Cooperative Caching for Shared Spectrum Networks

Dibakar Das
Electrical, Computer and Systems Engineering
Rensselaer Polytechnic Institute
Troy, NY 12180
Email: dasd2@rpi.edu

Alhussein A. Abouzeid
Electrical, Computer and Systems Engineering
Rensselaer Polytechnic Institute
Troy, NY 12180
Email: abouzeid@ecse.rpi.edu

***Abstract*—This paper considers cooperation between primary and secondary users in shared spectrum radio networks via caching. A network consisting of a single macro (primary) base-station and multiple small (secondary) base-stations is considered. Secondary base-stations can cache some primary files and thereby satisfy content requests generated from nearby primary users. For this cooperative scenario, we develop two caching and scheduling policies under which the set of primary and secondary user request generation rates that can be supported increases from the case without cooperation. The first of these algorithms provides maximum gain in the set of supportable primary and secondary request generation rates. However under this algorithm primary packet transmissions from secondary base-stations do not have higher priority than that of secondary packets. As a result, we propose another sub-optimal (with respect to set of supportable request generation rate vectors) algorithm wherein primary packet transmissions from secondary base-stations have higher priority than that of secondary packets. Extensive simulations are conducted to compare the performance of both algorithms with that of a non-cooperative algorithm that is optimal, with respect to set of supportable request generation rates, among all non-cooperative policies.**

I. Introduction

Shared spectrum or cognitive networks are promising solutions to the spectrum scarcity problem in future networks. In cognitive networks some unlicensed or secondary users are allowed to opportunistically access and transmit on a given channel provided the primary or licensed users of that channel are not active. Traditionally, primary and secondary networks have been assumed to be non-cooperative i.e., the primary and secondary users do not assist in each other's transmissions. However, if secondary users choose to assist the transmission of primary users, then it may reduce the overall duration for which the primary user is active on that channel. This in turn can also benefit the secondary users because it increases their own transmission opportunities. Cooperation between primary and secondary networks have been widely studied from a physical-layer perspective (e.g. [1]–[3]). Other recent works (e.g. [4]–[8]) study network-layer aspects of cooperation such as queuing and prioritized scheduling.

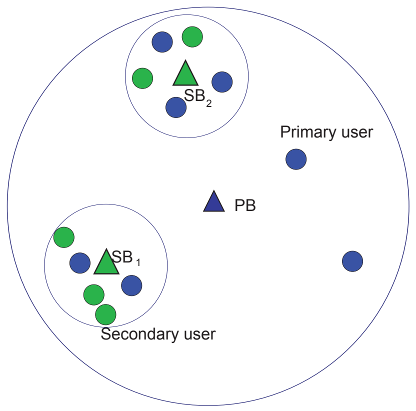


Fig. 1. A primary network with one base-station co-existing with two secondary base-stations. The outermost circle indicates the transmission region of the primary base-station.

Another popular solution to support increasing mobile network traffic is to encourage localized communication by using small base-stations. This leads to a higher spatial re-use of the spectrum. However, limited capacity of backhaul links at the base-stations diminish the impact of this approach [9]. As a result, it was suggested in [9] to cache popular files at nearby base-stations which leads to lower back-haul usage as well as improves delay performance for the users. Caching is also an attractive approach since storage-capacity is relatively inexpensive compared to other network resources.

In this work we explore cooperation between primary and secondary networks via caching. We consider a primary network consisting of a single base-station that serves primary users. Co-existing with the primary network is a set of small secondary base-stations that serve secondary users. An example of such a network is shown in Fig.1. Both primary and secondary users request content from their respective base-stations where these requests are queued at some *request-queue*. The base-stations serve these requests by transmitting the packets corresponding to requested files if the file is present in their cache. If the requested file is not currently present in the cache, then the file is fetched (after some delay) so as to satisfy the content-request. We assume that content is fetched at base-stations periodically and we refer to this period as the cache-refresh period. Since some of the secondary base-stations might be located closer to the primary users, they may have better downlink channels than the primary base stations. Therefore, if some of the content requests from primary users are served via secondary base-stations then that might free up spectrum resources for use by the secondary users.

For the above network scenario we address the important problem of developing caching and scheduling policies with performance guarantees. In particular we design algorithms under which a centralized network controller determines: (a) which files to cache at the secondary base-stations in every cache-refresh period and, (b) how to schedule transmissions in each time-slot. The goal of the network controller is to maintain stability of the request-queues i.e., to keep the length of all request-queues in the network bounded. Accordingly, our performance measure is the set of all primary and secondary user request generation rates for which every request-queue in the network is stable.

We develop two algorithms: Dynamic Caching and Scheduling Policy (DCSP) and Minimum Caching and Scheduling Policy (MCSP). Both algorithms are developed using Lyapunov-drift techniques and make decisions without knowledge of the request generation rates of secondary users. We find that the set of primary and secondary user request generation rate vectors for which the network is stable under both algorithms is greater than that under any non-cooperative algorithm. Under the DCSP algorithm each secondary base-station caches only one secondary file in every period while filling up the rest of the cache with primary files. However, only primary packet transmissions from the primary base-station enjoy higher priority of channel access. In the MCSP algorithm however, requests from primary users queued at any secondary base-station are always served with higher priority than that from secondary users. Furthermore, in every caching period, the network-controller dynamically varies the number of cached primary files while ensuring system stability. As a result secondary base-stations can serve more than one type of secondary file requests in each period. Simulation results in Section V show that this algorithm, with proper selection of a penalty parameter, tends to have better delay performance than the DCSP algorithm when request generation rates are low. In such scenarios it is not necessary for system stability to cache as much primary files as possible in every secondary base-station. In Section V we also show that the delay performance of MCSP is not guaranteed to be better than DCSP for any choice of the penalty parameter. However, the guaranteed stability region of the network under MCSP is less than that under the DCSP algorithm.

Related work: There is a substantial body of literature on caching in non-cognitive wireless networks. Some results include [9]–[11] and the references therein. On the other hand caching in cognitive networks has been studied recently in [12] and [13]. However they did not consider primary-secondary cooperation. Our periodic cache-refresh policy and the use of Lyapunov drift to develop a scheduling policy is motivated by [14] and [15]. However, their results are not directly applicable in the cognitive network setting since here primary users have higher priority of channel access than secondary nodes. While a simple Lyapunov drift-based algorithm tries to serve the queues with higher backlogs first, in our model the request-queues in the primary base-station would have to be served before that at the secondary base-stations even when it has relatively lower backlog (except possibly when a secondary base-station is serving a primary request). Such complications resulting from higher priority of service for primary users are addressed in this work, and, up to our knowledge, have not been addressed before.

In Section II we describe our system model. In Section III and IV we propose the DCSP and MCSP policies, respectively. Section V presents simulation results. Section VI concludes the paper. Detailed description of the policies, proof of their stability properties are relegated, due to lack of space, to the technical report [16].

II. System model

The network consists of a macrocell wherein a single primary base-station PB serves $N^{(p)}$ primary users: PU$_1$,..., PU$_{N^{(p)}}$. It also contains M secondary small-cells SC$_1, \ldots,$ SC$_M$ with secondary base-stations SB$_1$,...,SB$_M$ at their centres respectively. There are $N^{(s)}$ secondary users which are denoted as SU$_1$,...,SU$_{N^{(s)}}$. Each secondary user is located in exactly one of those small-cells. We consider a discrete time-slotted model. Every file request is served by successfully transmitting C packets of equal size. A base-station attempts a new packet transmission only at the beginning of a time-slot; it can attempt at most one such transmission at any slot. At the end of the time-slot the transmission is either successful and the packet is successfully received by the desired user or the transmission fails and the packet needs to be re-transmitted at some other slot. Next we present details about our transmission scheme, interference model, caching and scheduling constraints.

A. Transmission Model

The primary and secondary base-stations transmit at fixed power. All secondary base-stations have identical transmission range which is lower than that of PB. Some primary users are located relatively far away from PB but close to one or more secondary base-stations (i.e., within a small cell). As a result, the probability of a successful transmission from PB to such primary users is lower than that from adjacent secondary base-stations. For simplicity, we assume: (a) in every slot, transmission from PB to *any* user succeeds with probability p (where $0 < p \leq 1$) while transmissions from any secondary base-station to any user within its transmission range succeeds with probability 1, and (b) any primary user is within transmission range of at most one secondary base-station. We denote the set of primary and secondary users in SC$_i$ (where $1 \leq i \leq M$) by $\phi_i^{(p)}$ and $\phi_i^{(s)}$ respectively. All secondary users and base-stations are within the transmission range of PB.

B. Caching Model

Every time a secondary base-station caches a set of files it incurs some cost. This cost is modeled by requiring that secondary base-stations only cache files periodically. A higher frequency of caching reflects a higher cost. A cache-refresh period is defined to be the number of slots between two successive caching events. A cache refresh period may also represent how frequently contents in files become outdated thereby requiring their newer versions to be fetched. It consists of T slots with the very first caching event being at slot $t = 1$.

C. Generation of Content Requests

We consider the case where primary and secondary networks cater to different types of users. This can occur, for example, if the small-cells serve an industrial or academic environment while users of the macrocell are the general public who are typically interested in video content. We denote the library of files requested by primary users as $F^{(p)}$ with individual files in the set being referred to as F$_1^{(p)}$,...,F$_{|F^{(p)}|}^{(p)}$. Similarly we denote the library of files requested by secondary users as $F^{(s)}$ with individual files in the set being referred to as F$_1^{(s)}$,...,F$_{|F^{(s)}|}^{(s)}$. In this work we assume these two sets are mutually exclusive. Henceforth we will call files in $F^{(p)}$ and $F^{(s)}$ as primary and secondary files respectively. Similarly, we call packets corresponding to primary and secondary files as primary and secondary packets respectively. All files are of equal size.

Files are requested by users according to a popularity distribution which varies from period to period in an identical and independently distributed (iid) fashion reflecting the change in user's preference. We model this by assuming that in each period the network is in one of the "popularity states" $x \in \tilde{x}$ with probability q_x, where $\tilde{x}$ denotes a finite collection of such states. Given a primary (respectively secondary) user requests a file when the network is in state x, the file F$_i^{(p)}$ (resp. F$_i^{(s)}$) is requested with probability $P_{i,x}^{(p)}$ (resp. $P_{i,x}^{(s)}$).

In every slot primary (resp. secondary) user PU$_i$ (resp. SU$_j$) requests some file with probability $\lambda_i^{p)}$ (resp. $\lambda_j^{s)}$). We denote primary and secondary request generation-rate vectors:$(\lambda_1^{(p)}, \ldots, \lambda_{N^{(p)}}^{(p)})^T$ and $(\lambda_1^{(s)}, \ldots, \lambda_{N^{(s)}}^{(s)})^T$ as $\boldsymbol{\lambda}^{(p)}$ and $\boldsymbol{\lambda}^{(s)}$ respectively. All request generation process are assumed to be iid from slot to slot.

D. Serving Requests From Primary Users

File request from a primary user is queued at either a secondary base-station (if the primary user is located in a small-cell and the associated secondary base-station contains the file) or at the primary base-station. Every base-station maintains request-queues corresponding to each type of file requests. Items in each request-queue correspond to packet requests that need to be satisfied in response to a file request. For every file request at a base-station, C such packet requests are created and queued at the appropriate request-queue. These packet requests are satisfied in a first-come first serve (FCFS) manner from respective request-queues.

In order to focus only on the effect of cooperative caching by the secondary network, we assume the primary base-station can cache all the $|F^{(p)}|$ primary files. On other hand, the size of cache at each secondary base-station is finite. Each such cache can store at most B files where $0 \leq B \leq \min(|F^{(p)}|, |F^{(s)}|)$.

Furthermore, we assume that in each caching period a secondary base-station caches at least one secondary file i.e., the maximum number of primary files that can be cached at each base-station in any period is B-1. This is useful to construct efficient caching and scheduling algorithms with stability performance guarantees. In particular, it allows us to determine the stability constraints at secondary base-stations without accounting for probability of the event: "the network controller offers transmission rate to a secondary base-station which has no cached secondary file." Obtaining probability of such events is cumbersome but is required to analyze stability performance of algorithms in which secondary base-stations do not always have at least one secondary file in their cache.

We denote a request generated from PU$_i$ at slot t for file $f \in F^{(p)}$ by variable $A_{f,i}^{(p)}(t)$. This variable equals C if such a request is indeed generated at t; otherwise it equals 0. We denote the length of the request-queue of primary file f in SB_k and PB at slot t as $U_{f,k}^{(p)}(t)$ and $U_{f,0}^{(p)}(t)$ respectively. On successful transmission of a packet corresponding to file f from PB at t, $U_{\mathrm{f},0}^{(p)}(t)$ is decremented by 1. We denote the transmission rate offered to base-station PB and SB_k for packets of file f at time slot t by the binary variables $\mu_{f,0}(t)$ and $\mu_{f,k}(t) \in \{0,1\}$ respectively. We assume, no transmission-rate is offered to PB for transmitting packets corresponding to an empty request-queue and all queues are initially empty. The length of request-queue of every primary file f at PB is updated as:

$$\begin{aligned} U_{\mathrm{f},0}^{(p)}(t+1) &= U_{\mathrm{f},0}^{(p)}(t) - \mu_{f,0}(t) I_{\mathrm{PB}}(t) \\ &+ \sum_{\substack{i:\mathrm{PU}_i \notin \cup_j \phi_j^{(p)} \text{ or, } \mathrm{PU}_i \in \phi_k^{(p)} \\ \text{and } f \text{ is not present in } \mathrm{SB}_k\text{'s cache}}} A_{f,i}^{(p)}(t) \end{aligned}$$

where $I_{\mathrm{PB}}(t)$ is an indicator variable representing a successful transmission from PB to the receiving primary user at slot t; it equals 1 if the transmission is successful and is zero otherwise.

The length of request-queues of every primary file f at SB_k is updated as follows:

$$\begin{aligned} U_{f,k}^{(p)}(t+1) &= \max\{U_{f,k}^{(p)}(t) - \mu_{f,k}(t), 0\} \\ &+ \sum_{i:\mathrm{PU}_i \in \phi_k^{(p)}} A_{f,i}^{(p)}(t) \quad \forall 1 \le k \le M \end{aligned}$$

E. Serving Requests From Secondary Users

Requests from secondary users are submitted to the unique secondary base-station associated with each such user. Similar to the case of primary files, each secondary base-station maintains a request-queue for every secondary file. We denote a packet request generated from SU_j at slot t for a secondary file $f \in F^{(s)}$ by a variable $A_{f,j}^{(s)}(t) \in \{0, C\}$. We denote the length of request-queue in SB_k at t of file f as $U_{f,k}^{(s)}(t)$which is updated as,

$$\begin{aligned} U_{f,k}^{(s)}(t+1) &= \max\{U_{f,k}^{(s)}(t) - \mu_{f,k}(t), 0\} \\ &+ \sum_{j:\mathrm{SU}_j \in \phi_k^{(s)}} A_{f,j}^{(s)}(t) \quad \forall 1 \le k \le M \end{aligned}$$

We refer to request-queues corresponding to primary and secondary files as primary and secondary request-queues respectively.

Throughout the paper we use the notion of strong stability of request-queues, defined as follows.

Definition 1: A discrete time queue $U(t)$ is strongly stable if $\limsup_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}[|U(\tau)|] < \infty$. The network is strongly stable if every queue in the network is strongly stable.

F. Interference Model

We model co-channel interference among secondary base-stations by using a modified version of the protocol model of interference described in [17]. We call two secondary base-stations "interfering neighbors" if they lie within twice the transmission range of each other. A transmission from a secondary base-station to any user is feasible only if none of its interfering neighbors is transmitting at the same slot.

Note that under our system model assumptions, each secondary user is served by only one secondary base-station. Furthermore, we have ignored the possibility that two interfering secondary base-stations can transmit to users within their respective transmission range if the users are sufficiently far apart. Both assumptions simplify the construction of efficient caching and scheduling algorithms by allowing us to consider only interference among secondary base-stations rather than that among the downlink of every secondary user.

We denote as Λ the capacity region corresponding to the system model. This region contains all primary and secondary request generation rate vectors for which the network is stable under any algorithm. Exact description of the region is provided in [16].

III. DCSP Algorithm

In this section we give an overview of the DCSP algorithm that stabilizes the network of request-queues for all primary and secondary request generation rate vectors in the interior of the capacity region Λ.

Under DCSP, at the beginning of each period every secondary base-station caches $B-1$ most popular primary files (averaged over all popularity states) and the secondary file with highest instantaneous request-queue length at that base-station. Whenever the primary base-station is not transmitting, the controller offers transmission rates to secondary base-stations by solving a max-weight problem, over the set of all activation-vectors, similar to the backpressure type policy. The weights correspond to the sum of request-queue lengths of the cached primary and secondary files at each secondary base-station.

Intuitively, caching $B-1$ most popular primary files maximizes the rate of primary user requests that is served by secondary base-stations which in turn creates more transmission opportunities for secondary packets.

It can be shown that DCSP is throughput-optimal.

Theorem 1: DCSP stabilizes the network of request-queues for all ($\boldsymbol{\lambda}^{(p)}$,$\boldsymbol{\lambda}^{(s)}$)$\in$ Interior(Λ).

The vector pair ($\boldsymbol{\lambda}^{(p)}$,$\boldsymbol{\lambda}^{(s)}$) is in the interior of the region Λ if there exists some constant $\epsilon > 0$ s.t. $(\boldsymbol{\lambda}^{(p)} + \boldsymbol{\epsilon}^{(p)}, \boldsymbol{\lambda}^{(s)} + \boldsymbol{\epsilon}^{(s)}) \in \Lambda$. Here, $\boldsymbol{\epsilon}^{(p)}$, $\boldsymbol{\epsilon}^{(s)}$ are vectors of lengths $N^{(p)}$ and $N^{(s)}$ respectively and whose each component is ϵ.

Theorem 1 is valid even for the case when the sets $F^{(p)}$ and $F^{(s)}$ are not mutually exclusive and has arbitrary number of common files. However, in this case Λ is not the capacity region. Detailed description of DCSP and the proof of Theorem 1 can be found in technical report [16].

IV. MCSP Algorithm

In this section we briefly outline the MCSP algorithm in which only a fixed set of non-interfering secondary base-stations, denoted without loss of generality as $\mathrm{SB}_1, \ldots, \mathrm{SB}_G$ respectively (where $G \leq M$), cooperatively queue and serve primary packet requests.

In the MCSP algorithm we attempt to minimize the average number of primary files cached by secondary base-stations subject to the constraint that all request-queues are stable and primary packet requests are satisfied ahead of secondary ones in each secondary base-station. The algorithm is constructed using renewal frame based optimization methods. In these methods, the timeline is partitioned into contiguous collection of slots with each collection being referred to as a frame. Each frame is defined in terms of a "system state"; a new frame begins whenever the system-state is refreshed. In our case we define the system state to be the total length of all primary request-queues in the network similar to the way primary users' channel occupancy process was used as system-state in [8]. Note, in that work the authors did not study cooperative caching. Each frame begins when the sum of all primary request-queues transition from zero to a non-zero value.

Under the MCSP algorithm at the beginning of the r'th period (where $r = 1, 2, \ldots$) for every k (where $1 \leq k \leq G$) we first estimate the number of primary files ($\hat{n}_k^*(r)$) that would be cached at SB_k by a renewal-frame based method *had* the beginning of the period coincided with the beginning of a frame. $\hat{n}_k^*(r)$ is obtained by maximizing a standard Lyapunov drift plus penalty expression over a frame that consists of two terms: one decreasing with number of cached primary files and the other increasing with number of cached primary files and length of secondary request-queues. A constant penalty parameter V is associated with the former term; higher V means lesser number of cached primary files and vice-versa. The actual number of cached primary files ($n_k^*(r)$) at SB_k can be higher than $\hat{n}_k^*(r)$. This is because, at the beginning of a period some primary request-queues in a secondary base-station might be non-empty and we need to ensure that all primary files corresponding to those queues are cached at that base-station. More details about MCSP can be found in [16].

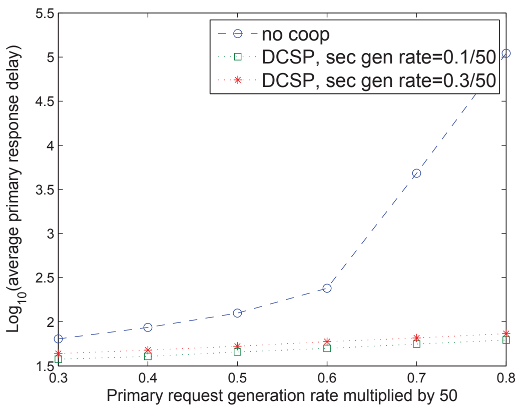


Fig. 2. Plot of base-10 logarithms of time-averaged primary response delay versus $50\lambda^{(p)}$, under no cooperation and the DCSP algorithm when $\lambda^{(s)}$ is $\frac{0.1}{50}$, $\frac{0.3}{50}$.

Guaranteed Stability Region for MCSP

Let $\Lambda_0^{(p)}$ denote the set of primary request generation rate vectors which can be satisfied even in absence of cooperation. Consider the region Λ^{MCSP} defined as the set $\{(\boldsymbol{\lambda}^{(p)}, \boldsymbol{\lambda}^{(s)}) : \boldsymbol{\lambda}^{(p)} \in \Lambda_0^{(p)}, \boldsymbol{\lambda}^{(s)} \in \text{Interior}(\Lambda^{(s)}(\boldsymbol{\lambda}^{(p)}))\}$. For a given $\boldsymbol{\lambda}^{(p)}$ the set $\Lambda^{(s)}(\boldsymbol{\lambda}^{(p)})$ defines the set of all secondary request generation rates that can be supported if $B-1$ most popular primary files are cached at each cooperative secondary base-station in each period and if only cooperative secondary base-stations can simultaneously transmit secondary packets when some cooperative base-station is transmitting a primary packet.

Theorem 2: Under the MCSP algorithm the network is stable for all request generation rate vectors in Λ^{MCSP}.

Proof is provided in [16]. Clearly, the guaranteed stability region is greater than the capacity region without cooperation. With respect to request generation rates of primary users, the guaranteed stability region under the MCSP algorithm remains the same as for the case without cooperation.

V. Simulation Results

In this section we observe the performance of the DCSP and MCSP algorithms through simulations using C-programming language. For simplicity we consider a network with 3 non-interfering secondary base-stations and one primary and secondary user in each of these 3 small cells; there is no other primary user in the network. We consider symmetric request generation: request generation rates by primary (and secondary) users in each small cell are equal. For convenience, we denote the sum of request generation rates of all primary users in the entire network as $\lambda^{(p)}$ and the request generation rates of every secondary user as $\lambda^{(s)}$ respectively. We use the following parameters: B is 200, C is 50, T is 100, p

is 0.7, $|F^{(p)}|$ and $|F^{(s)}|$ respectively, are both equal to 400. There is one popularity state and the popularity of primary and secondary files have a Zipf distribution with parameter 0.8. All simulations are run for 2,000,000 slots. For comparison we use a non-cooperative protocol (details can be found in [16]) wherein all primary user requests are served by PB.

In Fig. 2 we compare the base-10 logarithm of time-averaged *response delay* for primary packets under both DCSP and the non-cooperative algorithm for different values of primary user request generation rates. The response delay for every transmitted packet is measured as the time between generation of the corresponding file request by a user and the time slot when the packet is successfully transmitted to that user. Average primary (respectively secondary) response delay is obtained by averaging response delays of all primary (resp. secondary) packets that are transmitted during the simulation run-time[1]. Plotting base-10 logarithm values allow us to view relatively high values of the time-averaged delay along with smaller values. Two values of $\lambda^{(s)}$: $\frac{0.1}{50}$ and $\frac{0.3}{50}$ are used; for each value of $\lambda^{(s)}$, the value of $\lambda^{(p)}$ is varied from $\frac{0.1}{50}$ to $\frac{0.8}{50}$. Note that maximum $\lambda^{(p)}$ that can be satisfied without cooperation is $\frac{0.7}{50}$; as a result, average delay is very high without cooperation for $\lambda^{(p)} = \frac{0.7}{50}$ and $\frac{0.8}{50}$. From Fig. 2 we observe that for these primary user request generation rates, average primary response delay is lowered under DCSP since the system is stable under DCSP. The difference in average delays is more pronounced as $\lambda^{(p)}$ increases or $\lambda^{(s)}$ decreases.

In Fig. 3 we plot base-10 logarithm of time-averaged primary and secondary response delay and the average number of primary files cached versus $\lambda^{(s)}$ when $\lambda^{(p)}$ is $\frac{0.2}{50}$. We plot these values for the case without cooperation, under the DCSP algorithm, and under the MCSP algorithm with different values of the penalty parameter V. From Fig. 3a we observe the primary users do not benefit under the DCSP algorithm as compared to the case of no cooperation. This is due to relatively high request generation rates by secondary users as compared to $\lambda^{(p)}$ because of which primary user requests are not served very often under the DCSP algorithm. However, under the MCSP algorithm, average primary response delay is small as primary user requests are served with higher priority by secondary users over secondary user requests. Further, this delay improves with higher $\lambda^{(s)}$. This is because with higher $\lambda^{(s)}$ there is higher backlog of secondary user requests queued at each secondary base-station. As a result, each secondary base-station caches more primary files with increasing $\lambda^{(s)}$.

[1]Note that average response delay is directly proportional to average length of request-queues, by Little's law. Lower queue length implies lower response delay and vice-versa.

We observe from Fig. 3b that when $50\lambda^{(s)}$ is less than 0.75, average secondary response delay decreases under cooperation when DCSP or MCSP with penalty parameter $V = 0.01$ are used. This is because, as observed from Fig. 3c under both these algorithms 199 out of every 200 cache positions at each secondary base-station are filled by primary files[2]. This in turn reduces opportunities for secondary base-stations to satisfy multiple secondary file requests in a period. For higher $\lambda^{(s)}$ however, the system is unstable without cooperation and both these algorithms perform better than the case without cooperation. Comparing the average secondary response delay under MCSP with $V = 0.01$ and that under DCSP also shows that, it is not guaranteed that the average secondary response delay is lowered under MCSP for any value of V. However, for some choices of V (for example $V = 0.02$ in Fig. 3) the MCSP algorithm can lower average secondary response delay by striking a balance between the number of primary files cached versus system stability. We observe that, for this network, MCSP with $V = 0.02$ is beneficial for both primary and secondary users in terms of average response delay. For this value of V, the gain in secondary packet transmission opportunities in any period (compared to DCSP or MCSP with $V = 0.01$) due to higher number of cached secondary files, offsets the reduction in secondary packet transmission opportunities due to higher proportion of primary packets being transmitted by PB.

VI. Conclusion

In this work we studied cooperative caching in cognitive radio networks. Using Lyapunov drift techniques we proposed a throughput-optimal and a sub-optimal caching and scheduling policy. In future we will extend this analysis to more general settings such as multiple secondary base-stations per primary user and multiple channels.

Acknowledgement

This material is based upon work supported by the National Science Foundation under grant number NSF award No. 1422153.

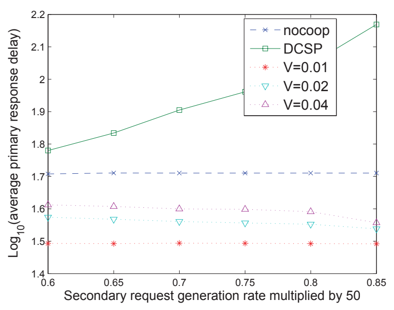
References

[1] I. Maric, R. Yates, and G. Kramer, "Capacity of interference channels with partial transmitter cooperation," *IEEE Trans. Inf. Theory*, vol. 53, no. 10, pp. 3536–3548, 2007.

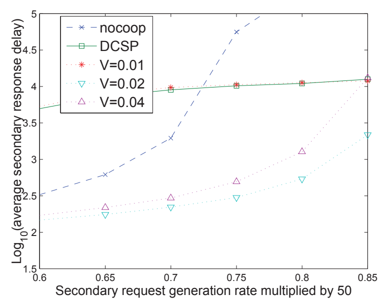
[2] I. Marić, A. Goldsmith, and G. Kramer, "On the capacity of interference channels with one cooperating transmitter," *European Transactions on Telecommunications*, vol. 19, no. 4, pp. 405–420, 2008.

[3] G. Kramer, M. Gastpar, and P. Gupta, "Cooperative strategies and capacity theorems for relay networks," *IEEE Trans. Inf. Theory*, vol. 51, no. 9, pp. 3037–3063, 2005.

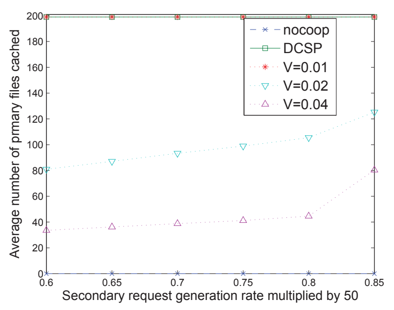
[2]Recall, lower V implies higher number of primary files to be cached with exactly 199 primary files being cached when V is 0.



(a) Average primary response delay



(b) Average secondary response delay



(c) Average number of primary files cached

Fig. 3. Plot of base-10 logarithms of time-averaged primary and secondary response delay and the average number of primary files cached per secondary base-station versus $50\lambda^{(s)}$ when $\lambda^{(p)}$ is $\frac{0.2}{50}$ under no cooperation, DCSP algorithm, and the MCSP algorithm for parameter V=0.01, 0.02 and 0.04.

[4] O. Simeone, Y. Bar-Ness, and U. Spagnolini, "Stable throughput of cognitive radios with and without relaying capability," *IEEE Trans. Commun.*, vol. 55, no. 12, pp. 2351–2360, 2007.

[5] I. Krikidis, N. Devroye, and J. Thompson, "Stability analysis for cognitive radio with multi-access primary transmission," *IEEE Trans. Wireless Commun.*, vol. 9, no. 1, pp. 72–77, 2010.

[6] A. Fanous and A. Ephremides, "Stable throughput in a cognitive wireless network," *IEEE Journal on Selected Areas in Communications*, vol. 31, no. 3, pp. 523–533, 2013.

[7] A. El-Sherif, A. Sadek, and K. Liu, "Opportunistic multiple access for cognitive radio networks," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 4, pp. 704–715, 2011.

[8] R. Urgaonkar and M. Neely, "Opportunistic cooperation in cognitive femtocell networks," *IEEE Journal on Selected Areas in Communications*, vol. 30, no. 3, pp. 607–616, 2012.

[9] N. Golrezaei, K. Shanmugam, A. Dimakis, A. Molisch, and G. Caire, "Femtocaching: Wireless video content delivery through distributed caching helpers," in *Proc. of IEEE Infocom*, 2012, pp. 1107–1115.

[10] N. Golrezaei, A. G. Dimakis, and A. F. Molisch, "Wireless device-to-device communications with distributed caching," in *IEEE International Symposium on Information Theory Proceedings (ISIT)*, 2012, pp. 2781–2785.

[11] N. Golrezaei, A. Molisch, and A. Dimakis, "Base-station assisted device-to-device communications for high-throughput wireless video networks," in *Communications (ICC), 2012 IEEE International Conference on*, June 2012, pp. 7077–7081.

[12] J. Zhao, W. Gao, Y. Wang, and G. Cao, "Delay-constrained caching in cognitive radio networks," in *Proc. of IEEE Infocom*, 2014, pp. 2094–2102.

[13] J. Zhao and G. Cao, "Spectrum-aware data replication in intermittently connected cognitive radio networks," in *Proc. of IEEE Infocom*, 2014, pp. 2238–2246.

[14] M. Amble, P. Parag, S. Shakkottai, and L. Ying, "Content-aware caching and traffic management in content distribution networks," in *Proc. of IEEE Infocom*, 2011, pp. 2858–2866.

[15] N. Abedini and S. Shakkottai, "Content caching and scheduling in wireless broadcast networks with elastic and inelastic traffic," in *WiOpt*, May 2011, pp. 125–132.

[16] D. Das and A. Abouzeid, "Cooperative caching for shared spectrum networks," 2014. [Online]. Available: http://www.ecse.rpi.edu/homepages/abouzeid/preprints/2015icc.pdf

[17] P. Gupta and P. Kumar, "The capacity of wireless networks," *IEEE Transactions on Information Theory*, vol. 46, no. 2, pp. 388–404, Mar 2000.