www.ijecs.in
*International Journal Of Engineering And Computer Science ISSN: 2319-7242*
*Volume 5 Issue 5 May 2016, Page No. 16679-16679*

# Content Caching and Scheduling in Wireless Networks with Elastic and Inelastic Traffic

*B.Muniswamy, Dr. N. Geethanjali*

Research scholar
Dept Of CSE,
Sri Krishnadevaraya University
Anantapuramu, AP. India
Associate Professor,

Dept Of CSE,
Sri Krishnadevaraya University
Anantapuramu, AP. India

*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 (implying no hard delay constraint) or inelastic (requiring that a delay target be met). Correspondingly, we have request queues that indicate the number of elastic requests, and deficit queues that indicate the deficit in 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-service (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), queueing.

## I. INTRODUCTION

THE PAST few years have seen the rise of smart handheld wireless devices as ameans 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 are at 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 be at the wireless gateway, which could be a cellular base station through which users obtain network access. Furthermore, it is natural to try to take advantage of the inherent broadcast nature of the wireless medium to satisfy multiple users simultaneously.

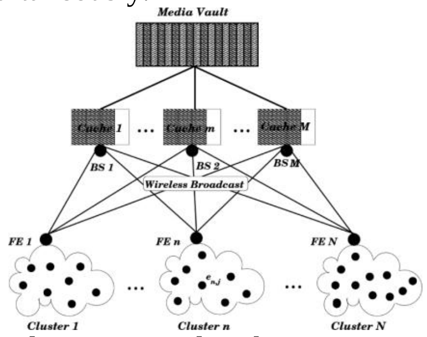

Fig. 1. Wireless content distribution. A media vault is used to place content in caches at wireless BSs, which can broadcast content. Users are grouped into clusters, each of whose requirements are aggregated at FEs.

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. The requests made by each cluster are aggregated at a logical entity that we call a *front end* (FE) associated with that cluster. The front end 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 network between the caches to the users has finite capacity;2) each cache can only host a finite amount of content;and 3) 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, and 2) inelastic requests that have a hard delay constraint. Elastic requests are stored in a *request queue* at each front end, with each type of request occupying a particular queue. Here, the objective is to stabilize the queue, so as to have finite delays. For inelastic requests, we adopt the model proposed in [2] wherein users request chunks of content that have a strict deadline, and the request is dropped if the deadline 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 playout.” 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. In doing so, we will also determine the value of predicting the demand for different types of content and what impact it has on the design of caching algorithms.

### *A. Related Work*

The problem of caching, and content scheduling has earlier been studied for onlineWeb 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 are [3]–[5]. Load balancing and placement with linear communication costs is examined in [6] and [7]. 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, or wireless aspects. The techniques that we will employ are based on the literature on scheduling schemes. Tassiulas *et al.* proposed the MaxWeight scheduling algorithm for switches and wireless networks in their seminal work [8]. 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. Various extensions of this work that followed since are [9]–[12]. These papers explore the delays in the system for single downlink with variable connectivity, multirate links, and multihop wireless flows. However, these do not consider content distribution with its attendant question of content placement. Closest to our work is [13], 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 distribution scenario. The first is the case of file distribution (elastic) along with streaming of stored content (inelastic), where we model cost in terms of the frequency with which caches are refreshed. The second is the case of streaming of content that is generated in real-time, where content expires after a certain time, and the cost of placement of each packet in the cache is considered.

• We first characterize 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 eachcontent 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 [14], 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

Consider the content distribution network depicted in Fig. 1. There is a set of base stations $\mathcal{M}$, and each base station is associated with a cache. The caches are all connected to a media vault that co$\mathcal{N}$ntains all the content. The users in the system are divided into clusters based on their geographical positions, and we let denote the set of these clusters. Also, as discussed in the Introduction, there are front ends in each cluster, also denoted **n€N** 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. Requestsmade by inelastic usersmust 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.

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 (e.g., following a Zipfslaw that captures the varied popularity of different types). 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 **μ** is $\mathbf{n_u y_u}$, in which $\mathbf{n_u}$ the minimum acceptable delivery ratio is. 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. 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. This measure would imply that any particular elastic request does not have to wait arbitrarily long.

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, wemay occasionally choose to reload a cache partially and not utilize the whole available capacity. For this variation, we will only consider inelastic traffic, which is consistent with the idea of real-time streaming of live events.

Table I summarizes the notations used in this paper. We will first study a pure elastic system in Section III,where the requests are served through wireless unicast channels between base stations and front ends. We will address the joint elastic-inelastic system in Section IV.

## 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

DOI: 10.18535/ijecs/v5i5.56

TABLE I
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 $k$ |
| $\lambda_u$ | Inelastic request rate by user $u$ |
| $\eta_u$ | 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 $k$ |
| $c_u^m(k)$ | 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 $k$ |

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 $e \in \mathbf{E}$ 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 $m$ to satisfy

$$\sum_{e\in\mathbf{E}} p_e^m(k) \le v, \qquad \text{for each frame } k. \qquad (2)$$

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**

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 In elastic 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.

**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 $(p_e^m)^*$:**

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

**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:**

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

**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 $(p_f^m)^* c_n^m q_{n,f}$.**

The expected drift can be written as

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

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

p

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

Note that each cache can broadcast at most D contents during a frame, hence we require

$$\sum_i s_i^m(k) \le D \qquad \forall m, k.$$

The actual inelastic service, provided to user for content , depends not only on the channel states $c^m_u(k)$ and the cache presence $P^m_i(K)$, but also on whether there is a new (not expired)request for content (*i.e.*$a_{ui}$(k)=1). It should be straightforward to verify that the total actual inelastic service provided to user during frame is

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

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

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

## 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 onlyconsider 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.

```
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
  $d_u = d_u(k) + \sum_i \bar{a}_{u,i}(k)$.
  Solve the following maximization problem to find the
  optimal placement and schedule $(p_i^m(k))^* = (s_i^m(k))^*$:
```

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

## 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.

TABLE II
PERFORMANCE OF ALGORITHM 2

| | $\lvert\mathbf{E}\rvert$ | 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

| | $\lvert\mathbf{E}\rvert$ | 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 $v$ to $v_E$ and $v - v_E$:
Elastic traffic:
Allocate $v_E$ 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 $k$, given the deficit queue lengths,
arrivals and the channel states, let $d_u = d_u(k) + \sum_i \bar{a}_{u,i}(k)$.
Solve the following maximization problem to find the
optimal inelastic schedule:
```

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

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 contenta *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 with less complexity.Note that, in general, finding the best fixed allocation of the cache capacities to elastic and inelastic traffic (i.e., and in Algorithm 4) is not straightforward and may be found using heuristic methods or by simulation. However, for the symmetric scenarios, in which the distributions of the requests are similar, it is straight forward to verify is sufficient for $V_E$ serving elastic requests.

TABLE IV
PERFORMANCE OF ALGORITHM 5

| | $\lvert\mathbf{E}\rvert$ | 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 |

## 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 queueing 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.

## 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] I. Hou, V. Borkar, and P. Kumar, “A theory of QoS for wireless,” in *Proc. IEEE INFOCOM*, Rio de Janeiro, Brazil, Apr. 2009, pp. 486–494.

[3] R. M. P. Raghavan, *Randomized Algorithms*. NewYork,NY,USA: Cambridge Univ. Press, 1995.

[4] P. Cao and S. Irani, “Cost-awareWWWproxy caching algorithms,” in *Proc. USENIX Symp. Internet Technol. Syst.*, Berkeley, CA, Dec. 1997,p. 18.
[5] 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.
[6] N. Laoutaris, O.T. Orestis, V.Zissimopoulos, and I. Stavrakakis, “Distributed selfish replication,” *IEEE Trans. Parallel Distrib. Syst.*, vol.17, no. 12, pp. 1401–1413, Dec. 2006.
[7] 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.