Stability, Rate and Delay Analysis of Single Bottleneck caching networks

Fatemeh Rezaei, *Member, IEEE*, and Babak H. Khalaj, *Member, IEEE*

***Abstract*— Caching has been widely considered an efficient way of reducing and balancing the growing traffic in communications networks in recent years. The cache network of interest consists of one content server connected via a shared link to a number of caching nodes, also known as a single bottleneck caching network. In this paper, for the first time, the stochastic requests traffic model in such networks is considered and a performance analysis is provided based on such a realistic assumption. In addition, we introduce new comprehensive performance metrics which simultaneously take into account, the cache hit probability, load on the bottleneck link, and requests arrival rates. The main contribution of this paper is to present a system model based on queuing theory and provide an analysis of the stability, maximum stable throughput, load on the bottleneck link and average response delay for various coded and uncoded caching schemes. Moreover, we propose a novel hybrid scheme which improves the shared link utilization factor, maximum stable throughput and delay of single bottleneck caching networks compared to existing methods. Our results, validated against simulations and real trace-driven experiments, provide interesting insights into the performance of single bottleneck caching networks.**

***Index Terms*— coded caching, delay, maximum stable throughput, single bottleneck caching networks, stability.**

I. Introduction

DEMAND for various types of contents and their growth in terms of network traffic has led to significant challenges for communication networks in terms of capacity. As an approach to alleviate such issues, network caching has become of interest in recent years. In this paper, a network with a number of caching nodes connected through a single bottleneck link to a server is considered. Such a scenario, for example, is applicable to femto-caching networks formed by small-cell base stations equipped with caches which receive data from a serving macro base station via the cellular downlink [1,2].

This work was supported in part by the Institute for Research in Fundamental Sciences (IPM) and in part by the Iran National Science Foundation under Grant 92017806.

Fatemeh Rezaei is with the Department of Electrical Engineering, Sharif University of Technology, Tehran 11365-11155, Iran (e-mail: f_rezaei@ee.sharif.edu).
Babak. H. Khalaj is with the Department of Electrical Engineering, Sharif University of Technology, Tehran 11365-11155, Iran, School of Computer Science, Institute for Research in Fundamental Sciences (IPM), Tehran 19538-33511, Iran (e-mail: khalaj@sharif.edu).

In our view, there are two categories of caching schemes. Traditional caching schemes, such as Least Recently Used (LRU) and Least Frequently Used (LFU) [3], are policies to manage a single cache by specifying the rules for the insertion of a new content or eviction of the old contents. Since, there is no coding in these schemes, we call them *uncoded caching schemes*. On the other hand, there are other works considering caching in communication networks.

These approaches consist of two phases, namely, *cache placement phase* and *delivery phase*. The motivation of these approaches is to propose caching schemes that enable use of coding in the delivery phase in order to reduce file transmissions in the network. We call these approaches as *coded caching schemes*. Coded caching schemes have been proposed for single bottleneck caching networks in some recent works [4-7]. One key shortcoming of the proposed coded caching schemes in the literature is their unrealistic assumption that the requests from all caching nodes are simultaneously present at the server. In other words, stochastic arrival time of the requests has not been taken into account in these schemes. As we will further discuss in the rest of this paper, taking into account such issues highly affects the performance of caching schemes in terms of key network characteristics such as the stability and delay.

In this paper, the stochastic arrival time and traffic model of the users' requests arriving at the caching nodes are considered. In addition, a general framework for the rate-time analysis of such networks is provided based on the aforementioned realistic assumption.

On the other hand, in the traditional single-node caching schemes, the cache hit probability is considered as the performance metric. Other works considering coded schemes in single bottleneck caching networks have considered the peak/average number of file transmissions through the shared link over all possible demands during the delivery phase as the performance metric. However, such a metric is not a viable metric when the requests arrival time is randomly distributed. In general, such metrics do not encompass the overall communication characteristics of the network, such as the load on the bottleneck link, stability, and delay performance which are of key importance in practical scenarios.

The problem addressed in this paper is how to provide a general framework based on queuing theory for the performance analysis of single bottleneck caching networks by considering stochastic arrivals of the requests at different nodes. In addition, we address the question that what is a proper metric for comparing the performance of different uncoded and coded caching schemes in such networks, and especially their effects on the performance of the bottleneck link. By means of the proposed queue models, we introduce comprehensive performance metrics which simultaneously take into account the cache hit probability, load on the bottleneck link, and requests arrival rates. Moreover, this paper provides key insight on the stability characteristic of the bottleneck link and bounded or unbounded delay behavior of such networks.

The main contributions of this paper which focuses on single bottleneck caching networks can be summarized as follows:

- Proposing a novel platform based on queuing theory and introducing new performance metrics in order to compare different uncoded and coded caching schemes.
- Providing the service rate, utilization factor, stability, maximum stable throughput and delay analysis of different caching schemes, considering a realistic framework based on the Independent Reference Model (IRM) and renewal traffic models for the request.
- Proposing a novel hybrid coded caching scheme which outperforms other existing schemes in terms of the load on the bottleneck link, maximum stable throughput and delay.

The paper is organized as follows. In section II, related works are reviewed. Section III describes the system model. In section IV, the performance analysis of caching networks, in addition to a novel coded caching scheme are proposed. In section V, the performance evaluation through numerical results is presented. Finally, section VI concludes the paper.

I. Related Works

The problem of caching in wireless networks has been addressed from different perspectives in [1, 2, 8-10]. Some studies have also investigated the caching problem in content centric networks [11, 12]. In [13, 14], the caching performance in terms of hit/miss probability has been investigated.

References [4-7] studied the coded caching schemes in caching networks. They have considered a single bottleneck network consisting of a file server connected through a shared link to a number of users, each equipped with a cache. These approaches consist of two phases. The cache placement phase consists of filling up the caches with functions of the files in the library. After this set-up phase, the network is used for an arbitrary long time, referred to as the delivery phase. At each request round, a subset of the nodes request subsets of the files in the library and the network must coordinate transmissions such that these requests are satisfied, i.e., at the end of each round all destinations must decode the requested set of files. The performance metric in these works is the number of time slots necessary to satisfy all the demands, which can be normalized by the number of time slots necessary to send a single file across the shared link. Therefore, their performance metric, which is called *rate* and denoted by R, is defined as the number of normalized file transmissions [6]. The authors in [4] have proposed a coded caching approach in order to achieve a reduction in the maximum number of transmitted files in the delivery phase compared to previously known caching schemes. In [5-6], the authors have reduced the average number of file transmissions over all possible requests, with nonuniform popularities, by generalizing the method presented in [4]. The authors in [7] have presented an index coding approach to address this problem.

II. System Model

In this paper, we consider a network model with one content server, N stations equipped with caches (i.e. caching nodes), and one-hop multicast transmission from the server to the stations, as illustrated in Fig. 1. Our network model is similar to [4-7] where requests are drawn from a specific same-size file library $\digamma = \{f_i, i = 1, \dots, F\}$ of size B bits, and the caching nodes are capable of storing C whole files (i.e. CB bits). The cache content of station n is denoted by Z_n. Moreover, unlike to previous works, users' requests sent to station n are modeled by an aggregate average arrival rate λ_{req_n}. Without lack of generality and for deriving closed-form equations, we assume that the model is homogenous. Therefore, the subscript $_{(n)}$ is omitted when considering a generic station. In order to derive closed form results, we consider the IRM traffic model for the stream of the requests, which is based on the following fundamental assumptions [13,14]: i) users request items from a fixed library of F files; ii) the probability p_i that a request is for file f_i, $1 \le i \le F$, is constant (*i.e.*, the file popularity does not vary over time) and is also *independent* of all past requests, generating an independent identically distributed (i.i.d.) sequence of requests. We also assume that requests sent to each station are independent from requests sent to the other stations.

The IRM traffic model ignores all temporal correlations in the stream of requests. In order to take into account the temporal locality, we also consider the *renewal traffic* model: The stream of requests arriving at a given station for each file f_i is considered to be an independent renewal process [15] where the CDF of the inter-request time t is denoted by $F_R(i,t)$. The average request rate λ_{req}^i for file f_i is then given by $\lambda_{req}^i = 1/\int_0^\infty \big(1 - F_R(i,t)\big)dt$. The overall average arrival rate of the requests at a given cache is $\lambda_{req} = \sum_{i=1}^F \lambda_{req}^i$. Note that, by adopting a file popularity law analogous to the one considered by the IRM, we also have $\lambda_{req}^i = \lambda_{req} p_i$ [14].

The hit probability for file f_i at a given station is denoted by $p_{hit}(i)$, and the hit probability at a given station is given by $p_{hit} = \sum_{i=1}^F p_i p_{hit}(i)$. In our model of interest, if the requested file is hit in the related cache, the file is delivered from the caching node, otherwise the request is forwarded to the content server via the uplink and the server delivers the requested file by broadcasting the file to all of the stations.

The bottleneck link is a shared channel and its average capacity is equal to Γ file per second. For instance, considering the LTE-A network with 500 Mbps average downlink rate [16], where users are interested in downloading videos of 30 MB size, the channel can approximately transmit $\Gamma = 2$ files per second on the average. We define a time slot (i.e. unit time) as the average transmission time of one complete file (of fixed size B bits) via the shared link. Hence, the average capacity of the shared link will be one file per time slot. In order to obtain a general framework for the performance analysis, we propose the following queue models for both uncoded and coded caching schemes.

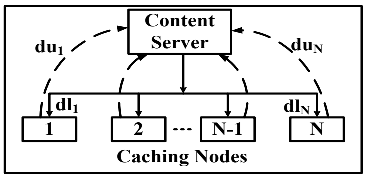


Fig. 1. Caching network model. du_n and dl_n denote the uplink and downlink delays, respectively.

A. Queue Model for Uncoded Caching Schemes

In single-bottleneck networks, as the bottleneck downlink is shared among all of the users, there is a competition for each station to receive files requested by its users via that downlink path. Therefore, we can model the function of the content server in the uncoded schemes by a controlled FIFO queue where a control unit ensures that when multiple users request the same file concurrently (requests overlap within a time slot), the server only stores the file in a single location of the queue. The requests that are missed in the corresponding caches and have not been serviced within the previous time slot, enter the server transmission queue and are served based on their arrival time. The average request arrival rate and service rate at the server transmission queue are denoted by λ and μ, respectively. The service times in our queue model have a general (arbitrary) distribution, with the mean and coefficient of variation, $\bar{s} = \frac{1}{\mu}$ and $c_s = \frac{\sigma_s}{\bar{s}}$, respectively. Therefore, in case of uncoded caching schemes, we considere a general G/G/1 queue for the content server, where inter-arrival and service times have arbitrary distributions. In case of IRM traffic, the content server functionality is then modeled by an M/G/1 queue.

B. Queue Model for Coded Caching Schemes

In contrast to the other works studying the coded caching schemes such as [4-7], in this paper, a realistic framework is assumed where requests of various users arrive at the stations randomly according to a traffic model, providing a platform for analysis of caching networks based on queuing theory. Since the traditional queuing theory models cannot handle packet combinations, network coding, and multicasting, we exploit the concept of virtual queues [17] in this paper. We consider N virtual queues with infinite buffer lengths at the content server denoted by Q_n, $n=1...N$, with the average arrival rates, λ_n, for the requests of station n sent to the server. Based on such a framework, at the server, the requested files of different caching nodes enter their corresponding virtual queues and are merged together to construct the coded packet to be transmitted. The server functionality in the coded caching schemes is then modeled by an overall queue $Q(t)$. It should be noted that in coded schemes, unlike the uncoded case, the service rate cannot be pre-determined through a given distribution and its characteristics may also be non-stationary. Therefore, we consider general definitions and theorems for queuing systems in order to model and analyze the functionality of the coded schemes. Let $Q(t)$ represent the contents of a single discrete time queuing system defined over integer steps $t \in \{0,1,2,\dots\}$. Specifically, the initial state $Q(0)$ is assumed to be a non-negative real valued random variable. Future states are driven by stochastic arrival and service processes, $\lambda(t)$ and $\mu(t)$, according to the following dynamic equation [18]:

$$Q(t+1) = max[Q(t) - \mu(t), 0] + \lambda(t) \qquad (1)$$

The value of $\lambda(t)$ represents the number of new requests that arrive during step t, and is assumed to be non-negative. The value of $\mu(t)$ represents the number of requests that can be served at step t. Without lack of generality and in order to keep consistency with the provided analysis for the uncoded schemes in this paper, the process $\lambda(t)$ is assumed to be stationary and the average request arrival rate at the server is denoted by λ.

In the following section, a performance analysis based on the proposed queue models for various uncoded and coded caching schemes is presented. Table 1 shows key notations adopted for the system model and subsequent analysis presented in this paper.

III. Stability, Throughput and Delay Analysis of Different Caching Schemes

By means of the proposed queue models, we introduce the server utilization factor, i.e. $\rho \triangleq \frac{\lambda}{\mu}$ as a comprehensive performance metric for the load on the shared link of a single bottleneck caching network. The key advantage of such a metric is that it simultaneously takes into account the cache hit probability, load on the bottleneck link, and requests arrival rates. Moreover, ρ provides key insight on the stability characteristic and bounded or unbounded delay behavior of such networks. We aim to minimize the load on the bottleneck link, by minimizing ρ, for a specific average requests arrival rate, λ_{req}. It should be noted that such utilization factor minimization is desirable as long as the stability conditions are met; naturally unstable systems lead to infinite delays.

Definition 1: A packet queue is *stable*, if the arrival and service processes of the queue are all stationary and the average arrival rate is less than the average service rate [17]. By definition, in order to have a stable system, we require $\lambda < \mu$, or in other words, $\rho < 1$. Moreover, the schemes which have lower ρ are preferred as they reduce the load on the bottleneck link and increase the maximum stable throughput of the network.

TABLE 1. Key notations for the system model and analysis

Notation	Semantics (unit)
N	Number of caching stations
F	Size of the file library users can request files from
p_i	Probability of requesting file f_i from the library
C	Cache size: The number of whole files that can be cached
λ_{req}	Average requests arrival rate at a station [packets / time slot]
Λ	Total throughput of the network [packets / time slot]
p_{hit}	Overall hit probability of the network caches
p_{miss}	Overall miss probability of the network caches
λ	Average arrival rate at the server queue [packets / time slot]
μ	Average service rate at the server queue [packets / time slot]
ρ	Shared link utilization factor
ΔT_N^k	Average waiting time in request round k in PCCS [time slot]
w	Waiting window size in WPCS [time slot]
$\overline{dh}$	Average delay of delivering a request via a station [time slot]
$\overline{du}$	Average uplink delay over all of the stations [time slot]
$\overline{dl}$	Average service delay over all of the stations [time slot]
$\overline{D}$	Average delay of responding to a file request [time slot]

Definition 2: Maximum stable throughput, i.e. Λ_{max}, is defined as the maximum average requests arrival rate over all of the stations for which the network remains stable, i.e. $\rho < 1$. It should be noted that the only requirement for properly defining the performance metric ρ is having a stationary queue and no specific queue model has to be assumed. In addition, whenever the stationary assumption for the arrival or service process does not hold, we can still use the following general definition in order to examine the stability of such a system, as will be described further in the following sections:

Definition 3: A discrete time queue *Q(t)* is *mean rate stable* if $\lim_{t\to\infty}\frac{\mathbb{E}[Q(t)]}{t} = 0$ [18].

A. Stability Analysis of Uncoded Caching Schemes

In this section, the utilization factor and maximum stable throughput for the uncoded caching schemes are proposed. First, the utilization factor for renewal traffic is derived in Theorem 1. Then, the utilization factor and maximum stable throughput in case of IRM traffic are proposed in Proposition 1 and Proposition 2, respectively. Finally, the utilization factor and maximum stable throughput for different uncoded caching policies LRU, q-LRU, LFU, and RAND in case of renewal and IRM traffics, are proposed.

We define $p_{req}(i)$ as the probability that file f_i is requested within a time slot at a given station. We also define the random variable $N_{\tau,i}$, as the number of the requests for file f_i coming to a given station within a time slot of length τ.

Lemma 1: For the renewal traffic model, $p_{req}(i)$ is obtained from $p_{req}(i) = 1 - G_i(\tau, 0)$, where $G_i(\tau,\xi)$ is the probability generating function of $N_{\tau,i}$.

Proof is provided in the Appendix.

Theorem 1: The utilization factor for uncoded caching schemes in single bottleneck caching networks in case of the renewal traffic model is derived as

$$\rho \triangleq \frac{\lambda}{\mu} = \sum_{i=1}^{F}\left(1-\left(1-p_{req}(i)(1-p_{hit}(i))\right)^N\right) \qquad (2)$$

where $p_{req}(i)$ is given by Lemma 1.

Proof: We define the random variable X_i as the number of the requests for file f_i arriving at the content server within a time slot. Taking into account the effect of the server control unit, the requests for file f_i enter the server queue with the average arrival rate λ_{f_i} given by

$$\lambda_{f_i} = \sum_{k=1}^{\infty}\mathrm{P}(X_i = k) = 1 - \mathrm{P}(X_i = 0) \qquad (3)$$

Since in the uncoded system model, the missed requests of all of the stations enter the content server, $\mathrm{P}(X_i = k)$ is obtained from

$$\begin{aligned}&\mathrm{P}(X_i = k)\\ &= \sum_{l=0}^{N-k} P\left\{\begin{matrix}\text{File } f_i \text{ is requested from } k+l \text{ stations}\\ \text{within a time slot and it is hit in the } l \text{ ones}\end{matrix}\right\}\\ &= \sum_{l=0}^{N-k}\binom{N}{k+l}\left(p_{req}(i)\right)^{k+l}\left(1-p_{req}(i)\right)^{N-(k+l)}\cdot\\ &\binom{k+l}{l}\left(p_{hit}(i)\right)^{l}\left(1-p_{hit}(i)\right)^{k} = \binom{N}{k}\left(p_{req}(i)\right)^{k}(1-\\ &p_{hit}(i))^{k}\left(1-p_{req}(i)\left(1-p_{hit}(i)\right)\right)^{N-k}\\ &= Binom\left(N, p_{req}(i)\left(1-p_{hit}(i)\right)\right)\end{aligned} \qquad (4)$$

Consequently, according to (3), the average arrival rate at the server queue, i.e. λ, is obtained from

$$\lambda = \sum_{i=1}^{F}\lambda_{f_i} = \sum_{i=1}^{F}\left(1-\left(1-p_{req}(i)(1-p_{hit}(i))\right)^N\right) \qquad (5)$$

In addition, since the bottleneck link is assumed to be error-free and no coding is performed at the server, the average service rate of the content server, μ, is considered to be equal to the link capacity, i.e. one file per time slot, and therefore, $\mu = 1$. Consequently, (2) is derived.□

Proposition 1: The utilization factor for the uncoded caching schemes in the single bottleneck caching networks, in case of the IRM traffic model, is given by

$$\rho \triangleq \frac{\lambda}{\mu} = \sum_{i=1}^{F}\left(1-\left(1-\left(1-e^{-\lambda_{req}p_i}\right)(1-p_{hit}(i))\right)^N\right) \qquad (6)$$

Proof: The proof is provided in the Appendix.

Proposition 2: The uncoded caching schemes in case of the IRM traffic model can stabilize the total throughput, $\Lambda \triangleq N\lambda_{Req}$, if $\Lambda < \frac{1}{1-p_{hit}}$.

Proof: The proof is provided in the Appendix.

As mentioned in the proof of Proposition 2 and verified in the numerical results, the aforementioned bound on the maximum stable throughput is in fact adequately tight.

The importance of the theorem and propositions proposed so far is that they are expressed in terms of the network parameters and hit probability of the applied schemes. Consequently, for each uncoded caching scheme, by inserting the value of p_{hit} in Theorem 1 and Propositions 1 and 2, the utilization factor and bound on the stable throughput can be computed. In the rest of this section, we will propose such performance metrics for LRU, q-LRU, LFU and RAND uncoded caching schemes. In order to express the hit probability of various uncoded caching schemes in terms of the network parameters, we use Che's approximation which enables us to simply express the hit probability in terms of *cache eviction time* T_c, i.e., the time needed before *C* distinct files are requested by the users. In other words, a file is in the cache at time *t*, if and only if a time smaller than T_C has elapsed since the last request for this file, where T_c can be determined based on the cache size [14].

1) *LRU*

In case of the LRU caching strategy, the following results are achieved.

Corollary 1: The utilization factor for LRU in case of IRM traffic is given by:

$$\rho = \sum_{i=1}^{F}\left(1-\left(1-\left(1-e^{-\lambda_{req}p_i}\right)e^{-\lambda_{req}p_iT_c}\right)^N\right) \qquad (7)$$

and LRU can stabilize the total throughput, if

$$\Lambda < \frac{1}{\left(1-\sum_{i=1}^{F}p_i\cdot\left(1-e^{-\lambda_{req}p_iT_c}\right)\right)} \qquad (8)$$

where T_c is obtained iteratively from $C = \sum_{i=1}^{F}(1 - e^{-\lambda_{req}p_iT_c})$.

Corollary 2: The utilization factor for LRU in case of renewal traffic is given by:

$$\rho = \sum_{i=1}^{F}\left(1-\left(1-p_{req}(i)\left(1-F_R(i,T_c)\right)\right)^N\right) \qquad (9)$$

2) *q-LRU*

q-LRU is one of the variations of LRU which differs from LRU for the insertion policy: upon arrival of a request, a content not yet stored in the cache is inserted into it with probability q. The eviction policy is the same as LRU [14].

Corollary 3: The utilization factor for q-LRU in case of IRM traffic is given by:

$$\rho = \sum_{i=1}^{F}\left(1-\left(1-\left(1-e^{-\lambda_{req}p_i}\right)\left(1-\frac{q\left(1-e^{-\lambda_{req}p_iT_C}\right)}{e^{-\lambda_{req}p_iT_C}+q\left(1-e^{-\lambda_{req}p_iT_C}\right)}\right)\right)^{N}\right) \tag{10}$$

and q-LRU can stabilize the total throughput, if

$$\Lambda < \frac{1}{\left(1-\sum_{i=1}^{F}p_i.\left(\frac{q(1-e^{-\lambda_{req}p_iT_C})}{e^{-\lambda_{req}p_iT_C}+q(1-e^{-\lambda_{req}p_iT_C})}\right)\right)} \tag{11}$$

Corollary 4: The utilization factor for q-LRU in case of renewal traffic is given by:

$$\rho = \sum_{i=1}^{F}(1-\left(1-p_{req}(i)\left(1-\frac{qF_R(i,T_C)}{1+(q-1)F_R(i,T_C)}\right)\right)^{N}) \tag{12}$$

3) *LFU*

Corollary 5: The utilization factor for LFU in case of IRM traffic is obtained from

$$\rho = \sum_{i=C+1}^{F}(1-\left(1-\left(1-e^{-\lambda_{req}p_i}\right)\right)^{N}) \tag{13}$$

and LFU can stabilize the total throughput, if

$$\Lambda < \frac{1}{(1-\sum_{i=1}^{C}p_i)} \tag{14}$$

4) *RAND*

RAND is the simplest cache replacement scheme considered in a single cache. In this scheme, to make room for a new file, a random file stored in the cache is evicted [3].

Corollary 6: The utilization factor for RAND in case of IRM traffic is given by

$$\rho = \sum_{i=1}^{F}(1-\left(1-\left(1-e^{-\lambda_{req}p_i}\right)(1-\frac{\lambda_{req}p_i\mathbb{E}[T_c]}{1+\lambda_{req}p_i\mathbb{E}[T_c]})\right)^{N}) \tag{15}$$

and RAND can stabilize the total throughput if

$$\Lambda < \frac{1}{\left(1-\sum_{i=1}^{F}p_i.\frac{\lambda_{req}p_i\mathbb{E}[T_c]}{1+\lambda_{req}p_i\mathbb{E}[T_c]}\right)} \tag{16}$$

The proofs of these corollaries are provided in the Appendix.

B. *Stability Analysis of Coded Caching Schemes*

As mentioned earlier, in the coded caching schemes, a virtual queue is considered for the requests of each station sent to the server. The role of the virtual queue modeling is to enable the server to code the requested files of different stations together and send the coded packets. The average rate of the requests of station n sent to the server, which is the average arrival rate of virtual queue Q_n, is denoted by λ_n. For the performance analysis, the server functionality is modeled by an overall queue with the aggregate average arrival rate λ and service rate $\mu(t)$. By considering the cache placement and delivery phase of the desired coded caching scheme, the parameters of the proposed queue model, i.e. λ and $\mu(t)$ can be determined. In [4], a coded caching scheme, which we denote by Partition Coded Caching Scheme (PCCS) is proposed. Such an approach is also the basis of the coded schemes for the subsequent papers [5-7], as discussed in section II. In PCCS, each file is partitioned into $\mu_0 = \binom{N}{\gamma}$ subfiles of equal size. It should be noted that PCCS is only proposed for cache sizes C such that $\gamma = C\frac{N}{F}$ is an integer less than N. In the placement phase, each station caches an equal number of the subfiles of all of the F files. The main idea of PCCS is to design the cache placement in order to create coded multicasting opportunities for any $\gamma+1$ users even with different requests. In such a scheme, the requests of all of the stations are considered together. Subsequently, in the delivery phase, the required subfiles of the requested files are coded together by linear coding (through XOR) and the coded small packets are created. The coded small packets are then transmitted via the shared link to simultaneously serve $\gamma+1$ stations. The corresponding requests at each station are obtained by decoding the received coded packets given the cache contents.

As an example, considering the case of a simple network with three caching nodes, we can model the coded caching scheme, as shown in Fig. 2, by defining three virtual queues, i.e. Q_1, Q_2 and Q_3 with average arrival rates λ_1, λ_2, λ_3 for requests of stations *1, 2,* and *3*, respectively. In this example, we have considered a library of three files, namely *A, B, C*, and three stations, i.e. *N=F=3* and caches of size one. Subsequently, γ is equal to one and so we have $\mu_0 = 3$. Therefore according to PCCS, each file is split into μ_0 equal size subfiles, i.e., $A = (A_1, A_2, A_3)$, $B = (B_1, B_2, B_3)$, and $C = (C_1, C_2, C_3)$. In the placement phase, the cache content of station n is selected as $Z_n = (A_n, B_n, C_n)$. For the delivery phase, assume for example that at a given request round, station one requests file *A*, station two file *B*, and station three file *C*. Consequently, the missing subfiles are A_2 and A_3 for station one, B_1 and B_3 for station two, and C_1 and C_2 for station three, which enter the virtual queue of the corresponding stations, i.e. Q_1, Q_2 and Q_3, respectively. Given the cache contents, stations one and two aim to exchange A_2 and B_1, stations one and three aim to exchange A_3 and C_1, and stations two and three aim to exchange B_3 and C_2. By sending $(A_2 \oplus B_1, A_3 \oplus C_1, B_3 \oplus C_2)$, the server enables all of these three exchanges. Consequently, in the delivery phase of the given request round, when the three corresponding virtual queues have packets to transmit, the server transmits three coded packets which are of size one third of a whole file. Therefore, the so called rate in this example equals *R=1* whole file.

In PCCS, for any requested file from a given station, its cache does not contain the whole file and only contains a subset of the required subfiles. Therefore, that station needs to submit that request to the server. Consequently, λ_n will be equal to the average requests arrival rate at station n, that is $\lambda_n = \lambda_{req}, \forall n = 1, \dots, N$. Therefore, the overall queue average arrival rate, λ, is equal to the sum of the average arrival rates of the virtual queues, that is

$$\lambda = \sum_{n=1}^{N}\lambda_n = N\,\lambda_{req} \tag{17}$$

It should be noted that due to the design of the studied coded caching schemes, unlike the uncoded schemes, λ_n is not reduced by the miss probability. In fact, the key role of caching in PCCS is that the server sends only the missing subfiles for each requested file, instead of sending the whole file for the missed requests in the uncoded schemes. Therefore, the total transmission time for sending the missing subfiles in PCCS is less than the transmission time of sending the whole file. We will consider this effect in our analysis of PCCS service rate, later in this section. In order to calculate the service rate of PCCS, we define the following parameters:

Definition 4: Let T_n^k, $n = 1, \dots, N$, denotes the random variable of the request arrival time of station n at the server at *request round (step)* $k \geq 0$. We define ΔT_N^k as the expectation of the *order statistical range* [20] of these random variables, i.e. $\Delta T_N^k = \mathbb{E}[T_{N:N}^k - T_{1:N}^k]$, where $T_{1:N}^k = \min(T_1^k, T_2^k, \dots, T_N^k)$ and $T_{N:N}^k = \max(T_1^k, T_2^k, \dots, T_N^k)$.

According to Definition 4, ΔT_N^k represents the average interval between the arrival of the first and last requests, or in other words the average waiting time at request round k. We also define T_{trans} as the number of time slots required for sending the coded packets over the shared link, in order to serve requests of all stations at any given request round. Since the subfiles from $\gamma + 1$ virtual queues are combined by linear coding before transmission, the number of transmitted coded packets to serve all of the requests for any given request round is $\binom{N}{\gamma+1}$. In addition, the transmission time of each small coded packet is equal to $1/\mu_0$. Consequently, we have $T_{trans} = \binom{N}{\gamma+1}/\mu_0$. (It should be noted that in the notation of [4], the so called rate parameter, i.e. $R = \binom{N}{\gamma+1}/\mu_0$, is equal to T_{trans}. The goal in [4] is to minimize the value of R under the assumption that all of the stations' requests are available simultaneously and the server can form any coded small packet at its own discretion at any time. Naturally, such an assumption is not valid in many practical scenarios. Therefore, by taking into account the stochastic arrival nature of the requests, the metric R on its own is no longer a proper figure of merit for such a network.)

In addition, let $T_{trans,1}$ denote the number of time slots needed for transmitting the required small coded packets over the shared link in order to serve the requests of any given station. In PCCS, each cache contains $\frac{C}{F}$ of each file. Therefore, the number of the required subfiles to be transmitted to each station is $(1 - \frac{C}{F})\mu_0$. Since the transmission time of each subfile is $\frac{1}{\mu_0}$, $T_{trans,1}$ is given by $T_{trans,1} = 1 - \frac{C}{F}$.

Proposition 3: The overall average service rate of the server queue at request round k in PCCS is obtained from

$$\bar{\mu}(k)) := \mathbb{E}[\mu(k)] = \frac{N}{\max(\frac{\binom{N}{\gamma+1}}{\mu_0},\ \Delta T_N^k + \left(1 - \frac{C}{F}\right))} \tag{18}$$

Proof: In PCCS, the server transmits the coded small packets at each request round to serve the requests of all of the N stations. Let $T_{wait\&trans}^k$ denote the number of time slots required to serve the requests of all of the stations at request round k, (including the waiting time at the request round and the transmission time of the coded packets sent through the shared link). Therefore, the average service time at request step k is obtained from

$$\bar{s}(k) = \frac{1}{\bar{\mu}(k)} = \frac{T_{wait\&trans}^k}{N} \tag{19}$$

In order to reduce latency, we assume that as soon as the server has received enough requests so that it can form the corresponding packet and transmit it, the server will do so. In other words, the server does not wait till all the requests arrive at each request round in order to start transmission. Based on such a strategy, in order to determine $T_{wait\&trans}^k$, we should consider the following two situations:

a) Let's consider the case that the waiting time at request round k, i.e. ΔT_N^k is large enough such that when the last request at this round arrives, all of the coded packets corresponding to the previous *N-1* requests have been already transmitted. After the arrival of the last request, only coded packets that rely on that request need to be transmitted, where by definition is performed in $T_{trans,1}$ time slots. Therefore, in this case, the total time for serving all of the requests at request round k, i.e. $T_{wait\&trans}^k$, takes $\Delta T_N^k + T_{trans,1}$ time slots. It is clear that this situation holds if the duration of the request step, i.e. ΔT_N^k, is larger than the transmission time of all of the coded packets to serve all of the node requests except for the coded packets corresponding to the last request, i.e. $T_{trans} - T_{trans,1}$. Therefore, if $\Delta T_N^k > T_{trans} - T_{trans,1}$, then $T_{wait\&trans}^k = \Delta T_N^k + T_{trans,1}$. Consequently, in case of $\Delta T_N^k > \frac{\binom{N}{\gamma+1}}{\mu_0} - (1 - \frac{C}{F})$, the average service time at request step k is obtained from

$$\bar{s}(k) = \frac{\Delta T_N^k + (1 - \frac{C}{F})}{N} \tag{20}$$

b) On the other hand, consider the case that $\Delta T_N^k \leq T_{trans} - T_{trans,1}$. In such a case, the waiting time at request step k is negligible compared to the total service time and we have $T_{wait\&trans}^k = T_{trans}$. Therefore, in case of $\Delta T_N^k \leq \frac{\binom{N}{\gamma+1}}{\mu_0} - (1 - \frac{C}{F})$, the average service time at request step k is

$$\bar{s}(k) = \frac{T_{trans}}{N} = \frac{\binom{N}{\gamma+1}}{N\mu_0} \tag{21}$$

Finally, combining (20) and (21) results in (18). □

The following two lemmas specify the characteristics of the parameter ΔT_N^k.

Lemma 2: ΔT_N^k in PCCS with IRM traffic is given by:

$$\Delta T_N^k = \left\{\begin{matrix} \sum_{j=1}^{N-1}(-1)^{j+1}\binom{N}{j}\varphi_j(k) & odd\ N \\ \sum_{j=1}^{N-1}(-1)^{j+1}\binom{N}{j}\varphi_j(k) - 2\varphi_N(k) & even\ N \end{matrix}\right\} \tag{22}$$

where

$$\varphi_j(k) = \frac{1}{\lambda_{Req}} \sum_{t_0+t_1+\dots+t_{k-1}=j} \binom{j}{t_0, t_1, \dots, t_{k-1}} \frac{(\sum_{s=0}^{k-1} st_s)!}{j^{\sum_{s=0}^{k-1} st_s + 1}} \prod_{s=0}^{k-1} \frac{1}{(s!)^{t_s}} \tag{23}$$

Proof: The proof is provided in the Appendix.

Lemma 3: The lower bound on ΔT_N^k in PCCS with IRM traffic is given by $\Delta T_N^k \geq \frac{d_N\sqrt{k}}{\lambda_{req}}$, where $d_N = \min(2\left(1 - \left(\frac{1}{2}\right)^{N-1}\right), \frac{(1-p^N-q^N)}{\sqrt{pq}})$, $p = \frac{a^2}{1+a^2}$ for any a that $T_n^k \leq a$, and $q = 1 - p$.

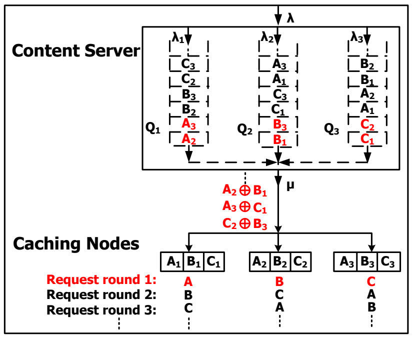


Fig. 2. An example of the virtual queue model for the coded caching schemes.

Proof: The proof is provided in the Appendix.

According to Proposition 3, the service rate in PCCS depends on request round k, and hence, PCCS does not yield stationary service rates. This is due to the fact that the service rate in PCCS depends on the waiting time at each request round. The waiting time also varies at each request round according to Lemma 2 and Lemma 3. Consequently, in order to analyze the stability of PCCS, we need to consider the general definitions and theorems of queuing systems that go beyond the stationary queues, as discussed in section III. B.

Lemma 4: (Necessary condition for mean rate stability [18]) Suppose $Q(t)$ evolves according to (1), with general processes $\lambda(t)$ and $\mu(t)$ such that $\lambda(t) \geq 0$ for all t and $\mathbb{E}[Q(0)] < \infty$. If $Q(t)$ is mean rate stable, then

$$\limsup_{t\to\infty} \frac{1}{t}\sum_{k=0}^{t-1} \mathbb{E}[\lambda(k) - \mu(k)] \leq 0 \tag{24}$$

Proof: The proof is provided in [18].

Theorem 2: The server queue is not mean rate stable in PCCS.

Proof: According to Proposition 3, we can write

$$\sum_{k=0}^{t-1} \mathbb{E}[\mu(k)] = \sum_{k=0}^{t-1} \frac{N}{\max\left(\frac{\binom{N}{\gamma+1}}{\mu_0},\ \Delta T_N^k + \left(1-\frac{C}{F}\right)\right)} \tag{25}$$

Moreover, according to Lemma 2 and 3, ΔT_N^k increases by increasing k. Therefore, there exists a $K > 0$ such that for $k > K$, the inequality $\Delta T_N^k + \left(1-\frac{C}{F}\right) \geq \frac{\binom{N}{\gamma+1}}{\mu_0}$ holds. Consequently, we have

$$\sum_{k=0}^{t-1} \mathbb{E}[\mu(k)] = K\frac{N\mu_0}{\binom{N}{\gamma+1}} + \sum_{k=K+1}^{t-1} \frac{N}{\Delta T_N^k + \left(1-\frac{C}{F}\right)} \tag{26}$$

Furthermore, according to Lemma 3, we can write

$$\sum_{k=0}^{t-1} \mathbb{E}[\mu(k)] \leq K\frac{N\mu_0}{\binom{N}{\gamma+1}} + \sum_{k=K+1}^{t-1} \frac{N\lambda_{req}}{d_N\sqrt{k}} \tag{27}$$

Due to the stationary assumption on $\lambda(k)$ and (17), we have

$$\begin{aligned}
&\limsup_{t\to\infty} \frac{1}{t}\sum_{k=0}^{t-1} \mathbb{E}[\lambda(k) - \mu(k)] \\
&= \limsup_{t\to\infty} \frac{1}{t}\sum_{k=0}^{t-1} \left(N\,\lambda_{req} - \mathbb{E}[\mu(k)]\right) \\
&= N\,\lambda_{req} - \limsup_{t\to\infty} \frac{1}{t}\sum_{k=0}^{t-1} \mathbb{E}[\mu(k)]
\end{aligned} \tag{28}$$

Substituting (27) in (28) results in

$$\begin{aligned}
&\limsup_{t\to\infty} \frac{1}{t}\sum_{k=0}^{t-1} \mathbb{E}[\lambda(k) - \mu(k)] \geq \\
&N\,\lambda_{req} - \limsup_{t\to\infty} \frac{1}{t}\left(K\frac{N\mu_0}{\binom{N}{\gamma+1}} + \sum_{k=K+1}^{t-1} \frac{N\lambda_{req}}{d_N\sqrt{k}}\right) \\
&= N\,\lambda_{req} - \frac{N\lambda_{req}}{d_N}\limsup_{t\to\infty} \frac{1}{t}\sum_{k=K+1}^{t-1} \frac{1}{\sqrt{k}}
\end{aligned} \tag{29}$$

In addition, using the Right Riemann Sum for underestimating the area under the function $\frac{1}{\sqrt{x}}$, we have $\sum_{k=K+1}^{t-1} \frac{1}{\sqrt{k}} < \int_K^{t-1} \frac{1}{\sqrt{x}}dx$. Therefore, (29) can be written as

$$\begin{aligned}
&\limsup_{t\to\infty} \frac{1}{t}\sum_{k=0}^{t-1} \mathbb{E}[\lambda(k) - \mu(k)] > \\
&N\,\lambda_{req} - \frac{N\lambda_{req}}{d_N}\limsup_{t\to\infty} \frac{1}{t}\int_K^{t-1} \frac{1}{\sqrt{x}}dx = \\
&N\,\lambda_{req} - \frac{N\lambda_{req}}{d_N}\limsup_{t\to\infty} \frac{1}{t}\left(2\sqrt{t-1} - \sqrt{K}\right) = N\,\lambda_{req} > 0
\end{aligned} \tag{30}$$

Therefore, according to Lemma 4, $Q(t)$ in PCCS is not mean rate stable for any nonzero requests arrival rate, $\lambda_{req} > 0$.□

C. WPCS (Window Partition Coded caching Scheme)

As shown in Theorem 2, PCCS cannot maintain the stability of the system. This fact is mainly due to the long waiting times for the requests arrivals from all of the stations to properly encode the packets. In order to overcome this problem, a natural extension is to encode as many packets as possible during a given window size and do not delay transmissions further in time. For such an approach, we consider a modified version of PCCS, denoted by Window Partition Coded caching Scheme (WPCS). In WPCS, the cache placement phase is the same as in PCCS, however, the delivery phase is modified. In WPCS, we consider a slotted transmission of window size w for the server transmissions. At the end of w, if only one request has arrived at the server, the requested file is sent without coding. Otherwise, the server codes as many subfiles as possible and sends the remaining subfiles without coding. If the requests of all of the stations have reached the server during a given window, the server sends the coded subfiles according to PCCS. WPCS is described in Table 2.

For example, in the case illustrated in Fig. 2 with three stations, consider the situation where only requests of the nodes 1 and 3 have arrived during a given window. In WPCS, packets that contain the subfiles of the nodes 1 and 3 are sent after proper coding and the other subfiles are sent without coding. Therefore, the packets $A_3 \oplus C_1$, A_2 and C_2 are transmitted. In the following proposition, the average service time of WPCS is derived.

Proposition 4: The average service time of WPCS is

$$\bar{s}_w = \begin{cases} \frac{w}{2} + \left(1 - \frac{C}{F}\right) & if\ N_w = 1 \\ \frac{w}{2} + \frac{1}{\mu_0}\left(\frac{1}{N_w} + \binom{N-1}{\gamma} - 1\right) & if\ 1 < N_w < \gamma + 1 \\ \frac{w}{2} + \frac{1}{\mu_0}\left(\frac{\binom{N_w}{\gamma+1}}{N_w} + \binom{N-1}{\gamma} - \binom{N_w - 1}{\gamma}\right) & O.W. \end{cases} \tag{31}$$

where N_w is the number of stations whose requests arrive during a window size w.

TABLE 2.WindowPartitionCodedcaching Scheme (WPCS)

Algorithm 1 Window Partition Coded caching Algorithm

```
1: N        :The set of all stations
2: R        :The stream of all requests sent to the server
3: Z_n      :The cache contents of station n
4: Q_n      :Virtual queue of station n at the server
5: $head(\mathbf{Q}_n)$ :The packets at the head of $\mathbf{Q}_n$
6: $head_{coded}(\mathbf{Q}_n)$ :The packets at the head of $\mathbf{Q}_n$ which will be sent coded
7: $head_{uncoded}(\mathbf{Q}_n)$:The packets at the head of $\mathbf{Q}_n$ which will be sent uncoded
8: t(.)     :Arrival time function
9: Npres(W) : Set of virtual queue indexes whose request arrives within the
              server transmission window W
10: Step 1: Cache Placement Phase
11: for all $n \in \mathbf{N}$ do
12:   $\mathbf{Z}_n \leftarrow$ Fill the cache based on PCCS cache placement
13:   $\mathbf{Q}_n \leftarrow$Send missed partitions of the requested files at cache n to the server
14: end for
15: Step 2: Delivery Phase
16: W=[0,w]
17: While R has unserved requests
18:   for all $n \in \mathbf{N}$ do
19:     if $t(head(\mathbf{Q}_n)) \in$ W then add n to Npres(W)
20:   end for
21:   Send $\oplus(head_{coded}(\mathbf{Q}_n))$ for $n \in Npres(W)$ (Send XOR of possible
                                                            packets at W)
22:   Send $head_{uncoded}(\mathbf{Q}_n)$ for $n \in Npres(W)$ (Send remaining
                                                     packets without coding)
23:   update $head(\mathbf{Q}_n)$, $head_{coded}(\mathbf{Q}_n)$, $head_{uncoded}(\mathbf{Q}_n)$ for $n \in Npres(W)$
24:   update W by shifting the window w time slots to the right
25: end while
```

Proof: The proof is provided in the Appendix.

Corollary 7: The shared link utilization factor of WPCS in single bottleneck caching networks is given by $\rho = N\,\lambda_{req}\bar{s}_W$.

Proof: Since the cache placement is the same as PCCS, the average requests arrival rate at the server queue is given by (17), and ρ is derived from (17) and (31).□

As we will present at the end of this section, unlike PCCS, WPCS can maintain the stability of the system for a given range of λ_{req} and w. Naturally, as w becomes very small, WPCS behaves more closely to the uncoded schemes, so the stability is achieved at the cost of a lower caching gain.

D. Proposing Coded-Delivery LRU caching Scheme (CDLS)

In this section, we propose a caching scheme, namely Coded-Delivery LRU caching Scheme (CDLS), which benefits from applying coding in the delivery phase, while maintaining the stability of the caching network. Moreover, it improves the performance compared to the caching schemes presented earlier, by decreasing the shared link utilization factor and increasing the maximum stable throughput. As illustrated in Table 3, CDLS performs LRU caching replacement at each cache station and uses coding in the delivery of the packets from the server. In the delivery phase, the server takes into account the contents of the virtual queues before each packet transmission. Through this process, at the earliest time that the server detects that the requested files of two stations are mutually present in each other's cache, it forms the XOR of these two requested files and transmits it at the first available transmission round. Consequently, both stations can obtain their desired files by decoding the received coded packet given their own cache content. It is evident that due to the nature of LRU, the cache of each station contains the last C files requested by that station and therefore, content of the caches are already known to the server by keeping a list of such files. We should mention that the proposed scheme can be easily extended to coding of three or more files. However, for simplicity, we concentrate on the coding of two files in this paper. CDLS is described in Table 3.

Proposition 5: The shared link utilization factor for the proposed CDLS scheme in the single bottleneck caching networks in case of IRM traffic is given by:

$$\rho = \alpha \sum_{i=1}^{F}\left(1-\left(1-\left(1-e^{-\lambda_{req}p_i}\right)e^{-\lambda_{req}p_i T_C}\right)^N\right) \quad (32)$$

and CDLS can stabilize the total throughput, if

$$\Lambda < \frac{1}{\alpha\left(1-\sum_{i=1}^{F} p_i \cdot (1-e^{-\lambda_{req}p_i T_C})\right)} \quad (33)$$

where $\frac{1}{2} \le \alpha \le 1$.

Proof: Due to the potential possibility of the coded delivery of two requested files in CDLS, we have $1 \le \mu \le 2$. Consequently, the desired results are simply derived from Corollary 1, by changing the value of μ. □

It can be easily verified that while CDLS maintains the system stability, it also results in a lower shared link utilization factor compared to the uncoded schemes.

E. Delay Analysis

In this section, we address the delay analysis of single bottleneck caching networks. The system model is as in Fig. 1: If the requested file is hit at caching node *n*, it is delivered from the cache with a cache response delay, *dh*_n, otherwise, the request is forwarded to the content server via the uplink with a communication delay, du_n, and the server delivers the requested file via the downlink with a response delay, dl_n.

Definition 5: The average response delay, $\bar{D}$, is defined as the average delay experienced by a given users in the single bottleneck caching network, for obtaining the requested files, either delivered directly from the corresponding station or sent from the content server.

By definition, the average response delay is obtained from

$$\bar{D} = p_{hit}\,\overline{dh} + (1-p_{hit})\left(\overline{du}+\overline{dl}\right) \quad (34)$$

TABLE 3. Coded-Delivery LRU caching Scheme (CDLS)

Algorithm 2 Coded-Delivery LRU caching Scheme (CDLS)

```
1: N     :The set of all caching nodes
2: R     : The stream of all requests sent to the server
3: R_n   : The stream of requests at station n
4: Z_n   : The cache contents of station n
5: Q_n   : Virtual queue of station n at the server
6: $head(\mathbf{Q}_n)$:The packet at the head of $\mathbf{Q}_n$
7: t(.)  :Arrival time function
8: at the stations side:
9:  for all $n \in \mathbf{N}$ and $i \in \mathbf{R}_n$ do
10:   $\mathbf{Z}_n \leftarrow$ Fill cache n based on LRU
11:   $\mathbf{Q}_n \leftarrow$send missed requested files at cache n to the server
12: end for
13: at the server side:
14: while R has unserved requests
15:   if (there is at least two nonempty virtual queues, $\mathbf{Q}_m$,and $\mathbf{Q}_n$)
16:       and ( $head(\mathbf{Q}_m) \in \mathbf{Z}_n$ and $head(\mathbf{Q}_n) \in \mathbf{Z}_m$ ) then
17:       send $head(\mathbf{Q}_n) \oplus head(\mathbf{Q}_m)$
18:       update $head(\mathbf{Q}_n)$
19:       update $head(\mathbf{Q}_m)$
20:   else
21:         $n = \underset{x}{\text{argmin}}\ t(head(\mathbf{Q}_x))$
22:       send $head(\mathbf{Q}_n)$
23:       update $head(\mathbf{Q}_n)$
24:   end if
25: end while
```

where the average service delay via the downlink path, $\overline{dl}$, is given by the sum of the average service time, $\bar{s}$, and the average time spent in the server queue, T_q: $\overline{dl} = \bar{s} + T_q$. Equation (34) describes the relation between the average response delay and the hit probability. The definition of the hit probability in the traditional cache schemes is the probability of existence of the requested files in the caches. In contrast, partition caching schemes, such as PCCS and WPCS, only cache fractions of the files rather than the whole files. Therefore, in order to have a comparison among different schemes, we consider the hit probability in the partition schemes as the ratio of the number of subfiles of the requested files that are stored in the cache, to the total number of subfiles of the requested files. In PCCS and WPCS, each cache contains an equal number of the subfiles of all of the F files. Therefore, due to the symmetry in the cache placement, the hit probability will be $p_{hit} = \frac{C}{F}$. Hence, according to (34), the average response delay of PCCS and WPCS is given by:

$$\bar{D} = \frac{C}{F}\,\overline{dh} + \left(1 - \frac{C}{F}\right)\left(\overline{du} + \bar{s} + T_q\right) \qquad (35)$$

Since, as shown earlier, the server queue becomes unstable in PCCS, the average time spent in the queue, T_q, and consequently, the average response delay will be unbounded.

In the following, we focus on obtaining the average response delay for the uncoded schemes. In case of IRM traffic, as discussed in section III, we consider an M/G/1 queue for the shared downlink. According to Pollaczek-Khinchin (P-K) relation [19], the average downlink delay is given by $\overline{dl} = \bar{s}\left(1 + \frac{\rho(1+c_s^2)}{2(1-\rho)}\right)$. Therefore, the average response delay is obtained from

$$\bar{D} = p_{hit}\,\overline{dh} + (1 - p_{hit})\left(\overline{du} + \frac{1}{\mu}\left(1 + \frac{\rho(1+c_s^2)}{2(1-\rho)}\right)\right) \qquad (36)$$

Proposition 6: In the single bottleneck caching networks with IRM traffic and the uncoded caching schemes, the average response delay is obtained from

$$\bar{D} = p_{hit}\,\overline{dh} + (1 - p_{hit}) \left(\overline{du} + 1 + \frac{(1+c_s^2)\sum_{i=1}^{F}(1-\left(1-\left(1-e^{-\lambda_{req}p_i}\right)(1-p_{hit}(i))\right)^N)}{2\left(1-\sum_{i=1}^{F}(1-\left(1-\left(1-e^{-\lambda_{req}p_i}\right)(1-p_{hit}(i))\right)^N)\right)}\right) \qquad (37)$$

Proof: Combining (6) and (36) results in (37). □

Proposition 6 shows that the average response delay of the uncoded schemes in unsaturated cases ($\rho < 1$) is bounded and is given as a function of the hit probability and network parameters. Moreover, using the formulas of ρ and p_{hit} for different uncoded caching schemes, provided in section IV.A, the average response delay in terms of the network parameters is obtained. For instance, $\bar{D}$ for LRU is provided by the following corollary.

Corollary 8: The average response delay for the LRU caching scheme in the single bottleneck caching networks with IRM traffic is given by

$$\bar{D} = \overline{dh}\sum_{i=1}^{F} p_i.\left(1 - e^{-\lambda_{req}p_iT_c}\right) + \left(1 - \sum_{i=1}^{F} p_i.\left(1 - e^{-\lambda_{req}p_iT_c}\right)\right).$$
$$\left(\overline{du} + 1 + \frac{(1+c_s^2)\sum_{i=1}^{F}(1-\left(1-\left(1-e^{-\lambda_{req}p_i}\right)e^{-\lambda_{req}p_iT_c}\right)^N)}{2\left(1-\sum_{i=1}^{F}(1-\left(1-\left(1-e^{-\lambda_{req}p_i}\right)e^{-\lambda_{req}p_iT_c}\right)^N)\right)}\right) \qquad (38)$$

Proof: Equation (38) is simply derived by applying $p_{hit}(i) = 1 - e^{-\lambda_{req}p_iT_c}$, derived from the Che's approximation for LRU [14], in (37).□

F. Discussion on the Performance Evaluation

Earlier in this section, we have shown that although PCCS decreases the number of file transmissions on the shared link with the assumption of co-existence of the requests of all of the stations, the price to pay is the instability of the system in case of the stochastic requests arrivals, as shown in Theorem 2. From the derivations presented so far, it is evident that uncoded schemes, WPCS, and CDLS, through proper system configuration result in stable systems up to the maximum stable throughput. The key question based on such a general view is then about finding a proper metric for comparison of different caching schemes. At first sight, increasing the average service rate seems to be a good measure of comparison. For example, it is evident from Proposition 4 that, if $w < \frac{2C}{F}$, the average service rate of WPCS is greater than the service rate of the uncoded schemes, for sufficiently small values of λ_{req}. However, it should be noted that such an increase in the average service rate comes at the cost of an increase in the average request arrival rate at the server, λ. In fact, for a specific value of λ_{req}, in the uncoded schemes only those requests that were missed in the caches enter the server queue. Thus, for the uncoded schemes, according to (5), λ, is reduced by $p_{miss} = 1 - p_{hit}$. However, according to (17), due to the cache partitioning in WPCS, all the requests arriving at the stations are sent to the server, and consequently λ will not change. Therefore, for a performance comparison, a sole comparison of the average service rates is not justified and it is the comparison of the utilization factors that provides a comprehensive measure of performance, due to the fact that the effect of λ and μ is jointly taken into account in ρ. As discussed earlier, the scheme that provides a smaller ρ, for a given λ_{req}, improves the performance of the caching networks by decreasing the load on the bottleneck link and increasing the maximum stable throughput.

It should be noted that the utilization factor also gives a measure on the shared link rate, R', which is defined as the average number of the whole files transmitted over the shared link at each time slot. Under the assumptions of [4] in which all of the stations' requests are available simultaneously and there is no constraint on the shared link capacity, R' will in fact be equal to $\mathbb{E}[R]$. However, as mentioned earlier, if these assumptions are not met, R and subsequently $\mathbb{E}[R]$ will not constitute well-justified performance metrics, while R'can still be considered a proper indicator of the shared link rate. In case of saturated and oversaturated systems, $\rho \geq 1$, R' will be equal to the full link capacity, i.e. $R' = 1$. On the other hand, in case of unsaturated systems where $\rho < 1$, we have $R' = \rho$. Therefore, in PCCS which leads to an unstable system, we have $R' = 1$, while in the stable regions of the other schemes, we have $R' = \rho < 1$. Therefore, in such networks, a scheme with a lower ρ is preferred as it will lead to a lower shared link average rate.

TABLE 4. The simulation and trace-driven experiment parameters

Notation	Value
N	10, 40, 360
F	100,78.9K
C	Variable, 10,40,1000
Number of Requests	10K, 123.3K

As will be shown numerically in section V, the proposed CDLS leads to a significant performance improvement as it takes the advantages of both uncoded and coded caching schemes. While the strength of the traditional uncoded schemes is in their high hit probabilities and reducing the average arrival rate at the server, the advantage of the coded schemes, such as PCCS and WPCS, is in their potential to increase the average service rate. Therefore, as shown earlier, CDLS achieves the smaller utilization factor compared to other schemes, in addition to the higher maximum stable throughput, by maintaining the smaller value of λ, resulting from the LRU cache replacement, and increasing μ by exploiting the proper coding at the server.

IV. Performance Evaluation through Simulations and Trace-Driven Experiments and Insights

In this section, the analytic expressions derived in this paper are validated through simulations and real trace-driven experiments. Figs. 3-5, provide simulations as well as the analytic results, demonstrating the accuracy of our analytic results. We model the network and arriving requests in MATLAB environment where simulations are performed for various number of caching nodes *N*, and cache sizes *C*. First, we present the results achieved with the assumption of IRM traffic and the Zipf file popularity distribution [21] with the exponent parameter $\alpha = 1$ for a total 10K of requests. Next, we further validate our results by a real trace-driven experiment on traffic of YouTube video requests, as illustrated in Fig. 6. The values of the simulation and trace-driven experiment parameters are given in Table 4.

Fig. 3 plots the hit probability as a function of the cache size for different caching schemes. As illustrated in Fig. 3, although the hit probabilities are close to each other under the Uniform popularity distribution, the hit probability of the partition coded schemes, i.e. PCCS and WPCS, is less than the hit probability of LRU, CDLS and LFU for all of the cache sizes in case of the Zipf distribution.

Fig. 4.a shows the server utilization factor as a function of the overall average requests rate, $\Lambda = N\lambda_{req}$, for all of the schemes except for PCCS. Naturally, as shown in Theorem 2, PCCS leads to an unstable system for which ρ is not well-defined. On the other hand, we can ensure stable WPCS systems by choosing small enough window sizes. We should note that in terms of the utilization factor, WPCS does not perform better than the uncoded schemes. However, in the proposed CDLS scheme, the arrival rate at the server queue is the same as in LRU. Therefore, the higher service rate of CDLS results in the lower server utilization factor and higher maximum stable throughput as illustrated in Fig. 4.a.

According to Fig. 4.a, the maximum stable throughput for LRU is $\Lambda_{max} = 4.02$ and for LFU is $\Lambda_{max} = 5.64$. As illustrated in Fig. 3, in case of the Zipf distribution and at $C = 40$, the hit probability of LRU is 0.752, and for LFU it is 0.825. According to Proposition 2, in case of LRU we have $\Lambda_{max} < \frac{1}{(1-0.752)} = 4.032$, and in case of LFU we have $\Lambda_{max} < \frac{1}{(1-0.825)} = 5.714$, illustrating that this bound is adequately tight. In addition, for WPCS with $w = 0.1$, we have $\Lambda_{max} = 1.5$, which is significantly less than Λ_{max} of LRU and LFU. By increasing the window size w, Λ_{max} of WPCS decreases until the system becomes totally unstable. As illustrated in Fig. 4.a, for the proposed CDLS scheme, we have $\Lambda_{max} = 7.8$, which is explicitly more than Λ_{max} of the other schemes. As shown in Fig. 4.a, Λ_{max} for RAND is less than other uncoded schemes, while the performance of q-LRU with $q = 0.01$ is slightly better than LRU.

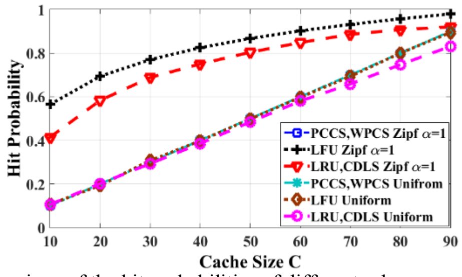


Fig. 3. Comparison of the hit probabilities of different schemes under Zipf and Uniform popularity distributions. *N=10, F=100.*

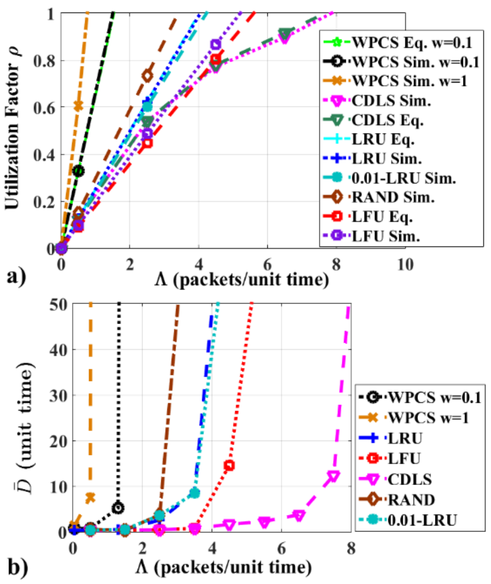


Fig. 4. a) Utilization factors and maximum stable throughputs of different schemes. b) The average response delay of different schemes as a function of Λ. Parameters: *F=100, N=10, C=40*,$\alpha = 1$, $\overline{dh} = 0.1$, $\overline{du} = 10^{-7}$.

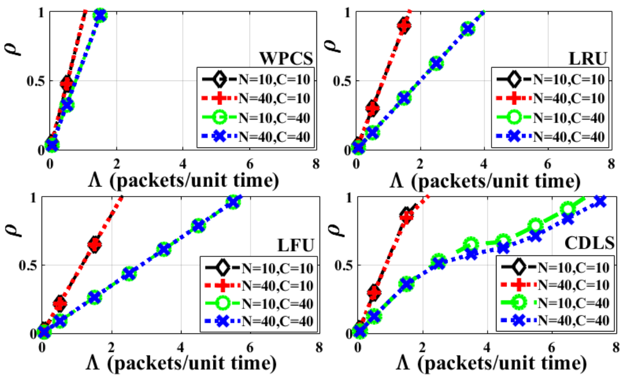


Fig. 5. Effects of the number of caching nodes, N, and the cache size, C, on the utilization factors. *F=100*, and *w=0.1* for WPCS.

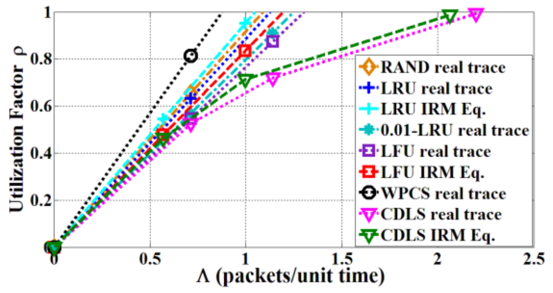


Fig. 6. Performance comparison of different schemes for the real trace-driven experiment and equations based on IRM traffic and $\alpha = 0.6$ Zipf distribution. Network parameters: *F=78.9K, N=360, C=1000.*

Fig. 4.b shows the average response delay, $\bar{D}$, versus Λ for different caching schemes. We consider an LTE-A network with 1 Gbps downlink peak rate, 500 Mbps uplink peak rate [16], files of size B=1 Gb, and 10 Byte for the uplink requests packets size. Therefore, we have Γ=1 file per second. In addition, we consider a typical 10 Gbps rate for the data access to the cache memories [22]. Therefore, the typical delay parameters $\overline{dh} = 10^{-1}(\mathrm{s})$ and $\overline{du} = 10^{-7}(\mathrm{s})$ are considered in this paper. Fig. 4.b shows the transition from stability to eventual instability at Λ_{max}. Due to the instability of PCCS, $\bar{D}$ becomes unbounded for all of the values of Λ and is not shown in this figure. We should mention that the confidence intervals for the desired parameters ρ, $\Lambda_{\max}$ and $\bar{D}$, with confidence level 0.95, are found as $\hat{\rho} \mp 0.04\hat{\rho}$, $\hat{\Lambda}_{\max} \mp 0.02\hat{\Lambda}_{\max}$ and $\hat{\bar{D}} \mp 0.05\hat{\bar{D}}$, respectively. (Note that the hat on top of each character shows the reported value of the related parameter in the numerical results.)

In Fig. 5, we investigate the effect of the number of caching nodes, *N,* and cache sizes, *C,* on the stability behavior. From a design perspective, for a fixed network-wide cache size, i.e. *NC*, designing a system with smaller *N* and larger *C* improves the performance by increasing the hit probability. As a result, the load on the bottleneck link decreases, leading to an increase in the maximum stable throughput. Therefore, in terms of Λ_{max}, increasing *C* always improves the system performance. However, the downside effect of increasing *C* (for a fixed value of *NC*) is the resulting increase in the delay due to an increase in $\overline{dh}$. We should note to the fact that as the number of the requests to a cache increases (due to the smaller *N* for a fixed total throughput), its response time will also increase, leading to larger values of $\overline{dh}$ (due to larger internal collisions between the requests or the internal cache queues). Taking into account all of such intra-cache effects, although important, is beyond the scope of the current paper.

In another perspective, it is evident that for a fixed size *N*, increasing *C* also improves the system performance. However, for a fixed *C*, as *N* is increased in LRU, LFU and WPCS schemes, no change in the overall system performance is observed, as Λ is kept fixed by decreasing λ_{req}. In CDLS, the possibility of exchanging information between the stations through proper coding increases as *N* is increased, leading to a slight increase in the maximum stable throughput as illustrated in Fig. 5. However, as can be observed, the effect of the cache size is dominant in all of the schemes.

To further validate our analysis, we have also run a trace-driven experiment, using a real trace of video clips requests from a campus network measurement on YouTube traffic in 2008 [23], with a total 123.3K requests for 78.9K videos, arriving at 360 distinct stations. Fig. 6 reports the utilization factors and maximum stable throughputs achieved by different caching schemes for the trace-driven experiment. We observe that the results achieved under synthetic traffic still hold when the cache is fed by real traffic taken from an operational network. As shown in Fig.6, in case of the real trace, the performance of different schemes is sorted as WPCS<RAND<LRU<q-LRU<LFU<CDLS, which matches the simulation results shown in Fig. 4.a. Moreover, the proposed CDLS scheme has significantly better performance than the other schemes, approximately by a factor 2, which matches with the results illustrated in Fig. 4.a. It should be noted that according to the discussion under Fig. 5, since in the real trace-driven experiment, the ratio of C/F is less than this ratio in the simulation scenarios, the values of $\Lambda_{\max}$ decreases, but the comparative performance of different schemes are the same as in the simulation results. In addition, we have compared the real trace results with the derived equations for IRM traffic. We have estimated the value of $\alpha = 0.6$ as the exponent parameter of the Zipf distribution for the popularity of the real trace requests. As shown in Fig. 6, the difference between the results of the equations derived in this paper under the assumption of IRM traffic and the results of the real trace is adequately small. Therefore, it validates that the models and derivations proposed in this paper can reasonably represent the performance of different schemes in real applications.

V. Conclusion

In this paper, we have presented the queue models for single bottleneck caching networks and derived the shared link utilization factors and delays for different coded and uncoded schemes. It has been shown that the uncoded caching schemes, guarantee the network stability by limiting the network throughput. On the other hand, earlier coded caching schemes in the literature lead to unstable systems if stochastic models for their requests arrivals are taken into account. Moreover, we have proposed a novel scheme, CDLS, which improves the network performance by decreasing the load on the bottleneck link and increasing the maximum stable throughput. We have shown that the sole use of the coded caching schemes does not lead to the performance improvement and its overall effect on the caching networks should be taken into account.

Appendix (Proofs)

Proof of Lemma 1: According to the definitions of $p_{req}(i)$ and $N_{\tau,i}$, we have

$$p_{req}(i) = 1 - P(N_{\tau,i} = 0) = 1 - G_i(\tau, 0) \qquad (39)$$

Áwhere $G_i(\tau,\xi)$ is the probability generating function of $N_{\tau,i}$. In case of the renewal traffic model, the Laplace transform of $G_i(\tau,\xi)$ is given by [15]

$$G_i^{\,*}(s,\xi) = \frac{1-f_R^*(i,s)}{s(1-\xi f_R^*(i,s))} \qquad (40)$$

where $f_R^*(i,s)$ is the Laplace transform of the PDF of the inter-request time distribution for file f_i.□

Proof of Proposition 1: In case of IRM traffic, the distribution of $N_{\tau,i}$ is Poisson with mean $\lambda_{req}p_i$. So, we have

$$p_{req}(i) = 1 - e^{-\lambda_{req} p_i} \tag{41}$$

Substituting (41) in Theorem 1, results in equation (6).□

Proof of Proposition 2: We use the function $h(x) = x + (1-x)u(x-1)$, where $u(x)$ denotes the step function to model the server's control unit. According to Jensen's inequality for concave functions, we have

$$\rho = \sum_{i=1}^{F} \lambda_{f_i} = \sum_{i=1}^{F} \mathbb{E}[h(X_i)] \le \sum_{i=1}^{F} h(\mathbb{E}[X_i])$$
$$= \sum_{i=1}^{F} h\left(N p_{req}(i)\big(1 - p_{hit}(i)\big)\right) \tag{42}$$

where the RHS results from (4) which proves that the distribution of X_i is $Binom(N, p_{req}(i)(1-p_{hit}(i)))$. Given that $h(x) \le x$, an upper bound for ρ is given by:

$$\rho \le \sum_{i=1}^{F} N p_{req}(i)\big(1 - p_{hit}(i)\big) \tag{43}$$

In case of IRM traffic, according to (41) and given that $1 - e^{-x} \le x$, the upper bound is obtained from

$$\rho \le \sum_{i=1}^{F} N\big(1 - e^{-\lambda_{req} p_i}\big)\big(1 - p_{hit}(i)\big)$$
$$\le \sum_{i=1}^{F} N \lambda_{req} p_i \big(1 - p_{hit}(i)\big) = N \lambda_{req} (1 - p_{hit}) \tag{44}$$

Therefore, for the system stability, i.e. $\rho < 1$, it is sufficient that Λ satisfies the condition $\Lambda < \frac{1}{1-p_{hit}}$. It should be noted that we have $0 \le p_{hit} \le 1$. Therefore, to ensure the stability of the server queue, it is sufficient that $\Lambda \le 1$. In case of $\Lambda \le 1$, the inequality $h(x) \le x$ converts to equality. Hence, we have

$$h\left(N\big(1 - e^{-\lambda_{req} p_i}\big)\big(1 - p_{hit}(i)\big)\right) = N\big(1 - e^{-\lambda_{req} p_i}\big)\big(1 - p_{hit}(i)\big) \tag{45}$$

which shows that the upper bound is tight enough. □

Proof of Corollary 1: In case of the LRU policy and IRM traffic, $p_{hit}(i)$ is given by

$$p_{hit}(i) = 1 - e^{-\lambda_{req} p_i T_c} \tag{46}$$

where T_c is obtained from $C = \sum_{i=1}^{F} (1 - e^{-\lambda_{req} p_i T_c})$[14]. Inserting (46) in Proposition 1 and 2 results in Corollary 1.□

Proof of Corollary 2: In case of the LRU policy and renewal traffic, according to [14], $p_{hit}(i)$ is given by

$$p_{hit}(i) = F_R(i, T_c) \tag{47}$$

Inserting (47) in Theorem 1 results in Corollary 2.□

Proof of Corollary 3: In case of the q-LRU policy and IRM traffic, according to [14], $p_{hit}(i)$ is given by

$$p_{hit}(i) = \frac{q(1 - e^{-\lambda_{req} p_i T_c})}{e^{-\lambda_{req} p_i T_c} + q(1 - e^{-\lambda_{req} p_i T_c})} \tag{48}$$

Inserting (48) in Proposition 1 and 2 results in Corollary 3.□

Proof of Corollary 4: In case of the q-LRU policy and renewal traffic, according to [14], $p_{hit}(i)$ is given by

$$p_{hit}(i) = \frac{q F_R(i, T_c)}{1 + (q-1) F_R(i, T_c)} \tag{49}$$

Inserting (49) in Theorem 1 results in Corollary 4.□

Proof of Corollary 5: The hit probability for file f_i in a cache with the LFU policy is given by

$$p_{hit}(i) = \begin{Bmatrix} 1 & \forall i \in \{1, \dots, C\} \\ 0 & O.W. \end{Bmatrix}, \quad \forall i \in \{1, \dots, F\} \tag{50}$$

Inserting (50) in Proposition 1 and 2 results in Corollary 5.□

Proof of Corollary 6: In case of the LRU policy and IRM traffic, according to [14], $p_{hit}(i)$ is given by

$$p_{hit}(i) = \frac{\lambda_{req} p_i \mathbb{E}[T_c]}{1 + \lambda_{req} p_i \mathbb{E}[T_c]} \tag{51}$$

Inserting it in Proposition 1 and 2 results in Corollary 6.□

Proof of Lemma 2: By definition, T_n^k is the time that the k^{th} request of station n arrives at the server. In case of IRM traffic, $T_1^k, T_2^k, \dots, T_N^k$ are i.i.d r.v.'s with the Erlang distribution with the CDF:

$$F^k(x) = 1 - \sum_{i=0}^{k-1} \frac{(\lambda_{req} x)^i e^{-\lambda_{req} x}}{i!}, x > 0 \tag{52}$$

Since ΔT_N^k is the expectation of the order statistical range of these random variables, according to [20], it is obtained from

$$\Delta T_N^k = \mathbb{E}[T_{N:N}^k - T_{1:N}^k] = \int_0^\infty [1 - \big(F^k(x)\big)^N - \big(1 - F^k(x)\big)^N] dx \tag{53}$$

Inserting (52) in (53) and using the multinomial theorem result in Lemma 2.□

Proof of Lemma 3: As discussed in the proof of Lemma 2, ΔT_N^k is the expectation of the order statistical range of *N* i.i.d r.v.'s with the *Erlang (k,* λ_{req}*)* distribution. According to [24], the lower bound on the expectation of the range of *N* i.i.d r.v.'s is given by $\Delta T_N^k \ge d_N \sigma_k$, where σ_k is the standard deviation of the distribution of these random variables. Therefore, given the standard deviation of the Erlang distribution, $\sigma_k = \frac{\sqrt{k}}{\lambda_{req}}$ [25], the desired bound is derived.

Proof of Proposition 4: If the number of the requests arriving during window *w* is one, no coding is performed on the packets and the server has to send all of the uncached subfiles, whose number is $1 - \frac{C}{F}$. Since during window *w*, the additional waiting time at the server for each request is on average equal to $\frac{w}{2}$, $\bar{s}_w$ in this case is $\frac{w}{2} + (1 - \frac{C}{F})$. Otherwise, the average service time in WPCS is given by

$$\bar{s}_w = \frac{w}{2} + \bar{s}_{w,c} + \bar{s}_{w,unc} \tag{54}$$

which consists of three components: first, the average additional waiting time for each request, i.e. $\frac{w}{2}$, second, the average service time for the coded packets, $\bar{s}_{w,c}$, and third, the average service time for the uncoded packets, i.e. $\bar{s}_{w,unc}$.

We denote the number of the transmittable coded small packets corresponding to the request that arrived at *w* by $N_{w,c}$. Since the number of stations that are served by each coded small packet is $\gamma + 1$, if the number of the requests arriving during window *w* is less than $\gamma + 1$, there is only one coded packet to be sent. Otherwise, $N_{w,c}$ is obtained from $\binom{N_w}{\gamma+1}$. Therefore, $\bar{s}_{w,c}$ is given by

$$\bar{s}_{w,c} = \frac{\frac{N_{w,c}}{\mu_0}}{N_w} = \begin{Bmatrix} \frac{\binom{N_w}{\gamma+1}}{\mu_0 N_w} & if \quad \gamma + 1 \le N_w \\ \frac{1}{\mu_0 N_w} & 1 < N_w < \gamma + 1 \end{Bmatrix} \tag{55}$$

Subsequently, the number of coded packets containing the subfiles of a given file is equal to $N_{w,f} = \binom{N-1}{\gamma}$. The number of transmittable coded packets containing the subfiles of each file is also obtained from

$$N_{w,c,f} = \begin{Bmatrix} \binom{N_w - 1}{\gamma} & if \quad \gamma + 1 \le N_w \\ 1 & 1 < N_w < \gamma + 1 \end{Bmatrix} \tag{56}$$

The average service time for the uncoded packets is then

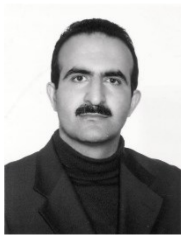
$$\bar{s}_{w,unc} = \frac{N_{w,f}-N_{w,c,f}}{\mu_0}\left\{\begin{matrix} \frac{\binom{N-1}{\gamma}-\binom{N_w-1}{\gamma}}{\mu_0} & if \;\; \gamma+1 \leq N_w \\ \frac{\binom{N-1}{\gamma}-1}{\mu_0} & 1 < N_w < \gamma+1 \end{matrix}\right\} \quad (57)$$

Finally, combining (54), (55) and (57) results in (31).□

References

[1] N. Golrezaei, A. F. Molisch, A. G. Dimakis, and G. Caire, “Femtocaching and device-to-device collaboration: A new architecture for wireless video distribution,”IEEE Communications Magazine, vol. 51, no. 4, pp. 142–149, 2013.
[2] X. Wang, M. Chen, T. Taleb, A. Ksentini, V. C. M. Leung, “Cache in the air: Exploiting content caching and delivery techniques for 5G systems,” IEEE Communications Magazine, 2014.
[3] S. Podlipnig and L. Boszormenyi, “A survey of Web cache replacement strategies”, ACM Computing Surveys, vol. 35, no. 4, 2003.
[4] M. A. Maddah-Ali and U. Niesen,“Fundamental limits of caching,” IEEE Transactions on Information Theory, vol. 60, no. 5, 2014.
[5] U. Niesen and M. A. Maddah-Ali, “Coded caching with nonuniform demands,” in IEEE INFOCOM WKSHPS, 2014.
[6] M. Ji, A. M. Tulinoy, J. Llorcay, and G. Caire,“Order optimal coded caching-aided multicast under Zipf demand distributions,” arXiv:1402.4576v1 [cs.IT] , 2014.
[7] M. Ji, A. M. Tulino, J. Llorcay, and G. Caire,“Order optimal coded delivery and caching: Multiple groupcast index coding,” arXiv:1402.4572v1 [cs.IT] , 2014.
[8] K. Poularakis, G. Iosifidis, L. Tassiulas, “Approximation Algorithms for Mobile Data Caching in Small Cell Networks”, IEEE Transactions on Communications, vol. 62, issue 10, 2014.
[9] M. Ji, G. Caire, and A. F. Molisch, “Fundamental limits of distributed caching in D2D wireless networks,” IEEE Information Theory Workshop (ITW) , 2013.
[10] A. Liu and V. Lau, “Mixed-timescale precoding and cache control in cached MIMO interference network,” IEEE Transactions on Signal Processing, vol. 61, pp. 6320-6332, 2013.
[11] J. M. Wang, J. Zhang, and B. Bensaou, “Intra-AS cooperative caching for content-centric networks,” in SIGCOMM WKSHPS on ICN, 2013.
[12] L. Wang, S. Bayhan, and J. Kangasharju, “Effects of cooperation policy and network topology on performance of in-network caching,” IEEE Communications Letters, vol. 18, pp. 680-683, 2014.
[13] E. Rosensweig, J. Kurose, and D. Towsley, “Approximate models for general cache networks,” in IEEE INFOCOM, 2010.
[14] V. Martina, M. Garetto, and E. Leonardi, “A unified approach to the performance analysis of caching systems,” in IEEE INFOCOM, 2014.
[15] D. R. Cox, *Renewal Theory*, Methuen & Co. LTD,1962.
[16] S. Parkvall, E. Dahlman, A. Furuskär, Y. Jading, M. Olsson, S. Wänstedt, and K. Zangi, “LTE-Advanced – Evolving LTE towards IMT-Advanced,” IEEE Vehicular Technology Conference, 2008.
[17] Y. E. Sagduyuand and A. Ephremides, “On broadcast stability of queue-based dynamic network coding over erasure channels,” IEEE Transaction on Information Theory, vol. 55, no. 12, pp. 5463-5478, 2009.
[18] M. J. Neely, “Stability and Capacity Regions for Discrete Time Queueing Networks”, arXiv:1003.3396v1 [cs.NI] , 2010.
[19] L. Kleinrock, *Queueing Systems: Vol. I,* New York: Wiley Interscience, 1975.
[20] B.C Arnold, N, Balakrishnan, *Relations,Bounds and Approximations for Order statistic,*Lecture Notes in Statistics, Springer,1989.
[21] L. Shi1, Z. Gu, L. Wei, and Y. Shi, “An Applicative Study of Zipf’s Law on Web Cache,” Interntionsl Journal of Information Technology, vol. 12, no.4, 2006.
[22] K. Sohn, T. Na, I. Song, Y. Shim, et al., “A 1.2 V 30 nm 3.2 Gb/s/pin 4 Gb DDR4 SDRAM With Dual-Error Detection and PVT-Tolerant Data-Fetch Scheme,” IEEE Journal of Solid-State Circuits, vol. 48, no. 1, 2013.
[23] M. Zink, K. Suh, Y.Gu and J.Kurose, “Characteristics of YouTube network traffic at a campus network -Measurements, models, and implications”, International Journal of Computer and Telecommunications Networking, vol. 53 no. 4, 2009.
[24] H. O. Hartley and H. A. David, “Universal Bounds for Mean Range and Extreme Observation”,The Annals of Mathematical Statistics, vol. 25, no. 1, pp. 85-99, 1954.
[25] Oliver Ibe, *Fundamentals of Applied Probability and Random Processes,* Elsevier Academic Press, 2005.

Fatemeh Rezaei received the B.Sc. and M.Sc. degrees in electrical engineering from Sharif University of Technology, Tehran, Iran in 2010 and 2012, respectively. She is currently working toward the Ph.D. degree. Her current research interests include communication systems and data networks.



Babak H. Khalaj received his B.Sc. degree from Sharif University of Technology, Tehran, Iran, in 1989, and the M.Sc. and Ph.D. degrees from Stanford University, Stanford, CA, in 1993 and 1996, respectively, all in Electrical Engineering. He joined KLA-Tencor in 1995 as a Senior Algorithm Designer working on advanced processing techniques for signal estimation. From 1996 to 1999, he was with Advanced Fiber Communications and Ikanos Communications. Since then, he has been a Senior Consultant in the area of Data Communications, and a visiting professor at CEIT, San Sebastian from 2006 till 2007. Dr. Khalaj has also been the recipient of Alexander von Humboldt Fellowship in the year 2007–2008. He has also been a senior member of Advanced Communications Research Institute (ACRI) at Sharif University. He was the Co-Editor of the Special Compatibility Standard Draft for ANSI-T1E1 group from 1998 till 1999, and he is the author of two U.S. patents and many papers in signal processing and digital communications.