1+\alpha}\lambda_{\max})},\tag{21}$$

*Proof:* Please refer to Appendix C. ■

*Remark 6 (Interpretation of Theorem 4):* For any input rate vector $\boldsymbol{\lambda}$ interior to the capacity region $\Lambda-\frac{1}{1+\alpha}\lambda_{\max}\mathbf{I}$, all queues in the system can be stabilized by Theorem 4. ■

Finally, the performance overhead brought by allowing the nodes to forward the requests is analysed, including the queue system overhead and the communication overhead.

*Queue System Overhead:* To render the dynamic mapping valid, we introduce the concept of dummy data, which in return brings the extra overhead to the queue system. In the following, we analyze the amount of dummy data in the following theorem to show that the amount of dummy data is bounded in probability and adding dummy data will not affect the stability of the proposed algorithm.

*Theorem 5 (Bound of the Generated Dummy Data): The total amount of the generated dummy data in the network satisfies*

$$\Pr\left(\left|\sum_{i\in\mathcal{N}}\sum_{t=0}^{T}H_i^k(t)-\frac{(1+\alpha)B}{(\epsilon-\frac{1}{1+\alpha}\lambda_{\max})}\right|\ge\nu\right)\le\frac{2var\left(\sum_{i\in\mathcal{N}}A_i^k(t)\right)+2var\left(\sum_{i\in\mathcal{L}}C_{ji}(t)\right)}{\nu^2},\tag{22}$$

*where var represents the variance,* $var\left(\sum_{i\in\mathcal{N}}A_i^k(t)\right)$ *and* $var\left(\sum_{i\in\mathcal{L}}C_{ji}(t)\right)$ *can be calculated according to their corresponding probability density functions.*

*Proof:* Please refer to Appendix D. ■

*Remark 7 (Stability with Dummy Data): According to Theorem 5, when the amount of the dummy data exceeds a threshold, the probability of adding more dummy data to the system declines by $O(1/\nu^2)$, where $\nu$ is the distance between the amount of the dummy data and the threshold. Therefore, the probability of the amount of the dummy data becoming infinity approaches 0, suggesting that the average amount of the dummy data is stochastically upper-bounded and adding dummy data does not affect the system stability. However, the generated dummy data induces the same amount of data eventually, which always stay in the system to render the dynamic mapping valid in the dual queue system.* ■

*Communication Overhead:* The communication overhead will slightly increase by allowing the nodes to forward the requests. To clearly describe such influence, suppose the length of the request queue of node $i$ is larger than that of the neighbor nodes, and the length of the frame is $T$. The signaling procedure includes three stages:

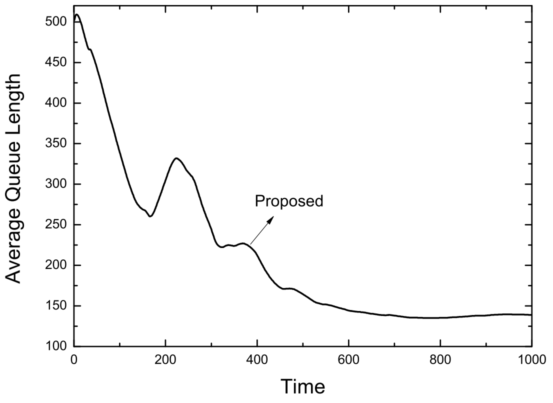


Fig. 2. Stability of request queue length.

- Node $i$ broadcasts the queue information to its neighbor nodes, which takes $T_{sig}$ amount of time.
- Node $i$ forwards the requests to its neighbor nodes separately, which takes at most $HT_{sig}$ amount of time, where $H$ is the maximum degree of the network graph.
- Node $j$ transmits data/dummy data to node $i$, which takes $T_{data}$ amount of time.

Due to the added request forwarding time, we will suffer a slight performance degradation. Specifically, the influence brought by the request forwarding time $T_{sig}$ is given in the following theorem.

*Theorem 6 (Influence of the Communication Overhead): Suppose the input rate vector $\boldsymbol{\lambda}$ satisfies $\left(1+\frac{(H+1)T_{sig}}{T}\right)\boldsymbol{\lambda}$ is interior to the capacity region $\Lambda$, where $T$ is the length of the frame, $T_{sig}$ is the communication time. For any $T_{sig}$ satisfying $0<\left(\frac{(H+1)T_{sig}}{T}\right)\lambda_{max}\le\epsilon$, where $\lambda_{max}=\max_{i,k}\{\lambda_i^k\}$, the proposed caching and forwarding algorithms stabilize all queues of the system. If the performance under the proposed algorithm is $\frac{1}{1+\alpha}$ of that under the optimal algorithm, then the capacity region will shrink by $\frac{1}{1+\alpha}\lambda_{\max}\mathbf{I}$, and the corresponding average queue lengths satisfy*

$$\lim_{T\to\infty}\frac{1}{T}\left(\sum_{t=0}^{T}\sum_{i\in\mathcal{N}}\sum_{k\in\mathcal{K}}D_i^k(t)\right)\le\lim_{T\to\infty}\frac{1}{T}\left(\sum_{t=0}^{T}\sum_{i\in\mathcal{N}}\sum_{k\in\mathcal{K}}Q_i^k(t)\right)\le\frac{(1+\alpha)B}{2(\epsilon-\frac{1}{1+\alpha}\lambda_{\max}-\left(\frac{(H+1)T_{sig}}{T}\right)\lambda_{max})},\tag{23}$$

$$\lim_{T\to\infty}\frac{1}{T}\left(\sum_{t=0}^{T}\sum_{i\in\mathcal{N}}\sum_{k\in\mathcal{K}}W_i^k(t)\right)\le\frac{(1+\alpha)B}{2(\epsilon-\frac{1}{1+\alpha}\lambda_{\max}-\left(\frac{(H+1)T_{sig}}{T}\right)\lambda_{max})}.\tag{24}$$

Note that the performance loss due to exchanging queues information is suffered by almost all queue-based algorithms.

## VI. Simulation

We now evaluate the performance of the proposed distributed forwarding and caching algorithm by simulation.

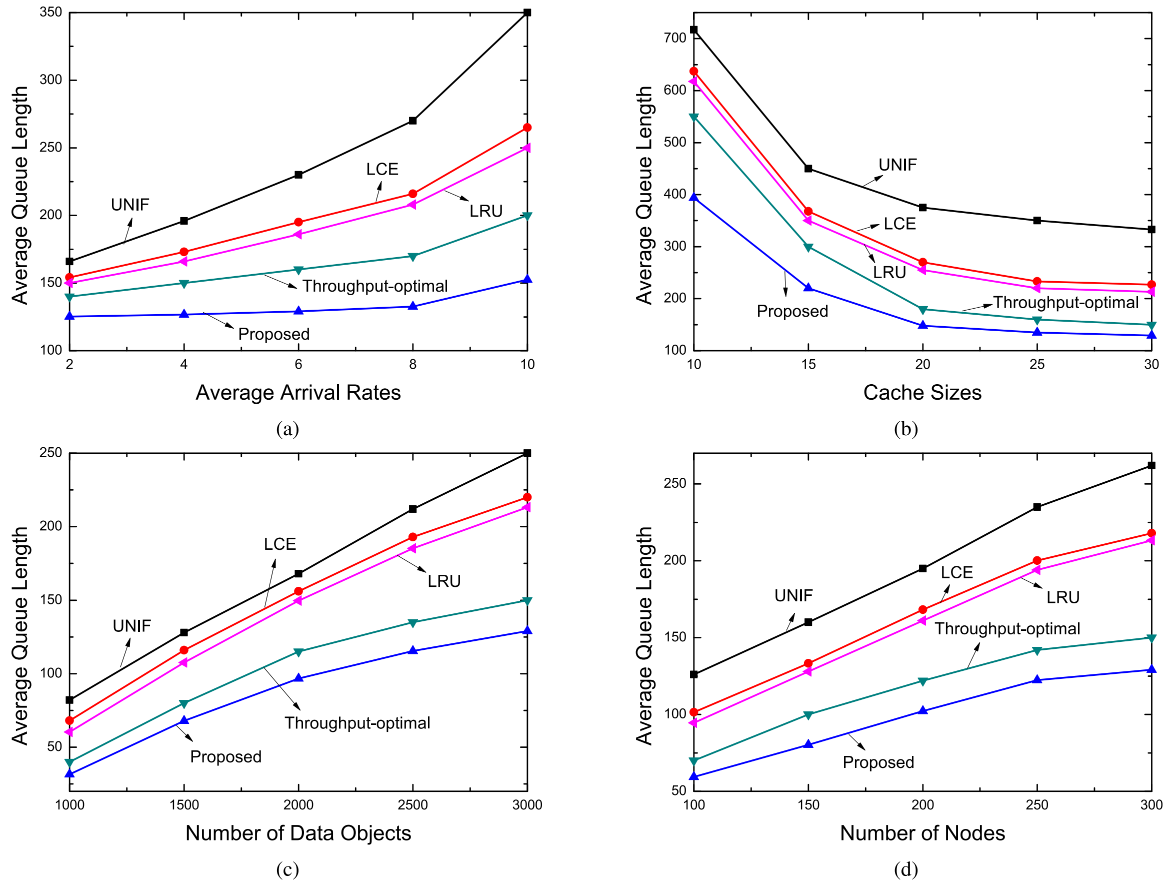


Fig. 3. Performance comparison. (a) $K = 3000$, $N = 300$ and $S_i = 30$. (b) $K = 3000$, $N = 300$ and $\lambda = 6$. (c) $N = 300$, $\lambda = 6$ and $S_i = 30$. (d) $K = 3000$, $S_i = 30$ and $\lambda = 6$.

Our simulation setup includes 300 nodes randomly forming a fully connected graph. The link capacity $C_{ab}(t)$ is calculated by Shannon formula $C_{ab}(t) = \log_2\left(1 + g_{ij}(t)p_{\text{link}}\right)$, where $g_{ij}(t)$ is the channel fading factor following the Rayleigh distribution with fading coefficient 0.01 and $p_{\text{link}}$ represents the power for each link which is $1\ mW$. Only one node works as a server which caches all the data objects to ensure that the system is stabilizable, while the other nodes have limited caches and initially, caches are empty. All nodes have the same cache size which is set to 30. Requests arrive randomly according to a Poisson distribution with the same average arrival rate for all nodes, and 3000 files are considered. In each case, 10 instances are generated to obtain the average performance.

Fig. 2 shows the stability of the proposed algorithm. We adopt an average request arrival rate as 6 per slot per data object, 3000 data objects, 300 nodes and cache size of 30 for each node. From $0s$ to $600s$, the average request queue backlog drops quickly and oscillatorily. Such an oscillatory behavior is caused for two reasons. First, if a node caches a data object, it will generate as much data as necessary to satisfy all requests in a time slot, which accounts for a fast drop. Second, due to the link capacity constraint, it takes several time slots for a data object transmitting from a node to another, which results in the occasional increase. After $600s$, the average request queue length becomes close to 150 and fluctuates within a limited region, which verifies our stability analysis of the proposed algorithm.

Fig. 3 compares the performance of the proposed algorithm and two baseline algorithms:

- *Baseline 1 (Throughput-optimal algorithm)*: Forwarding and caching algorithms are optimized by solving a decomposable Lyapunov optimization problem [15].
- *Baseline 2 (LRU-based back-pressure algorithm)*: Least recently used (LRU) cache replacement [29] combines with back-pressure data forwarding [10].
- *Baseline 3 (LCE-based back-pressure algorithm)*: Leave copies everywhere (LCE) cache replacement [23], which decides to cache all new data objects, combines with back-pressure data forwarding.
- *Baseline 4 (UNIF-based back-pressure algorithm)*: UNIF cache replacement [30], which randomly chooses a currently cached data object for replacement, combines with back-pressure data forwarding.

The simulation results in Fig. 3 show that the proposed algorithm always outperforms two baseline algorithms, because we consider forwarding and caching jointly, while the baseline algorithms optimize them separately. With the proposed framework, the local request queues capture the combined effects of both content demands and network congestion, and we dynamically replace the cached contents and forward the requests based on these information, which accounts for the performance improvement. The simulation results also demonstrate that the queue-based cache replacement schemes outperform the conventional LRU-based, LCE-based and UNIF-based cache replacement schemes. Moreover, the performance improvement brought by optimizing a tighter bound of the Lyapunov drift out-weighs the performance loss due to the sub-optimality of the stochastic optimization in each slot.

Fig. 3(a) shows that the performance gaps are almost the same for small average request arrival rates (e.g., the rates smaller than 8 in this figure), because if a node determines to cache the data object, it generates as much data as necessary in a slot to satisfy all the requests. However, when the average request arrival rates are large enough (e.g., the rates larger than 8 in this figure) with limited cache sizes and the capacity constraint, the performance degrades rapidly with the increase of the average request arrival rates. In Fig. 3(b), the performance gap is large when the cache sizes is small, which signifies the importance of joint optimization. Moreover, we observe an interesting phenomenon that the performance gains are almost the same for large cache sizes. Thus, the proposed algorithm achieves good performance with limited cache sizes. In Figs. 3(c) and 3(d), the performance gain of the proposed algorithm is large when the numbers of data objects and nodes are large, because the forwarding and caching algorithm has a large number of degrees of freedom (DoFs).

## VII. Conclusions

In this paper, we develop a low-complexity distributed forwarding and caching algorithm by embracing both Lyapunov optimization and stochastic network utility maximization. First, to extract the network state information from local queue information, we introduce a dual queue system with dynamic mapping between the request and data queues. Second, to derive an efficient forwarding and caching algorithm, we iteratively update the request/data forwarding vector and the cache placement vector using stochastic network utility maximization. We prove that the proposed algorithm achieves the queue stability and derive its tracking performance in a stochastic environment. The simulation results confirm the queue stability and show that the proposed algorithm outperforms the conventional algorithms.

## Appendix A
## Lyapunov Optimization Problem

The Lyapunov drift in slot $t$ is given by $\Delta\big(\mathbf{Q}(t)\big) = \mathbb{E}\big[L\big(\mathbf{Q}(t+1)\big) - L\big(\mathbf{Q}(t)\big)|\mathbf{Q}(t)\big]$. To calculate $\Delta\big(\mathbf{Q}(t)\big)$, taking square on both sides of (7), we have

$$\begin{aligned}&\big(\mathbf{Q}(t+1)\big)^2\\&\quad\le \sum_{i\in\mathcal{N}}\sum_{k\in\mathcal{K}}\big(1-b_i^k(t)\big)\\&\qquad\cdot\Bigg(\big(Q_i^k(t)\big)^2+\big(A_i^k(t)+\min\{H_i^k(t),A_i^k(t)\}\big)^2\\&\qquad+\Big(\sum_{j\in Z(i)}\mu_{ij}^k(t)\Big)^2+2Q_i^k(t)(A_i^k(t)\\&\qquad+\min\{H_i^k(t),A_i^k(t)\}\big)-2Q_i^k(t)\sum_{j\in Z(i)}\mu_{ij}^k(t)\Bigg),\end{aligned}\tag{25}$$

To obtain the Lyapunov drift in (11) and (12), we rearrange the items in (25) as

$$\begin{aligned}\Delta\big(\mathbf{Q}(t)\big)\le B_1-\sum_{i\in\mathcal{N}}\sum_{k\in\mathcal{K}}\Bigg(&2\big(1-b_i^k(t)\big)Q_i^k(t)\sum_{j\in Z(i)}\mu_{ij}^k(t)\\&-4Q_i^k(t)A_i^k(t)+b_i^k(t)\big(Q_i^k(t)\big)^2\\&+b_i^k(t)\Big(\sum_{j\in Z(i)}\mu_{ij}^k(t)\Big)^2\Bigg),\end{aligned}\tag{26}$$

where it is used that $\min\{H_i^k(t),A_i^k(t)\}\le A_i^k(t)$, $\mu_{ij}^k(t)\le C_{ij}(t)$ and $B_1$ is given by

$$B_1=\sum_{i\in\mathcal{N}}\sum_{k\in\mathcal{K}}\mathbb{E}\big[4\big(A_i^k(t)\big)^2\big]+\sum_{i\in\mathcal{N}}\sum_{j\in Z(i)}\mathbb{E}\big[\big(C_{ij}(t)\big)^2\big],\tag{27}$$

which is a bounded constant. To stabilize the system, we minimize the Lyapunov drift in (26) and obtain (11) and (12).

To obtain the Lyapunov drift in (14), we rearrange the terms in (25) as

$$\begin{aligned}\Delta\big(\mathbf{Q}(t)\big)\le B-\mathbb{E}\Bigg[\sum_{i\in\mathcal{N}}\sum_{k\in\mathcal{K}}\Bigg(&2\big(1-b_i^k(t)\big)Q_i^k(t)\sum_{j\in Z(i)}\mu_{ij}^k(t)\\&-4Q_i^k(t)A_i^k(t)+b_i^k(t)\big(Q_i^k(t)\big)^2\Bigg)\Bigg],\end{aligned}\tag{28}$$

where $B$ is defined as a finite constant that bound the terms $\sum_{i\in\mathcal{N}}\sum_{k\in\mathcal{K}}\big(1-b_i^k(t)\big)\Bigg(\big(A_i^k(t)+\min\{H_i^k(t),A_i^k(t)\}\big)^2+\Big(\sum_{j\in Z(i)}\mu_{ij}^k(t)\Big)^2+\min\{H_i^k(t),A_i^k(t)\}\Bigg)$. Observing that $\min\{H_i^k(t),A_i^k(t)\}\le A_i^k(t)$, $\mu_{ij}^k(t)\le C_{ij}(t)$, we have that $B$ is given by

$$\begin{aligned}B=&\sum_{i\in\mathcal{N}}\sum_{k\in\mathcal{K}}\mathbb{E}\big[4\big(A_i^k(t)\big)^2\big]\\&+\sum_{i\in\mathcal{N}}\sum_{j\in Z(i)}\mathbb{E}\big[\big(C_{ij}(t)\big)^2\big]+\sum_{i\in\mathcal{N}}\sum_{k\in\mathcal{K}}\lambda_i^k.\end{aligned}\tag{29}$$

To stabilize the system, we minimize the Lyapunov drift in (28) and obtain (14). Note that we have $B > B_1$, so $B_1$ is replaced with $B$.

## Appendix B
## Proof of Theorem 3

Consider a time interval $[0,T]$. During this interval, the optimal value changes $T$ times. As a result, the distance $\{g(t), 1\le t\le T\}$ between the iterated $Y(t)$ and the target optimal solution at the end of each slot can be bounded by

$$\begin{aligned}&g(1)\le Y(0)\beta(1)\\&g(m)\le g(m-1)\beta(m)+\delta_{m-1,m},\end{aligned}\tag{30}$$

where $\delta_{m-1,m}$ denotes the distance between the target optimal solution of slot $m-1$ and that of slot $m$. Iterating (30) from

$m = 1$ to $m = T$, we can obtain

$$g(T) \le \Big( (Y(0)\beta(1) + \delta_{1,2})\beta(2) + \delta_{2,3} \Big)\beta(3) + \cdots$$
$$\le Y(0)\prod_{m=1}^{T}\beta(m) + \sum_{l=1}^{T}\prod_{m=l}^{T}\delta_{m-1,m}\beta(m)$$
$$\le Y(0)\beta^{(1+T)T/2} + \delta\beta\frac{1-\beta^{(T-1)}}{1-\beta}, \tag{31}$$

where $\delta = \max_m\{\delta_{m-1,m}\}$ is the maximum distance between any two optimal values and $\beta = \max_m \beta(m)$. Hence, $g(T)$ is a strictly decreasing function of $T$.

Therefore, there exists a $T$ such that for $t > T$, for a region $\mathcal{X} = \{x | x \le \max_{t>T} Y^*(t) + \delta\frac{\beta}{1-\beta} + \Delta\}$, such that the trajectory is always in the limit region $\mathcal{X}$ for $t > T$, where $\Delta$ can be chosen arbitrarily small because $Y(0)\beta^{(1+T)T/2} - \delta\frac{\beta^T}{1-\beta}$ can be arbitrarily small with a large enough $T$.

## Appendix C
## Proof of Theorem 4

First, we analyze the stability region and the average queue length when optimally solving the scheduling problem. Let $\mu_{ij}^{k*}(t)$ and $b_i^{k*}(t)$ denote the optimal solution by solving (11).

### *A. Stability Region*

The Lyapunov drift should be negative semi-definite to stabilize the system. According to (26), we have

$$B + \sum_{i\in\mathcal{N}}\sum_{k\in\mathcal{K}} 2Q_i^k(t)A_i^k(t)$$
$$\le \sum_{i\in\mathcal{N}}\sum_{k\in\mathcal{K}} \Bigg( b_i^{k*}(t)\Big(Q_i^k(t) - \sum_{j\in Z(i)}\mu_{ij}^{k*}(t)\Big)^2 + 2Q_i^k(t)\sum_{j\in Z(i)}\mu_{ij}^{k*}(t)\Bigg). \tag{32}$$

If the average input vector $\boldsymbol{\lambda}$ satisfies the above equation, then the average input vector $\boldsymbol{\lambda}$ is within the stability region $\Lambda$.

### *B. Average Queue Length*

For any input rate vector $\boldsymbol{\lambda}$ inside the capacity region $\Lambda$, the average output rate should not be smaller than the sum of $\lambda_i^k$ and $\epsilon$. Note that the influence of (6) is included in $\epsilon$. The corresponding Lyapunov drift can be obtained

$$\mathbb{E}[L(\mathbf{Q}(t+1))] - \mathbb{E}[L(\mathbf{Q}(t))]$$
$$\le B + \sum_{i\in\mathcal{N}}\sum_{k\in\mathcal{K}} 2\mathbb{E}[Q_i^k(t)]\lambda_i^k - \sum_{i\in\mathcal{N}}\sum_{k\in\mathcal{K}} 2\mathbb{E}[Q_i^k(t)]\big(\lambda_i^k + \epsilon\big)$$
$$\le B - 2\epsilon\sum_{i\in\mathcal{N}}\sum_{k\in\mathcal{K}}\mathbb{E}[Q_i^k(t)]. \tag{33}$$

Summing over $t$ from 0 to $T$, and let $T \to \infty$, we have

$$\lim_{T\to\infty}\frac{1}{T}\sum_{t=1}^{T}\sum_{i\in\mathcal{N}}\sum_{k\in\mathcal{K}}\mathbb{E}[Q_i^k(t)] \le \frac{B}{2\epsilon}. \tag{34}$$

If the performance under the proposed algorithm is $\frac{1}{1+\alpha}$ of that under the optimal algorithm,

$$\sum_{i\in\mathcal{N}}\sum_{k\in\mathcal{K}}\big(b_i^{k*}(t)(Q_i^k(t) - \sum_{j\in Z(i)}\mu_{ij}^{k*}(t))^2 + 2Q_i^k(t)\sum_{j\in Z(i)}\mu_{ij}^{k*}(t)\big)$$
$$\le (1+\alpha)\sum_{i\in\mathcal{N}}\sum_{k\in\mathcal{K}}\big(b_i^k(t)(Q_i^k(t) - \sum_{j\in Z(i)}\mu_{ij}^k(t))^2 + 2Q_i^k(t)\sum_{j\in Z(i)}\mu_{ij}^k(t)\big). \tag{35}$$

Substituting (35) into (25), we get

$$\frac{B}{(1+\alpha)} + \frac{\sum_{i\in\mathcal{N}}\sum_{k\in\mathcal{K}} 2\mathbb{E}[Q_i^k(t)]\lambda_i^k}{(1+\alpha)}$$
$$\le \sum_{i\in\mathcal{N}}\sum_{k\in\mathcal{K}}\mathbb{E}\Bigg[b_i^k(t)\Big(Q_i^k(t) - \sum_{j\in Z(i)}\mu_{ij}^k(t)\Big)^2 + 2Q_i^k(t)\sum_{j\in Z(i)}\mu_{ij}^k(t)\Bigg]. \tag{36}$$

Due to the sub-optimality of the iterative algorithm, the capacity region shrinks by $\frac{1}{1+\alpha}\lambda_{\max}$, i.e., $\Lambda' = \Lambda - \frac{1}{1+\alpha}\lambda_{\max}\mathbf{I}$, and the parameter $B'$ should satisfy $B' = (1+\alpha)B$.

Therefore, the average request queue length should satisfy

$$\lim_{T\to\infty}\frac{1}{T}\sum_{t=1}^{T}\sum_{i\in\mathcal{N}}\sum_{k\in\mathcal{K}}\mathbb{E}[Q_i^k(t)] \le \frac{(1+\alpha)B}{2(\epsilon - \frac{1}{1+\alpha}\lambda_{\max})}. \tag{37}$$

Using conservation property of the number of requests,

$$\lim_{T\to\infty}\frac{1}{T}\sum_{t=1}^{T}\sum_{i\in\mathcal{N}}\sum_{k\in\mathcal{K}}Q_i^k(t) \ge \lim_{T\to\infty}\frac{1}{T}\sum_{t=1}^{T}\sum_{i\in\mathcal{N}}\sum_{k\in\mathcal{K}}W_i^k(t). \tag{38}$$

Note that it is an inequality, because $W_i^k(t)$ does not capture the requests which re-enter the network due to receiving the dummy data. Substitute (38) into (37) and take $T \to \infty$,

$$\lim_{T\to\infty}\frac{1}{T}\sum_{t=1}^{T}\sum_{i\in\mathcal{N}}\sum_{k\in\mathcal{K}}W_i^k(t) \le \frac{(1+\alpha)B}{2(\epsilon - \frac{1}{1+\alpha}\lambda_{\max})}. \tag{39}$$

We analyze the average data queue length. For one thing, the arrival and the departure of the data queues just equal those of the request queues in the reversed direction, and the stability of the request queue implies that the average arrival rate equals the average departure rate. For another, the generate of data is triggered by the requests, meaning that the number of requests should be larger than that of data. Therefore, we have

$$\mathbb{E}\Bigg[\sum_{i\in\mathcal{N}}\sum_{k\in\mathcal{K}}D_i^k(t)\Bigg] \le \mathbb{E}\Bigg[\sum_{i\in\mathcal{N}}\sum_{k\in\mathcal{K}}Q_i^k(t)\Bigg] \le \frac{(1+\alpha)B}{2(\epsilon - \frac{1}{1+\alpha}\lambda_{\max})}. \tag{40}$$

## Appendix D
## Proof of Theorem 5

To evaluate the number of the dummy data objects, we consider a node $i$ whose data queue length is 0, i.e., the dummy data must be generated to balance the transmission. Observing the transmission rate of the requests is the same as the total transmission rate of the actual and dummy data, we have

$$\sum_{j\in Z(i)}\sum_{k\in\mathcal{K}}\gamma_{ij}^k(t) = \sum_{j\in Z(i)}\sum_{k\in\mathcal{K}}(r_{ji}^k(t))^+ + \sum_{j\in Z(i)}\sum_{k\in\mathcal{K}}(\mu_{ji}^k(t))^+ - \sum_{j\in Z(i)}\sum_{k\in\mathcal{K}}(\mu_{ij}^k(t))^+. \quad (41)$$

Summing (41) over $t$ from 0 to $T$, we obtain

$$\sum_{i\in\mathcal{N}}\sum_{k\in\mathcal{K}}\sum_{t=0}^{T}H_i^k(t) = \sum_{i\in\mathcal{N}}\sum_{k\in\mathcal{K}}\left(D_i^k(t)+Q_i^k(t)\right). \quad (42)$$

To study the total number of dummy data in a probabilistic sense, we first analyze the time expectation of the average queue lengths of requests/data. According to Theorem 4,

$$\mathbb{E}\left[\sum_{i\in\mathcal{N}}\sum_{k\in\mathcal{K}}D_i^k(t)\right]\le\mathbb{E}\left[\sum_{i\in\mathcal{N}}\sum_{k\in\mathcal{K}}Q_i^k(t)\right]\le\frac{(1+\alpha)B}{2(\epsilon-\frac{1}{1+\alpha}\lambda_{\max})}. \quad (43)$$

Then, we analyze the variance of the request/data queue lengths. The arrival rate of the request queue is $A_i^k$. The departure rate of the request queue is $\sum_{i\in\mathcal{N}}b_i^k(t)\sum_{j\in Z(i)}(\mu_{ji}^k(t))^+$ from the network's perspective. We have

$$\mathrm{var}\left(\sum_{i\in\mathcal{N}}Q_i^k(t)\right)\le\mathrm{var}\left(\sum_{i\in\mathcal{N}}A_i^k(t)\right)+\mathrm{var}\left(\sum_{i\in\mathcal{L}}C_{ji}(t)\right), \quad (44)$$

where var represents the variance and the inequality is obtained

$$\sum_{i\in\mathcal{N}}b_i^k(t)\sum_{j\in Z(i)}(\mu_{ji}^k(t))^+\le\sum_{i\in\mathcal{N}}\sum_{j\in Z(i)}(\mu_{ji}^k(t))^+\le\sum_{i\in\mathcal{L}}C_{ji}(t), \quad (45)$$

where the first inequality is established using $b_i^k(t)\le 1$, and the second one is established using capacity constraint. Similarly, considering the arrival/departure of the data queues are just the reverse of that of the request queues, we have

$$\mathrm{var}\left(\sum_{i\in\mathcal{N}}D_i^k(t)\right)\le\mathrm{var}\left(\sum_{i\in\mathcal{N}}A_i^k(t)\right)+\mathrm{var}\left(\sum_{i\in\mathcal{L}}C_{ji}(t)\right). \quad (46)$$

Therefore, the time expectation and the variance of the total number of the dummy data are upper bounded by

$$\mathbb{E}\left[\sum_{i\in\mathcal{N}}\sum_{t=0}^{T}H_i^k(t)\right]\le\frac{(1+\alpha)B}{(\epsilon-\frac{1}{1+\alpha}\lambda_{\max})}$$
$$\mathrm{var}\left(\sum_{i\in\mathcal{N}}\sum_{t=0}^{T}H_i^k(t)\right)\le 2\mathrm{var}\left(\sum_{i\in\mathcal{N}}A_i^k(t)\right)+2\mathrm{var}\left(\sum_{i\in\mathcal{L}}C_{ji}(t)\right). \quad (47)$$

The Chebyshev's theorem [31] finishes the proof.

## References

[1] C. Yang, Y. Yao, Z. Chen, and B. Xia, "Analysis on cache-enabled wireless heterogeneous networks," *IEEE Trans. Wireless Commun.*, vol. 15, no. 1, pp. 131–145, Jan. 2016.

[2] J. Luo, J. Zhang, Y. Cui, L. Yu, and X. Wang, "Asymptotic analysis on content placement and retrieval in MANETs," *IEEE/ACM Trans. Netw.*, vol. 25, no. 2, pp. 1103–1118, Apr. 2017.

[3] E. Bastug, M. Bennis, and M. Debbah, "Living on the edge: The role of proactive caching in 5G wireless networks," *IEEE Commun. Mag.*, vol. 52, no. 8, pp. 82–89, Aug. 2014.

[4] X. Wang, M. Chen, T. Taleb, A. Ksentini, and V. C. M. Leung, "Cache in the air: Exploiting content caching and delivery techniques for 5G systems," *IEEE Commun. Mag.*, vol. 52, no. 2, pp. 131–139, Feb. 2014.

[5] H. Ahlehagh and S. Dey, "Video caching in radio access network: Impact on delay and capacity," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Apr. 2012, pp. 2276–2281.

[6] W. Wang, R. Lan, J. Gu, A. Huang, H. Shan, and Z. Zhang, "Edge caching at base stations with device-to-device offloading," *IEEE Access*, vol. 5, pp. 6399–6410, 2017.

[7] K. Shanmugam, N. Golrezaei, A. G. Dimakis, A. F. Molisch, and G. Caire, "FemtoCaching: Wireless content delivery through distributed caching helpers," *IEEE Trans. Inf. Theory*, vol. 59, no. 12, pp. 8402–8413, Dec. 2013.

[8] Y. Bao, X. Wang, S. Zhou, and Z. Niu, "An energy-efficient client precaching scheme with wireless multicast for video-on-demand services," in *Proc. IEEE APCC*, Oct. 2012, pp. 566–571.

[9] X. Wang, M. Chen, Z. Han, T. T. Kwon, and Y. Choi, "Content dissemination by pushing and sharing in mobile cellular networks: An analytical study," in *Proc. 9th IEEE Int. Conf. Mobile Adhoc Sensor Syst. (MASS)*, Oct. 2012, pp. 353–361.

[10] M. J. Neely, "Stochastic network optimization with application to communication and queueing systems," *Synthesis Lectures Commun. Netw.*, vol. 3, no. 1, pp. 1–211, 2010.

[11] B. Awerbuch and T. Leighton, "A simple local-control approximation algorithm for multicommodity flow," in *Proc. IEEE FOCS*, Nov. 1993, pp. 459–468.

[12] Y. Cui, E. M. Yeh, and R. Liu, "Enhancing the delay performance of dynamic backpressure algorithms," *IEEE/ACM Trans. Netw.*, vol. 24, no. 2, pp. 954–967, Apr. 2016.

[13] W. Wang, K. G. Shin, and W. Wang, "Distributed resource allocation based on queue balancing in multihop cognitive radio networks," *IEEE/ACM Trans. Netw.*, vol. 20, no. 3, pp. 837–850, Jun. 2012.

[14] E. Athanasopoulou, L. X. Bui, T. Ji, R. Srikant, and A. Stolyar, "Backpressure-based packet-by-packet adaptive routing in communication networks," *IEEE/ACM Trans. Netw.*, vol. 21, no. 1, pp. 244–257, Feb. 2013.

[15] E. Yeh *et al.*, "VIP: A framework for joint dynamic forwarding and caching in named data networks," in *Proc. ACM ICN*, Sep. 2014, pp. 117–126.

[16] M. Dehghan *et al.*, "On the complexity of optimal request routing and content caching in heterogeneous cache networks," *IEEE/ACM Trans. Netw.*, vol. 25, no. 3, pp. 1635–1648, Jun. 2017.

[17] A. Khreishah, J. Chakareski, and A. Gharaibeh, "Joint caching, routing, and channel assignment for collaborative small-cell cellular networks," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 8, pp. 2275–2284, Aug. 2016.

[18] S. Ioannidis and E. Yeh. (Aug. 2017). "Jointly optimal routing and caching for arbitrary network topologies." [Online]. Available: https://arxiv.org/abs/1708.05999

[19] F. Lai, F. Qiu, W. Bian, Y. Cui, and E. Yeh, (Aug. 2016). "Scaled VIP algorithms for joint dynamic forwarding and caching in named data networks." [Online]. Available: https://arxiv.org/abs/1608.04198

[20] S. Eum, K. Nakauchi, M. Murata, Y. Shoji, and N. Nishinaga, "CATT: Potential based routing with content caching for ICN," in *Proc. SIGCOMM*, Aug. 2012, pp. 49–54.

[21] Z. Ming, M. Xu, and D. Wang, "Age-based cooperative caching in information-centric networks," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM)*, Mar. 2012, pp. 268–273.

[22] W. Wang and V. K. N. Lau, "Delay-aware cross-layer design for device-to-device communications in future cellular systems," *IEEE Commun. Mag.*, vol. 52, no. 6, pp. 133–139, Jun. 2014.

[23] Y. Cui, V. K. N. Lau, R. Wang, H. Huang, and S. Zhang, "A survey on delay-aware resource control for wireless systems—Large derivation theory, stochastic Lyapunov drift and distributed stochastic learning," *IEEE Trans. Inf. Theory*, vol. 58, no. 3, pp. 1677–1700, Mar. 2012.

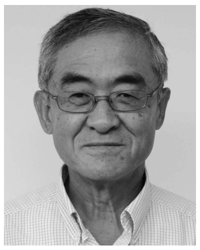
[24] E. M. Yeh, "Multiaccess and fading in communication networks," Ph.D. dissertation, Dept. Elect. Eng. Comput. Sci., MIT, Cambridge, MA, USA, Sep. 2001.

[25] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, 3rd ed. Belmont, MA, USA: Athena Scientific, 2007.
[26] J. Chen, V. K. N. Lau, and Y. Cheng, "Distributive network utility maximization over time-varying fading channels," *IEEE Trans. Signal Process.*, vol. 59, no. 5, pp. 2395–2404, May 2011.
[27] Y. Cheng and V. K. N. Lau, "Distributive power control algorithm for multicarrier interference network over time-varying fading channels: Tracking performance analysis and optimization," *IEEE Trans. Signal Process.*, vol. 58, no. 9, pp. 4750–4760, Nov. 2009.
[28] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1989.
[29] T. R. Puzak, "Analysis of cache replacement-algorithms," Ph.D. dissertation, Dept. Elect. Comput. Eng., Univ. Massachusetts Amherst, Amherst, MA, USA, 1985.
[30] E. Yeh *et al.* (Oct. 2013). "Forwarding, caching and congestion control in named data networks." [Online]. Available: https://arxiv.org/abs/1310.5569
[31] G. G. Lorentz, *Approximation of Functions*, 2nd ed. New York, NY, USA: Chelsea, 1986.

**Yitu Wang** (S'16) received the B.S. degree from Zhejiang University, Hangzhou, China, in 2013, where he is currently pursuing the Ph.D. degree. In 2014, he was a Visiting Student with the University of Paris-Sud, Orsay, France. His research interests mainly focus on stochastic optimization for cross-layer resource allocation in wireless networks and cache-enabled networks.

**Wei Wang** (S'08–M'10–SM'15) received the B.S. and Ph.D. degrees from the Beijing University of Posts and Telecommunications, China, in 2004 and 2009, respectively. From 2007 to 2008, he was a Visiting Student with the University of Michigan, Ann Arbor, MI, USA. From 2013 to 2015, he was a Hong Kong Scholar with the Hong Kong University of Science and Technology, Hong Kong. He is currently an Associate Professor with the College of Information Science and Electronic Engineering, Zhejiang University, China. His research interests mainly focus on stochastic optimization for cross-layer resource allocation in wireless networks, and caching and computing in wireless networks. He is an Editor of the book *Cognitive Radio Systems*, and serves as the Guest Editor for the *Series on Network Softwarization and Enablers*, the IEEE JOURNAL OF SELECTED AREAS IN COMMUNICATIONS, an Editor for the IEEE ACCESS, *Transactions on Emerging Telecommunications Technologies*, and the *KSII Transactions on Internet and Information Systems*.

**Ying Cui** (S'08–M'12) received the B.E. degree in electronic and information engineering from Xi'an Jiao Tong University, Xi'an, China, in 2007, and the Ph.D. degree in electronic and computer engineering from The Hong Kong University of Science and Technology, Hong Kong, in 2011. From 2012 to 2013, she was a Post-Doctoral Research Associate with the Department of Electrical and Computer Engineering, Northeastern University, Boston, MA, USA. From 2013 to 2014, she was a Post-Doctoral Research Associate with the Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA, USA. Since 2015, she has been an Associate Professor with the Department of Electronic Engineering, Shanghai Jiao Tong University, China. Her current research interests include cache-enabled wireless networks, future Internet architecture, delay-sensitive cross-layer control, and network coding. She was selected into China's 1000 Plan Program for Young Talents in 2013. She received the Best Paper Award at the IEEE ICC, London, U.K., in 2015.



**Kang G. Shin** (S'75–M'78–SM'83–F'92–LF'12) is currently the Kevin and Nancy O'Connor Professor of computer science with the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI, USA. He was a co-founder of a couple of startups and also licensed some of his technologies to industry. His current research focuses on QoS-sensitive computing and networking and embedded real-time and cyber-physical systems.

He has supervised the completion of 82 Ph.D. students, authored/coauthored over 900 technical articles, a textbook, and over 30 patents or invention disclosures, and received numerous awards, including the 1987 Outstanding IEEE Transactions of Automatic Control Paper Award, the 2003 IEEE Communications Society William R. Bennett Prize Paper Award, the Best Paper Award from the 2010 and the 2000 USENIX Annual Technical Conferences, the Best Paper Award from the 2011 ACM International Conference on Mobile Computing and Networking and the 2011 IEEE International Conference on Autonomic Computing. He has also received several institutional awards, including the Research Excellence Award in 1989, the Outstanding Achievement Award in 1999, the Distinguished Faculty Achievement Award in 2001, and the Stephen Attwood Award from the University of Michigan (the highest honor bestowed to Michigan Engineering faculty) in 2004, the Distinguished Alumni Award of the College of Engineering, Seoul National University, in 2002, the 2003 IEEE RTC Technical Achievement Award, and the 2006 Ho-Am Prize in Engineering (the highest honor bestowed to Korean-origin engineers). He is a fellow of the ACM.

**Zhaoyang Zhang** (M'10) received the Ph.D. degree in communication and information systems from Zhejiang University, Hangzhou, China, in 1998, where he is currently a Full Professor with the College of Information Science and Electronic Engineering. He has coauthored over 150 refereed international journal and conference papers, as well as two books in his areas of interest. His research interests are mainly focused on information theory and coding theory, signal processing techniques, and their applications in wireless communications and networking. He is a co-recipient of three conference Best Paper Awards/Best Student Paper Award. He is currently serving as an Editor for the IEEE TRANSACTIONS ON COMMUNICATIONS, *IET Communications*, and several other international journals. He has served as a Technical Program Committee Co-Chair or a Symposium Co-Chair for many international conferences, such as the 2013 International Conference on Wireless Communications and Signal Processing and the 2014 IEEE Global Communications Conference Wireless Communications Symposium.