Scheduling and Buffering the Content in Wireless Networks over the CDNs

[#1] Kancharla Bhanu Kumar, [#2] Maddirala Syam, [#3]Sk.Abdul Rasheed

[1]*M.Tech Student, NEC, Narasaraopet*
Guntur Dist, A.P, India
[2, 3] *Asst.Professor, NEC, Narasaraopet,*
Guntur Dist, A.P, India
1 Bhanu.kumar567@gmail.com
2 syam.jnsa@gmail.com
3 rasheed4321@gmail.com

***Abstract*—The rapid growth of wireless content access implies the need for content placement and scheduling at wireless base stations. We study a system under which users are divided into clusters based on their channel conditions, and their requests are represented by different queues at logical front ends. Requests might be elastic or inelastic correspondingly, we have request queues that indicate the number of elastic requests, and deficit queues that indicate the deficit inelastic service. Caches are of finite size and can be refreshed periodically from a media vault. We consider two cost models that correspond to inelastic requests for streaming stored content and real-time streaming of events, respectively. We design provably optimal policies that stabilize the request queues (hence ensuring finite delays) and reduce average deficit to zero [hence ensuring that the quality-of-ser- vice (QoS)target is met]at small cost. We illustrate our approach through simulations.**

***Index Terms*—Content distribution network (CDN), delay-sensitive traffic, prediction, quality of service (QoS), queuing, buffering.**

I.INTRODUCTION

The past few years have seen the rise of smart hand held wireless devices as a means of content consumption .Content might include streaming applications in which chunks of the file must be received under hard delay constraints, as well as file downloads such as software updates that do not have such hard constraints. The core of the Internet is well provisioned, and network capacity constraints for content delivery area the media vault (where content originates) and at the wireless access links at end-users. Hence a natural location to place caches for a content distribution network (CDN) would beat the wireless gateway, which could be a cellular base station through which users obtain network access.

Network between the caches to the users has finite ca-pacity;2)each cache can only host a finite amount of content; and3)refreshing content in the caches from the media vault incurs a cost. Users can make two kinds of requests, namely: 1) elastic requests that have no delay constraints, and2) inelastic requests that have a hard delay

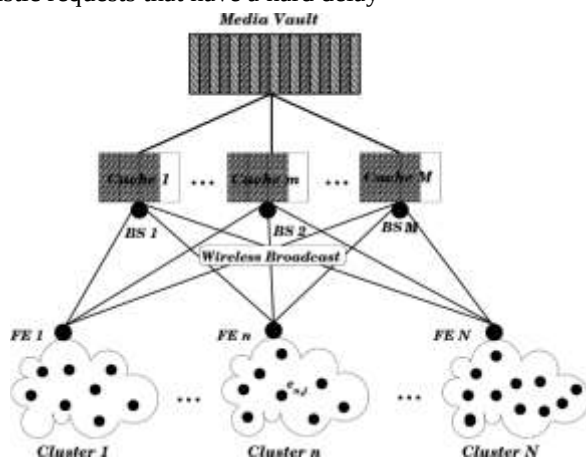

Fig.1. Wireless content distribution.

An abstraction of such a network is illustrated in Fig.1.There are multiple cellular base stations (BSs), each of which has a cache in which to store content. The content of the caches can be periodically refreshed through accessing a media vault. We divide users into different clusters, with the idea that all users in each cluster are geographically close such that they have statistically similar channel conditions and are able to access the same base stations. Note that multiple clusters could be present in the same cell based on the dissimilarity of their channel conditions to different base stations. There quests made by each cluster are aggregated at a logical entity that we call a frontend (FE) associated with that cluster. The frontend could be running on any of the devices in the cluster or at a base station, and its purpose is to keep track of the requests associated with the users of that cluster. The following constraints affect system operation:1)the wireless For inelastic requests, we adopt the

model proposed wherein users request chunks of content that have a strict dead line, and the request is dropped if the dead line cannot be met. The idea here is to meet a certain target delivery ratio which could be something like"90%of all requests must be met to ensure smooth play out."Each time an inelastic request is dropped, a deficit queue is updated by an amount proportional to the delivery ratio. We would like the average value of the deficit to be zero. In this paper, we are interested in solving the joint content Placement and scheduling problem for both elastic and inelastic traffic in wireless networks.

A.Related Work

The problem of caching and content scheduling has earlier been studied for online Web caching and distributed storage systems. A commonly used metric is a competitive ratio of misses, assuming an adversarial model. Examples of work in this context. Load balancing and placement with linear communication costs are examined. Here, the objective is to use distributed and centralized integer programming approaches to minimize the costs. However, this work does not take account for network capacity constraints, delay sensitive traffic.

The techniques that we will employ are based on the literature on scheduling schemes. Tassiulas *et al.* proposed the Max Weight scheduling algorithm for switches and wireless networks in their seminal work. They proved that this policy is throughput-optimal and characterized the capacity region of the single-hop networks as the convex hull of all feasible schedules. These papers explore the delays in the system for single downlink with variable connectivity, multi rate links, and multi hop wireless flows. However, these do not consider content distribution with its attendant question of content placement. Closest to our work in which, however, only considers elastic traffic and has no results on the value of prediction.

B. Main Results

In this paper, we develop algorithms for content distribution with elastic and inelastic requests. We use a request queue to implicitly determine the popularity of elastic content. Similarly, the deficit queue determines the necessary service for inelastic requests. Content may be refreshed periodically at caches. We study two different kinds of cost models, each of which is appropriate for a different content

• We first characterizes the capacity region of the system and develop feasibility constraints that any stabilizing algorithm must satisfy. Here, by stability we mean that elastic request queues have a finite mean, while inelastic deficit values are zero on average.

• We develop a version of the max-weight scheduling algorithm that we propose to use for joint content placement and scheduling. We show that it satisfies the feasibility constraints and, using a Lyapunov argument, also show that it stabilizes the system of the load within the capacity region. As a by-product, we show that the value of knowing the arrival rates is limited in the case of elastic requests, while it is not at all useful in the inelastic case.

• We next study another version of our content distribution problem with only inelastic traffic, in which each content has an expiration time. We assume that there is a cost for replacing each expired content chunk with a fresh one. For this model, we first find the feasibility region and, following a similar technique to develop a joint content placement and scheduling algorithm that minimizes the average expected cost while stabilizing the deficit queues.

• We illustrate our main insights using simulations on a simple wireless topology and show that our algorithm is indeed capable of stabilizing the system. We also propose two simple algorithms, which are easily implementable, and compare their performance to the throughput-optimal scheme.

II. System Model

There is a set of base stations M and each base station is associated with a cache. we let N denote the set of these clusters. Also, as discussed in the Introduction, there are front ends in each cluster, also denoted n € N by whose purpose is to aggregate requests from the users. Time is slotted, and we divide time into frames consisting of D time-slots. Requests are made at the beginning of each frame. There are two types of users in this system inelastic and elastic based on the type of requests that they make. Requests made by inelastic users must be satisfied within the frame in which they were made. Elastic users do not have such a fixed deadline, and these users arrive, make a request, are served, and depart.

The base stations employ multiple access schemes (e.g., OFDMA), and hence each base station can support multiple simultaneous unicast transmissions, as well as a single broadcast transmission. It is also possible to study other scenarios (e.g., multicast transmissions to subsets of users) using our framework. We adopt a slow-fading packet erasure model for the wireless channels. Accordingly, the channel between cache m and user u(or front end n) is modeled as a Stochastic ON- OFF process

Content is partitioned into two disjoint sets of inelastic content I and elastic content E .We denote the set of inelastic users by u€U At the beginning of each frame , each inelastic user makes at most one request . The idea is that an inelastic request must either be satisfied by the end of the frame or dropped. Inelastic requests are served using broadcast transmissions. We model this request by a Bernoulli process with the mean value λu.

au(k)=1,with probability λu

au(k)=0,with probalbilty 1- λu (1)

Note that while the Bernoulli process models an inelastic request for each user, the distribution of the requests over different content types can be chosen arbitrarily. Since there are limited resources in the system, all requests cannot be served. In order to provide enough service to each user, we need to decide on a minimum delivery ratio for inelastic users. The delivery ratio is the proportion of inelastic requests that are served, and hence the expected service required by user u is nu λu , in which nu is the minimum acceptable delivery ratio. This model follows that of [2] and is consistent with the idea that streaming media can tolerate a fraction of chunk losses, but has hard delay constraints on the received chunks.

We further assume that arrivals are independently and identically distributed over frames. An elastic request that does not get served during a frame will be enqueued and wait for the service during the next frames. However, we need to make sure that the request queue lengths in each cluster remain bounded as time passes so that the delay does not become unboundedly large. Thus, we require that the expected elastic service for content in cluster is. Furthermore, in order to ensure that these requests are not served with infinite periodicity, we assume that each must be served using a unicast transmission.

Each cache m has a finite capacity of chunks of content. In what follows, constraint is to consider different timescales for cache reloading and request arrivals. Hence, lower capacity can be modeled as a slower timescale for refreshing cache contents. For simplicity, we use the same timescale (i.e., frames) for request arrivals and refreshing cache contents. Hence, base stations can reload their caches with new content at the beginning of each frame. The same framework can be used to study the general case at the expense of more computational complexity. In Section V, we explicitly model the reloading cost for a variation of our caching model. For this model, we assume the content of the caches expires and will not be useful at the end of each frame. However, placing each chunk in a cache induces a cost. Therefore, in order to reduce the cost, we may occasionally choose to reload a cache partially and not

III PURE UNICAST ELASTIC SCENARIO

In this section, we assume there are only requests for elastic content. As noted in Section II, these requests are to be served using unicast communications. For notational convenience, we assume that transmissions are between base stations and front ends, rather than to the actual users making the requests. We first determine the capacity region, which is the set of all feasible requests. Note that this model, in which front ends have independent and distinct channels to the caches, differs from the previously studied wired caching systems (see, e.g., [13]) because the wireless channels are not always ON. Therefore, the placement and scheduling must be properly coordinated according to the channel states.

A. Capacity Region

Let $p_e^m(k) \in \{0, 1\}$ denote the presence of content e € E at cache m, that is $p_e^m(k) = 1$ if e is present in cache m at frame k , and $p_e^m(k) = 0$,otherwise. The cache capacity constraint requires each cache to satisfy

$$\sum_{e \in E} p_e^m(k) \leq v \quad \text{for each frame k} \qquad (2)$$

The scheduled service to content e at front end n, which is provided by cache m during frame k, is indicated using $s^m_{n,e}(k)$. The actual service depends on the presence of the corresponding content and the corresponding channel state, and one can verify that would be $s^m_{n,e}(k)\ p^m_e(k)\ c^m_e(k)$. Note that $\sum_e s^m_{n,e}(k) \leq D$ because each channel can transmit at most one chunk during a time-slot, when it is ON, and each frame consists of slots. In general $p^m_e(k)$ and $s^m_e(k)$, and may follow the joint distribution of some random processes and at each frame . Since the processes of request arrivals and channel states are assumed to be i.i.d. over frames, it suffices (for the purpose of defining the capacity region) to only consider stationary processes, which are independent of time. In order to achieve the capacity region, the content placement and service scheduling rely on the realization of the channel states.

TABLE 1
SUMMARY OF NOTATION

Notation	Definition
$\mathcal{U}$	Set of all inelastic users u
$\mathcal{N}$	Set of all clusters (also front ends) n
$\mathcal{M}$	Set of all caches m
$\mathbf{I}$	Set of all inelastic contents i
$\mathbf{E}$	Set of all elastic contents e
$v_m = v$	Capacity of cache m
D	Duration of a frame (in terms of time slots)
$a_u(k)$	Number of requests by user u in frame k
$a_{u,i}(k)$	Number of requests by u for content i in frame k
λ_u	Inelastic request rate by user u
η_u	Minimum delivery ratio required by user u
$a_{n,e}(k)$	Number of requests at front end n for content e in frame k
$\lambda_{n,e}$	Elastic request rate at front end n for content e
$A(k)$	Vector of all elastic and inelastic requests in frame k
$c_{\ell}^m(k)$	State of the channel between m and u in frame k
$C(k)$	Vector of all channel states in frame k
$p_e^m(k)$	Presence of content e in cache m during frame k
$s_{n,e}^m(k)$	*Scheduled* service to e at front end n, by m in frame k
$s_{n,e}(k)$	*Actual* service to content e at n, in frame k
$q_{n,e}(k)$	The request queue length for content e at n, in frame k
$s_i^m(k)$	*Scheduled* service to i by cache m, in frame k
$s_u(k)$	*Actual* service to inelastic user u, in frame k
$S(k)$	Vector of all scheduled service and placement in frame k

Therefore, the processes p_e^m(k) c_e^m(k). and depend on c_e^m(k). and can be accordingly denoted by $s_{n,e}^m$(k) p_e^m(k) c_e^m(k). . A necessary and sufficient condition on (strict) feasibility of the set of requests{λn,e:n€N,e€E} can be expressed as follows:

$$\exists \text{ a set of random variables } \mathbf{p}_e^m(\mathbf{c}_n^m), \mathbf{s}_{n,e}^m(\mathbf{c}_n^m) \text{ such that}$$

$$\lambda_{n,e} < \mathbb{E}\left[\sum_m \mathbf{c}_n^m \mathbf{p}_e^m(\mathbf{c}_n^m)\mathbf{s}_{n,e}^m(\mathbf{c}_n^m)\right] \quad \forall e,n$$

$$\sum_e \mathbf{s}_{n,e}^m(\mathbf{c}_n^m) \le D \quad \forall m,n$$

$$\sum_e \mathbf{p}_e^m(\mathbf{c}_n^m) \le v \quad \forall m \qquad (3)$$

where denotes the expectation. Our objective is to provide placement and scheduling algorithms that can fulfill any set of strictly feasible requests.

B. Value of Prediction

Suppose we know the statistics of the elastic requests, i.e., the values of are known. The question is whether this information would help in designing a throughput-optimal caching and scheduling scheme. Using the capability to predict requests, we could potentially decide on the elastic content distribution scheme a priori. Notice that this is

Equivalent to solving (3) to find the appropriate joint distribution of the content placement and the service schedule. The solution would yield a set of caching and scheduling choices, and a probability with which to use each one based on channel realizations.While such an algorithm is very simple to implement, solving (3) for the set of schedules is quite hard. Consequently, we see that prediction of the elastic requests has limited value in the context of devising appropriate content distribution algorithms. We will see in Section IV that prediction is even less useful for the case of inelastic requests.

C. Throughput-Optimal Scheme

Since it is hard to realize an offline prediction, placement and scheduling scheme, we now study our system of elastic requests in a queueing context. The development here is similar to the traditional switch scheduling problem, as relevant to our model. We assume the elastic requests in cluster go through a set of request queues whose lengths at frame are denoted by for each content , and follow the dynamic

$$q_{n,e}(k+1) = q_{n,e}(k) + a_{n,e}(k) - s_{n,e}(k) \qquad (4)$$

where , $s_{n,e}$(k)-min($q_{n,e}$(k+)∑m $s_{n,e}^m$(k) p_e^m(k) c_e^m(k). and $q_{n,e}$(k+1)= $q_{n,e}$(k)+$a_{n,e}$(k) Note $s_{n,e}^m$(k) that is the total number of requests for content at front end that are served during frame k . One can verify that

$$q_{n,e}(K+1) = \sum_{k=1}^{K} a_{n,e}(k) - \sum_{k=1}^{K} s_{n,e}(k). \qquad (5)$$

The evolution of these request queues can be studied by a Markov chain @, whose state at each frame is the vector of ($q_{n,e}$(k):e € E n € N) If n @ is shown to be stable (positive recurrent) under some policy , then for all k , and subsequently [from (5)]

$$\lim_{K\to\infty} \frac{1}{K}\mathbb{E}[q_{n,e}(K)] = \lambda_{n,e} - \lim_{K\to\infty} \frac{1}{K}\mathbb{E}\left[\sum_{k=1}^{K} s_{n,e}(k)\right] = 0$$

Algorithm 1: Optimal Content Placement and Scheduling of Elastic Requests

At the beginning of each frame k: given the queue lengths $q_{n,e}(k)$, and the arrivals $a_{n,e}(k)$, let $q_{n,e} = q_{n,e}(k) + a_{n,e}(k)$.

Content placement:

At each cache m, solve the following maximization problem to find the optimal placement $(p_e^m)^*$:

$$\begin{aligned} \max \quad & \sum_{e,n} c_n^m q_{n,e} \alpha_{n,e} \\ \text{s.t.} \quad & \alpha_{n,e} \le p_e^m \quad \forall n,e \\ & \sum_e \alpha_{n,e} \le 1 \quad \forall e \\ & p_e^m \in \{0,1\} \quad \forall e \\ & \sum p_e^m \le v \end{aligned} \qquad (6)$$

in which c_n^m. values denote the channel states during this frame and are given.

Service scheduling:

For each cache m and front end n , determine the optimal Schedule($s_{n,e}^m$) *as follows:

$$(s_{n,e}^m)^* = \begin{cases} D, & \text{if } e = rand\left(\arg\max_{f\in\mathbb{E}} \left((p_f^m)^* c_n^m q_{n,f}\right)\right) \\ 0, & \text{otherwise.} \end{cases}$$

Thus, the capacity of the link between cache m and front end n is completely devoted to serve one of the contents (randomly chosen) that maximizes $p_f^m(k)^*$ c_n^mq n f. .
where is the average expected service, and hence the requests for content at front end are fulfilled using policy . Our objective in this section is to find such a policy , which determines the scheduled service and the content placement for each frame , and results in being stable for any set of feasible requests. Next, we will present a content placement and a service scheduling scheme in Algorithm 1.
Theorem 1 verifies the throughput optimality of these policies.
Theorem 1: The placement and scheduling scheme in Algorithm 1 can fulfill any set of requests satisfying the conditions in (3). Therefore, the proposed scheme is= throughput-optimal.
Proof: We will show the stability of Markov chain using
The Lyapunov criterion [15]. Consider the candidate Lyapunov function. We will show that for any set of strictly feasible requests, the scheme in Algorithm 1 will result in an expected drift Δ v(k)=

$$\mathbb{E}\left[V(k+1) - V(k) | \text{state of the system at frame } k\right]$$

which is negative except in a finite subset of the state space.
The expected drift can be written as

$$\begin{aligned}
\Delta V(k) &= \mathbb{E}[V(k+1) - V(k) | q_{n,e}(k^+) = q_{n,e}] \\
&= \frac{1}{2}\mathbb{E}\left[\sum_{n,e}(q_{n,e} + a_{n,e}(k+1) - s_{n,e}(k))^2 - (q_{n,e})^2\right] \\
&= \mathbb{E}\left[\sum_{n,e} q_{n,e}(a_{n,e}(k+1) - s_{n,e}(k))\right] \\
&\quad + \frac{1}{2}\mathbb{E}\left[\sum_{n,e}(a_{n,e}(k+1) - s_{n,e}(k))^2\right] \\
&\overset{(a)}{=} \sum_{n,e} q_{n,e}\lambda_{n,e} - \mathbb{E}\left[\sum_{n,e,m} q_{n,e} c_n^m(k) p_e^m(k) s_{n,e}^m(k)\right] + B
\end{aligned}$$

in which (a) follows since , $s_{n,e}$(k)-min($q_{n,e}$(k+)∑m $s_{n,e}^m$(k) p_e^m(k) c_e^m(k) , and B is shown to have a finite value (please refer to Appendix-A).

We will show (in Appendix-B) the optimal values(p_e^m)* and ($s_{n,e}^m$)* chosen by Algorithm 1 are indeed the solution to the following problem:

$$\begin{aligned}
\max \quad & \sum_{n,e,m} q_{n,e} c_n^m(k) p_e^m(k) s_{n,e}^m(k) \\
& \sum_e p_e^m(k) \le v \qquad \forall m \\
& \sum_e s_{n,e}^m(k) \le D \qquad \forall m, n. \qquad (7)
\end{aligned}$$

Consequently,

$$\sum_{n,e,m} q_{n,e} c_n^m(k)(p_e^m)^*(s_{n,e}^m)^* \ge \sum_{n,e,m} q_{n,e} c_n^m(k) p_e^m s_{n,e}^m$$

For any set of values $p_e^m$and ($s_{n,e}^m$) ,satisfying the constraints in (7). Specifically, these values can be chosen as the random processes, [defined in (3)]. Hence, we have

$$\begin{aligned}
&\mathbb{E}\left[\sum_{n,e,m} q_{n,e} c_n^m(k)(p_e^m)^*(s_{n,e}^m)^*\right] \\
&\ge \mathbb{E}\left[\sum_{n,e} q_{n,e} \sum_m \mathbf{c}_n^m \mathbf{p}_e^m(\mathbf{c}_n^m)\, \mathbf{s}_{n,e}^m(\mathbf{c}_n^m)\right] \\
&\overset{(b)}{\ge} \sum_{n,e} q_{n,e}\lambda_{n,e}(1+\epsilon) \qquad (8)
\end{aligned}$$

in which the expectation is with respect to the distribution of ($c_{n,e}^m$) , and random processes ($p_{n,e}^m$) (c_n^m)and($s_{n,e}^m$) ($c_{n,e}^m$) , and follows from (3) for a sufficiently small . Considering the above relation in the right hand side (RHS) of results in a drift

$$\Delta V(k) \le -\epsilon \sum_{n,e} q_{n,e} + B \qquad (9)$$

which is negative for large enough queue length values q n,e , and the Lyapunov theorem implies the stability of the request queues.

In Section IV, we will see that the case of inelastic requests is different and the prediction has even less significance on the scheduling of the inelastic content distribution network.

IV. JOINT ELASTIC-INELASTIC SCENARIO

In this section, we study the general case where elastic and inelastic requests coexist in the system. Recall that the elastic requests are assumed to be served through unicast communications between the caches and front ends, while the base stations broadcast the inelastic contents to the inelastic users. We further assumed servers can employ OFDMA method to simultaneously transmit over their single broadcast and multiple unicast channels. Although these two types of traffic do not share the access medium, all the content must share the common space in the caches. Consequently, we

require an algorithm that jointly solves the elastic and inelastic scheduling problems. In this section, we first determine the general capacity region of the system and then present our algorithm.

A. Joint Elastic-Inelastic Capacity Region

Similar to the case of elastic content, we use $p_i^m(k)$ to denote the presence of inelastic content i € I in cache during frame . Since the channel states do not change during a frame, and there is at most one request (a u, i(k) € {0,1} for inelastic content by each user , each cache may schedule to broadcast content at most once per frame. We let $s_{n,e}^m$ € {0,1}represent this scheduled service for frame. Note that each cache can broadcast at most contents during a frame, hence we require

$$\sum_i s_i^m(k) \le D \qquad \forall m, k. \tag{10}$$

The actual inelastic service, provided to user u for content i, depends not only on the channel states and the cache presence, but also on whether there is a new (not expired) request for content . It should be straightforward to verify that the total actual inelastic service provided to user u during frame k is

$$s_u(k) = \sum_i \min\left(a_{u,i}(k), \sum_m c_u^m(k) p_i^m(k) s_i^m(k)\right). \tag{11}$$

For each frame k, we also denote the vector of all request arrivals by

$$A(k) = (a_{u,i}(k), a_{n,e}(k) : u \in \mathcal{U}, n \in \mathcal{N}, i \in \mathbf{I}, e \in \mathbf{E})$$

The channel states using

$$C(k) = (c_n^m(k), c_u^m(k) : m \in \mathcal{M}, u \in \mathcal{U}, n \in \mathcal{N})$$

And the scheduled service and placement by

$$S(k) = (s_{n,e}^m(k), s_i^m(k), p_e^m(k), p_i^m(k) : \forall m, u, n, i, e).$$

Note that a legitimate schedule S(k) must satisfy

$$\sum_{f \in \mathbf{E} \cup \mathbf{I}} p_f^m(k) \le v \qquad \forall m$$
$$\sum_i s_i^m(k) \le D \qquad \forall m$$
$$\sum_e s_{n,e}^m(k) \le D \qquad \forall m, n$$
$$s_i^m(k) \in \{0,1\} \qquad \forall m, i. \tag{12}$$

Since the channel states and the request arrivals are identically and independently distributed over frames, following the same argument as in [16], we can formally define the capacity region of the joint elastic and inelastic system based on the existence of a randomized stationary policy.

Definition 1 (Capacity Region of the Joint Scenario):

A set of elastic requests and inelastic requests (with corresponding expected delivery ratios) are (strictly) feasible if the following holds. There exists a policy that, during each frame, given and , chooses a schedule among all legitimate schedules with respect to a probability distribution , such that

$$\eta_u \lambda_u < \mathbb{E}[s_u(k)] \qquad \forall u$$
$$\lambda_{n,e} < \mathbb{E}[\sum_m p_e^m(k) c_n^m(k) s_{n,e}^m(k)] \qquad \forall e, n \tag{13}$$

where the expectation is over the randomness of the arrival processes, channel states, and the probability distribution P(S(k) |A(k), C(k))

.

B. Joint Throughput-Optimal Scheme

In Section III, we studied the elastic traffic using request queues . For the inelastic requests, we now define a deficit queue for each user that captures the accumulated unhappiness of the user about the provided inelastic service. denotes the length of the corresponding deficit queue at frame and follows

$$d_u(k+1) = d_u(k) + \sum_i \tilde{a}_{u,i}(k) - s_u(k) \tag{14}$$

where with probability , and it is zero otherwise. Note that the deficit queue is a virtual queue whose length can be negative. A negative length shows up when the provided inelastic service is greater than the required service.

$$\lim_{K\to\infty} \mathsf{E}\left[\frac{1}{K}d_u(K)\right] = \eta_u\lambda_u - \lim_{K\to\infty}\frac{1}{K}\mathsf{E}\left[\sum_{k=1}^{K} s_u(k)\right]. \quad (15)$$

We will present a joint scheme in Algorithm 2 that can stabilize the requests queues and the (positive part of) deficit queues for any feasible set of requests Therefore

$$\lim_{K\to\infty} \mathsf{E}[\frac{1}{K}d_u(K)] \le \lim_{K\to\infty} \mathsf{E}[\frac{1}{K}\{d_u(K)\}^+] = 0 \quad (16$$

Hence, the joint scheme can fulfill all feasible inelastic requests in addition to also satisfying all elastic requests. It is possible to simplify this algorithm by noting that we only need to

Algorithm 2: Joint Elastic-Inelastic Scheduling and Placer Scheme

At the beginning of frame k, given the queue lengths, arrivals, and the channel states, let $d_u = d_u(k) + \sum_i \tilde{a}_{u,}$ $q_{n,c} = q_{n,c}(k) + a_{n,c}(k)$.
Solve the following maximization problem to find the optimal schedule $S^*(k)$:

$$\max \sum_u \{d_u\}^+ \sum_i \min\left(a_{u,i}, \sum_m c_u^m p_i^m s_i^m\right) + \sum_{c,n,m} q_{n,c} c_n^m p_c^m s_{n,c}^m$$

subject to (12).

cache content that is scheduled to be served in a frame. We present a simplified version of Algorithm 2 in Appendix-D for completeness.

Theorem 2: The joint scheme in Algorithm 2 is throughput optimal. That is, it can fulfill any set of strictly feasible elastic and inelastic request. In the proof of this theorem (see Appendix-C), we show that by applying Algorithm 2, the deficit and request queues are stable. Therefore, the corresponding Markov chain is positive recurrent and converges to a unique steady state.

The following corollary (refer to Appendix-E for the proof)
provides a bound on the queue lengths at the steady state.

Corollary 1: Sum of the average request and deficit queue lengths at the steady state satisfies

$$\sum_u \mathsf{E}[d_u] + \sum_{n,c} \mathsf{E}[q_{n,c}] \le \frac{1}{2\epsilon}\left(\sum_{n,c}\sigma_{n,c}^2 + 3|\mathcal{N}||\mathcal{M}|^2D^2 + |\mathcal{U}|\right) \quad (18)$$

for some $\epsilon > 0$ that determines how close to the boundary of the capacity region the requests are.

We now discuss whether prediction is useful in the case of inelastic traffic. The service to an inelastic user is subject to the existence of a new unexpired request. In case there is a valid request, we can only reduce the deficit of a user by at most 1 unit. In other words, even if a user's deficit is large, it cannot be reduced by a large amount by scheduling that user multiple times during a frame. This property of inelastic traffic reduces the value of request prediction in the sense that the content placement and scheduling must be done in a complete accordance to the realization of the channel states as well as the new request arrivals. Hence, planning for the cache placement of the inelastic content cannot be performed in advance. Moreover, the capacity region of the inelastic content distribution network in general has a complicated form (as in Lemma 1), which requires dealing with probability distributions. As a result, even foreseeing the required amount of cache resources is not straightforward. Hence, we conclude that prediction of arrival rates for inelastic traffic is of marginal value.

V. INELASTIC CACHING WITH CONTENT EXPIRY

In this section, we study an inelastic caching problem where the contents expire after some time. In this new model, which is compatible with real-time streaming of live events, we only

consider inelastic traffic and assume that the lifetime of an inelastic content is equal to the length of a frame. Hence, we can cache a content only for the duration of a frame after which the content will not be useful any longer.

We propose a new model for cache refresh cost that is consistent with this scenario, in which loading a cache during frame incurs a cost of per content. is a random variable identically and independently distributed over frames, with the average of for all . The total cost of replacing new contents in the caches, at frame , is denoted by , where

$$\Gamma(k) = \sum_{m,i} \gamma^m(k) p_i^m(k) \quad (19)$$

and denotes the presence of a *fresh* chunk of content in cache for the th frame. Our objective is to find a policy that stabilizes the deficit queues in the system at the minimum long-time average expected cost of cache replacement .

The following lemma indicates the existence of a randomized stationary policy that can fulfill any set of feasible requests at the minimum average cost.

Lemma 3: For any set of feasible inelastic requests, there exists a randomized stationary policy such that at frame , given the arrivals , channel states , and the costs , itchooses

a legitimate schedule according to a probability distribution

$$\mathbb{P}^*(S(k)|A(k), C(K), \{\gamma^m(k) : m \in \mathcal{M}\})$$

On average, R* provides enough service, that is

$$\mathbb{E}[s_u(k)] \geq \eta_u \lambda_u \qquad (20)$$

at the minimum average expected cost .
The above expectations are with respect to the randomness of the request arrivals, loading costs, channel states, and the probability distribution used by policy . The proof follows the same argument as in [14] and is omitted for brevity.

A.Minimum-Cost Throughput-Optimal Policy

Following a similar queueing analysis as in the previous sections, we consider a deficit queue for each user u . Our objective is to find a policy that stabilizes these deficit queues while minimizing the average cost. To achieve this goal, our framework is to minimize an upper bound on expected (Lyapunaov drift +cost) at each frame. The resulting scheme is presented in Algorithm 3, and Theorem 4 evaluates its performance.
Theorem 4: The proposed scheme stabilizes the deficit queues for any set of feasible requests, and hence is throughput optimal. Moreover, it incurs a long-time average expected cost that deviates from the minimum cost by an amount less than $\frac{|u|}{2Y}$

$$\hat{\Gamma} \leq \Gamma^* + \frac{|\mathcal{U}|}{2Y}.$$

Observation 1: By increasing the control parameter , we can achieve an average expected cost that is arbitrarily close to the minimum cost. However, this will potentially lead to larger expected deficit queue lengths. Hence, there is a tradeoff between the cost and the average deficit queue lengths.
Proof: Consider the Lyapunov function as mentioned before, we will try to minimize an upper bound on the expected sum of the Lyapunov drift and cost

$$\mathbb{E}[V(k+1) - V(k)|d_u(k)] + Y\mathbb{E}[\Gamma(k)|d_u(k)]$$

Algorithm 3: Inelastic Traffic with Expiry: Cost-effective Scheme

At the beginning of each frame k, given the queue lengths, arrivals, channel states and the refresh costs $\gamma_m(k)$, let $d_u = d_u(k) + \sum_i \tilde{a}_{u,i}(k)$.
Solve the following maximization problem to find the optimal placement and schedule $(p_i^m(k))^* = (s_i^m(k))^*$:

$$\begin{aligned} \max \quad & \sum_{u,i} \{d_u\}^+ \min\left(a_{u,i}(k), \sum_m c_u^m s_i^m(k)\right) \\ & - Y \sum_{m,i} \gamma^m(k) s_i^m(k) \\ \text{subject to} \quad & \sum_i s_i^m(k) \leq \min(D, v) \qquad \forall m \\ & s_i^m(k) \in \{0,1\} \qquad \forall m, i. \end{aligned}$$

Where Y >0 is a control parameter to trade off cost with performance.
From the analysis in appendix-c we have

$$\begin{aligned} &\mathbb{E}[\Delta V(k)|d_u(k)] + Y\mathbb{E}[\Gamma(k)|d_u(k)] \\ &\leq \frac{|\mathcal{U}|}{2} + Y\mathbb{E}[\Gamma(k)|d_u(k)] + \sum_u \{d_u(k)\}^+ \eta_u \lambda_u \\ &\quad - \sum_u \{d_u(k)\}^+ \mathbb{E}[s_u(k)]|d_u(k)]. \qquad (21) \end{aligned}$$

At each frame k, given the arrival s,channel states and the costs $_{\gamma}$m (k), we minimize

$$Y \sum_{m,i} \gamma^m(k) p_i^m(k) - \sum_u \{d_u(k)\}^+ s_u(k) \qquad (22)$$

Overall legitimate schedules to get $p_i^m(k)$ and s_u (k).therfore, we will have

$$\begin{aligned} &\frac{|\mathcal{U}|}{2} + Y\mathbb{E}\left[\sum_{m,i} \gamma^m(k) \tilde{p}_i^m(k)|d_u(k)\right] + \sum_u \{d_u(k)\}^+ \eta_u \lambda_u \\ &- \sum_u \{d_u(k)\}^+ \mathbb{E}[\hat{s}_u(k)|d_u(k)] \\ &\leq \frac{|\mathcal{U}|}{2} + Y\mathbb{E}^*[\Gamma(k)] + \sum_u \{d_u(k)\}^+ (\eta_u \lambda_u - \mathbb{E}^*[s_u(k)]) \end{aligned} \qquad (23)$$

Where the right-hand side is what the randomized stationary Policy R* achieves. If the requests are strictly feasible, then for
All u $\sum \epsilon > 0$,Ê *[Su(k)] >nu λ u +€ such that . Considering this fact and in (23) and (21) gives

$$\mathbb{E}[\hat{V}(k+1)-\hat{V}(k)|d_i^m(k)]+Y\mathbb{E}\left[\sum_{m,i}\gamma^m(k)\hat{p}_i^m(k)|d_u(k)\right] \le \frac{|\mathcal{U}|}{2}+Y\Gamma^*-\epsilon\sum_u\{d_u(k)\}^+ \quad (24)$$

Where V(k) is the value of the Lypaunov function at frame k when we use our proposed schedule. It is clear that for large enough values, the expected drift is negative, and hence the scheme is stabilizing the deficit queues. Note that (24) holds for any frame . We take expectation from both sides of this inequality,with respect to the distribution of the deficit queues , to get

$$\mathbb{E}[\hat{V}(k+1)-\hat{V}(k)]+Y\mathbb{E}[\sum_{m,i}\gamma^m(k)\hat{p}_i^m(k)] \le \frac{|\mathcal{U}|}{2}+Y\Gamma^*. \quad (25)$$

Assume the initial deficit queue lengths are zero i.e., V(0)=0
Now sum both sides of(25) from k = 0 to k = k and divide by k+1 to get

$$\frac{\mathbb{E}[\hat{V}(K+1)]}{K+1}+\frac{Y}{K+1}\sum_{k=0}^{K}\mathbb{E}\left[\sum_{m,i}\gamma^m(k)\hat{p}_i^m(k)\right] \le \frac{|\mathcal{U}|}{2}+Y\Gamma^*. \quad (26)$$

By letting K tend to infinity and nothing that E[v(K+1)] is a bounded positive value for each K, we get

$$\hat{\Gamma}=\lim_{K\to\infty}\frac{1}{K+1}\sum_{k=0}^{K}\mathbb{E}[\hat{\Gamma}(k)] \le \Gamma^*+\frac{|\mathcal{U}|}{2Y}. \quad (27)$$

Note that in the studied model, we may fetch a fresh chunk of content at frame only if it is scheduled to be served because otherwise it gets expired and becomes useless by the end of this frame. Therefore $p_i^m(k) = s_i^m(k)$, and the optimization in (22) can be simplified to the one presented in Algorithm 3.

VI. SIMULATION

In this section, we use MATLAB simulations of a wireless content distribution network to evaluate the performance of: 1) the proposed throughput-optimal algorithms; 2) a suboptimal decomposed scheme; and 3) a distributed greedy policy. The simulated CDN is an example of the one shown in Fig. 1 with the following specifications. There are |M|=3 caches, |N|=4 clusters, |U| = 12 and inelastic users. The capacity of each cache is v = 3, and D = 4 is the duration of a frame. Each user requires a delivery ratio of nu = 0.9 and has a request rate of λu content/frame. The popularity of inelastic requests is uniformly distributed among |I|= 12 different types of inelastic content. There are a total of |E| elastic contents, for each there is a binomial Bin(4,0.2) number of requests in each cluster (i.e., λ n,e=0.8). We further assume the packet erasure probability of each wireless channel is 25%. The mean delivery ratio (average over all users, denoted by ṅ*) and the mean elastic service rate (denoted by s *el) provided by Algorithm 2 are presented in Table II for different numbers

TABLE II
PERFORMANCE OF ALGORITHM 2

	|E|	4	6	8	10	12
$\lambda_u = 0.9$	$\bar{\eta}^*$	0.88	0.86	0.85	0.84	0.83
	$\bar{s}_{el}^*$	0.8	0.8	0.8	0.8	0.75
$\lambda_u = 1$	$\bar{\eta}^*$	0.87	0.85	0.83	0.81	0.80
	$\bar{s}_{el}^*$	0.8	0.8	0.8	0.79	0.74

TABLE III
PERFORMANCE OF ALGORITHM 4

	|E|	4	6	8	10	12
$\lambda_u = 0.9$	$\bar{\eta}/\bar{\eta}^*$	0.96	0.99	1	1.01	1.02
	$\bar{s}_{el}/\bar{s}_{el}^*$	1	1	1	1	1
$\lambda_u = 1$	$\bar{\eta}/\bar{\eta}^*$	0.94	0.96	0.99	1.01	1.02
	$\bar{s}_{el}/\bar{s}_{el}^*$	1	1	1	1	1

Algorithm 4: Decomposed Elastic-Inelastic Scheduling and Placement Scheme

Given the statistics of the requests and channel states, divide the available cache capacity to v to v E and v –v E

Elastic traffic:

Allocate vE of the caches' capacity to elastic contents and use Algorithms 1 for service scheduling and content placement of the elastic requests.

Inelastic traffic:

At the beginning of frame , given the deficit queue lengths, arrivals and the channel states, let . du=du(k)+∑ i au,I (k) Solve the following maximization problem to find the optimal inelastic schedule:

$$\begin{aligned} \max \quad & \sum_{u,i}\{d_u\}^+ s_{u,i} \\ \text{subject to} \quad & s_{u,i} \le a_{u,i} \qquad \forall u,i \\ & s_{u,i} \le \sum_m c_u^m s_i^m \qquad \forall u,i \\ & \sum_i s_i^m \le \min(D, v-v_E) \qquad \forall m \\ & s_i^m \in \{0,1\} \qquad \forall m,i. \end{aligned} \quad (28)$$

Of elastic contents (denoted by |E|). As expected, by increasing, |E|, the performance drops. We saw, in Section IV, that a throughput-optimal scheme must jointly decide on elastic and inelastic scheduling and dynamically allocate the cache spaces to these two types of traffic based on the channel states and new request arrivals. This will result in a fairly complex optimization problem as in Algorithm 2. In Algorithm 4, we propose a simple (suboptimal) scheme that divides the cache spaces for different types of content *a priori*. Following this static cache resource allocation, the scheduling of inelastic and elastic requests can be completely decomposed, and the (sub) optimal schedule can be found

TABLE IV
PERFORMANCE OF ALGORITHM

	|E|	4	6	8	10	12
$\lambda_u = 0.9$	$\bar{\eta}/\bar{\eta}^*$	0.9	0.91	0.89	0.88	0.87
	$\bar{s}_{el}/\bar{s}^*_{el}$	0.99	0.99	0.98	0.97	0.99
$\lambda_u = 1$	$\bar{\eta}/\bar{\eta}^*$	0.87	0.86	0.87	0.86	0.85
	$\bar{s}_{el}/\bar{s}^*_{el}$	0.99	0.98	0.97	0.96	0.98

TABLE V
COST EFFECTIVE

	Trade-off parameter	0	5	10	20	50
$\lambda_u = 0.9$	Avg delivery ratio	0.9	0.85	0.80	0.69	0.46
	Avg cost	69.5	54.5	48.9	37.5	17.7
$\lambda_u = 1$	Avg delivery ratio	0.9	0.84	0.79	0.68	0.46
	Avg cost	70.7	58.4	52.5	41.2	21.7

Algorithm 5: Decentralized Greedy Joint scheme

Each cache m places the content and schedules the service independently from the other caches by solving the following:

$$\max \quad \sum_{u,i} \{d_u\}^+ a_{u,i}(k) c_u^m s_i^m + \sum_{e,n} q_{n,e} c_n^m p_e^m s_{n,e}^m$$

$$\text{s.t.} \quad \sum_{e \in \mathbf{E}} p_e^m(k) + \sum_{i \in \mathbf{I}} s_i^m(k) \le v$$

$$\sum_i s_i^m(k) \le D$$

$$\sum_e s_{n,e}^m(k) \le D \qquad \forall n$$

$$s_i^m(k) \in \{0,1\} \qquad \forall m, i.$$

In Table III, we observe that the reduction of the performance in Algorithm 4 is at most 4%–6% compared to Algorithm 2. It is worth noting that when the elastic requests are not achievable due to wireless channel constraints (e.g., for|E|=10,12 in Table II), separating inelastic and elastic scheduling can actually be beneficial. As seen in Table III, there is up to 2% improvement in the provided inelastic service for these cases. Essentially, by devoting a fixed proportion of the cache capacities to inelastic content, we ensure that long elastic request queues will not cause the scheduler allocate excess cache space to elastic content. As mentioned earlier, Algorithm 2 can be very hard to implement in large networks. Therefore, we propose a distributed greedy scheme (Algorithm 5), whose performance is evaluated in Table
In this algorithm, each cache, independent of the others, loads and serves content. The simulation results suggest that although the greedy algorithm is not throughput-optimal, the performance loss is limited to at most ~ 15% compared to the throughput-optimal scheme.
Finally, we study the performance of Algorithm 3, which is aimed toward real-time streaming. The results are presented in Table V. Here, we use a trade off parameter Y that determines how much we value refresh cost versus throughput. As expected, the average provided delivery ratio decreases with the trade off parameter Y, while increasing Y causes the average Cost decrease and converge to its minimum value.

VII. CONCLUSION

In this paper, we studied algorithms for content placement and scheduling in wireless broadcast networks. While there has been significant work on content caching algorithms, there is much less on the interaction of caching and networks. Converting the caching and load balancing problem into one of queuing and scheduling is hence interesting. We considered a system in which both inelastic and elastic requests coexist. Our objective was to stabilize the system in terms of finite queue lengths for elastic traffic and zero average deficit value for the inelastic traffic. We showed how an algorithm that jointly performs scheduling and placement in such a way that Lyapunov drift is minimized is capable of stabilizing the system. In designing these schemes, we showed that knowledge of the arrival process is of limited value to taking content placement decisions. We incorporated the cost of loading caches in our problem with considering two different models. In the first model, cost corresponds to refreshing the caches with unit periodicity. In the second model relating to inelastic caching with expiry, we directly assumed a unit cost for replacing each content after expiration. A max-weight-type policy was suggested for this model, which can stabilize the deficit queues and achieves an average cost that is arbitrarily close to the minimum cost.

APPENDIX

A. Bounds on the constant term in the Lyapunov drift (Theorem 1)

For the ease of notation, let

$$\underline{s}_{n,e}(k) = \sum_m s^m_{n,e}(k) c^m_n(k) p^m_e(k).$$

Hence,

$$s_{n,e}(k) = \min(q_{n,e}, \underline{s}_{n,e}(k)), \text{ and}$$

$$s_{n,e}(k) \le \underline{s}_{n,e}(k) \le \sum_m s^m_{n,e}(k).$$

For the drift expression in the proof of theorem 1, it is straight forward to verify

$$\begin{aligned} B &= \frac{1}{2}\mathbb{E}\left[\sum_{n,e}(a_{n,e}(k+1) - s_{n,e}(k))^2\right] \\ &\quad + \mathbb{E}\left[\sum_{n,e} q_{n,e}\max(0, \underline{s}_{n,e}(k) - q_{n,e})\right] \\ &\overset{(a)}{\le} \frac{1}{2}\mathbb{E}\left[\sum_{n,e}(a_{n,e}(k+1))^2\right] + \frac{1}{2}\mathbb{E}\left[\sum_{n,e}(\underline{s}_{n,e}(k))^2\right] \\ &\quad + \mathbb{E}\left[\sum_{n,e}(\underline{s}_{n,e}(k))^2\right] \\ &\le \frac{1}{2}\sum_{n,e}\sigma^2_{n,e} + \frac{3}{2}\mathbb{E}\left[\sum_n(\sum_{e,m} s^m_{n,e}(k))^2\right] \\ &\overset{(b)}{\le} \frac{1}{2}\sum \sigma^2_{n,e} + \frac{3}{2}|\mathcal{N}||\mathcal{M}|^2 D^2 \end{aligned}$$

Where (a) follows from

$$\begin{aligned} &\mathbb{E}[\sum_{n,e} q_{n,e}\max(0, \underline{s}_{n,e}(k) - q_{n,e})] \\ &\le \mathbb{E}[\sum_{n,e}\max(q_{n,e}, \underline{s}_{n,e})\max(0, \underline{s}_{n,e}(k) - q_{n,e})] \\ &\le \mathbb{E}[\sum_{n,e}(\underline{s}_{n,e}(k))^2] \end{aligned}$$

And (b) holds because $\sum_e (s^m_{n,e})(k) < D$

B. Simplified Implementation of algorithm I

We show that $(p^m_{n,e})^*$, $(s^m_{n,e})^*$,values chosen by algorithm Are the optimal values for the following problem:

$$\begin{aligned} \max \quad & \sum_{n,e,m} q_{n,e} c^m_n(k) p^m_e(k) s^m_{n,e}(k) \\ & \sum_e p^m_e(k) \le v \qquad \forall m \\ & \sum_e s^m_{n,e}(k) \le D \qquad \forall m, n. \end{aligned} \tag{30}$$

Suppose that $p^m_{n,e}(k)=$, $p^m_{n,e}(k)^*$ (satisfying $\sum_e (p^m_{n,e})(k) < v$) Are given we can now separately solve (30) for each cache m and front endn to find $(s^m_{n,e})^*$

$$(s^m_{n,e})^* = \arg\max \sum_e q_{n,e} c^m_n(k)(p^m_e)^* s^m_{n,e}(k)$$

$$\sum_e s^m_{n,e}(k) \le D.$$

One can verify the optima l value of the objective function is $Dc^m_n(k)$ max f € E $(p^m_f)^* q$ n,f). Which can be achieved by?

$$(s^m_{n,e})^* = \begin{cases} D, & \text{if } e = rand\left(\arg\max_{f\in\mathbf{E}}\left((p^m_f)^* c^m_n q_{n,f}\right)\right) \\ 0, & \text{otherwise.} \end{cases}$$

Next, we observe the optimal $(p^m_f)^*$ values must be chosen such that

$$D\sum_{m,n} c^m_n(k) \max_{f\in\mathbf{E}}\left(p^m_f q_{n,f}\right)$$

Is maximized .It is straight forward to see such $(p^m_f)^*$ values can also be derived from solving the problem

C.Throughput optimality of Algorithm 2

Consider the joint Lyapunov function (k) =VE(k)+VI(k) where

$$V_I(k) = \frac{1}{2}\sum_u(\{d_u(k^+)\}^+)^2$$

And vE (k) is defined as in the proof of theorem 1.the expected drift Δ V(k) =E[v(K+1)-V(k)|qn,e,du(k+)=du] can be written

$$\Delta V(k) = \Delta V_E(k) + \Delta V_I(k)$$

And we have already shown (proof of theorem 1) that

$$\Delta V_E(k) = \sum_{n,e} q_{n,e}\lambda_{n,e} - \mathbb{E}\left[\sum_{n,e,m} q_{n,e} c_n^m(k) p_e^m(k) s_{n,e}^m(k)\right] + B$$

The expected in elastic drift can also be bounded as follows:

$$\begin{aligned}\Delta V_I(k) &= \frac{1}{2}\mathbb{E}\left[\sum_u (\{d_u + \sum_i \tilde{a}_{u,i}(k+1) - s_u(k)\}^+)^2\right] \\ &\quad - \frac{1}{2}\sum_u (\{d_u\}^+)^2 \\ &\overset{(a)}{\leq} \frac{1}{2}\mathbb{E}\left[\sum_u (\{d_u\}^+ + \sum_i \tilde{a}_{u,i}(k+1) - s_u(k))^2\right] \\ &\quad - \frac{1}{2}\sum_u (\{d_u\}^+)^2 \\ &= \sum_u \{d_u\}^+ \eta_u \lambda_u - \mathbb{E}\left[\sum_u \{d_u\}^+ s_u(k)\right] + B''\end{aligned}$$

In which (a) follows since ({X+Y)+)2<({x}+y)2,and B'' has a finite value

$$\begin{aligned}B'' &= \frac{1}{2}\mathbb{E}\left[\sum_u \left(\sum_i \tilde{a}_{u,i}(k+1) - s_u(k)\right)^2\right] \\ &\leq \frac{1}{2}\sum_u 1 = \frac{|\mathcal{U}|}{2}. \qquad (31)\end{aligned}$$

Consequently, the following bound holds for the joint drift:

$$\begin{aligned}\Delta V(k) &\leq \sum_u \{d_u\}^+ \eta_u \lambda_u - \mathbb{E}\left[\sum_u \{d_u\}^+ s_u(k)\right] \\ &\quad + \sum_{n,e} q_{n,e}\lambda_{n,e} - \mathbb{E}\left[\sum_{n,e,m} q_{n,e} c_n^m(k) p_e^m(k) s_{n,e}^m(k)\right] \\ &\quad + B'' + B. \qquad (32)\end{aligned}$$

At the beginning of each frame k , the scheme in Algorithm 2 chooses the schedule S*(k) by solving (17).

$$\begin{aligned}&\sum_u \{d_u\}^+ s_u^*(k) + \sum_{e,n,m} q_{n,e} c_n^m p_e^m (s_{n,e}^m)^* \\ &\geq \mathbb{E}_{\mathbf{P}^*}\left[\sum_u \{d_u\}^+ s_u(k) + \sum_{e,n,m} q_{n,e} c_n^m p_e^m s_{n,e}^m\right]\end{aligned}$$

where the expectation is taken over all legitimate schedules S(k) with respect to the distribution used by the policy P* .Taking expectation from both sides of the above inequality over the arrival and channel state processes and using (13) will result in (for a small enough $\epsilon > 0$)

$$\begin{aligned}&\mathbb{E}\left[\sum_u \{d_u\}^+ s_u^*(k)\right] + \mathbb{E}\left[\sum_{e,n,m} q_{n,e} c_n^m p_e^m (s_{n,e}^m)^*\right] \\ &\geq \sum_u \{d_u\}^+ (\lambda_u \eta_u + \epsilon) + \sum_{e,n} q_{n,e}(\lambda_{n,e} + \epsilon).\end{aligned}$$

Considering the above inequality in(32) concludes

$$\Delta V(k) \leq -\epsilon(\sum_u \{d_u\}^+ + \sum_{n,e} q_{n,e}) + B + B''. \qquad (33)$$

Thus, the expected drift is negative for large enough queue lengths, and hence the queues will be stable using the scheme in Algorithm 2.

D. Simplified Implementation of Algorithm 2

Note that in the current model, we assumed that the caches can refresh their content at the beginning of each frame. Hence, we only need to cache a chunk of content at frame if it is scheduled to be served. Therefore, we let $p_i^m(k) = s_i^m(k)$ and simplify the maximization in Algorithm 2 as follows:

E. Proof of corollary 1

We have shown, using the Lyapunov analysis, that the Markov chain of the deficit and request queues is positive recurrent (stable).

$$\begin{aligned}\mathbb{E}[\Delta V(k)] &= \mathbb{E}[V(k+1)] - \mathbb{E}[V(k)] \\ &\leq B + B'' - \epsilon\left(\sum_u \mathbb{E}\left[\{d_u(k^+)\}^+\right] + \sum_{n,e} \mathbb{E}\left[q_{n,e}(k^+)\right]\right)\end{aligned}$$

Note that E[V(k+1)]-E[v(k)] holds at the steady state ,and hence

$$\sum_{u} \mathbb{E}\left[d_u\right] + \sum_{n,e} \mathbb{E}\left[q_{n,e}\right] \leq \frac{B + B''}{\epsilon}. \qquad (34)$$

From (31) and (29) we know

$$B + B'' \leq \frac{1}{2}\left(\sum_{n,e} \sigma_{n,e}^2 + 3|\mathcal{N}||\mathcal{M}|^2 D^2 + |\mathcal{U}|\right)$$

And the proof will follow by considering the above bound in (34)

ACKNOWLEDGEMENT

In this paper we have algorithms to maintain the stability of the system load within the capacity region. We have Minimizes the average expected cost while stabilizing the deficit queues. In this we have to refresh the caches.

Here the content is transferred through wireless base station at front end terminals .Finally the content is transferred to all the clusters that are elastic and inelastic traffic type.

REFERENCES

[1] N. Abedini and S. Shakkottai, "Content caching and scheduling in wireless broadcast networks with elastic and inelastic traffic," in Proc. IEEE WiOpt, 2011, pp. 125–132.
[2] M. M. Amble, P. Parag, S. Shakkottai, and L. Ying, "Content-aware caching and traffic management in content distribution networks," in Proc. IEEE INFOCOM, Shanghai, China, Apr. 2011, pp. 2858–2866.
[3] A. Stolyar, "Maximizing queueing network utility subject to stability: Greedy primal-dual algorithm," Queueing Syst. Theory Appl., vol. 50, no. 4, pp. 401–457, 2005.
[4] S. Borst, V. Gupta, and A. Walid, "Distributed caching algorithms for content distribution networks," in Proc. IEEE INFOCOM, San Diego, CA, USA, Mar. 2010, pp. 1–9.
[5] P. Cao and S. Irani, "Cost-awareWWWproxy caching algorithms," in Proc. USENIX Symp. Internet Technol. Syst., Berkeley, CA, Dec. 1997, p. 18.
[6] K. Psounis and B. Prabhakar, "Efficient randomized Web-cache replacement schemes using samples from past eviction times," IEEE/ACM Trans. Netw., vol. 10, no. 4, pp. 441–455, Aug. 2002.
[7] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," IEEE Trans. Autom. Control, vol. 37, no. 12, pp. 1936–1948, Dec. 1992.
[8] X. Lin and N. Shroff, "Joint rate control and scheduling in multihop wireless networks," in Proc. 43rd IEEE CDC, Paradise Islands, Bahamas, Dec. 2004, vol. 2, pp. 1484–1489.
[9] J. Jaramillo and R. Srikant, "Optimal scheduling for fair resource allocation in ad hoc networks with elastic and inelastic traffic," in Proc. IEEE INFOCOM, San Diego, CA, USA, Mar. 2010, pp. 1–9.

Selected Paper from International Conference on Computing (NECICC-2k15)