Recouping Opportunistic Gain in Dense Base Station Layouts Through Energy-Aware User Cooperation

Qing Wang*† and Balaji Rengarajan*
*Institute IMDEA Networks, Madrid, Spain †University Carlos III of Madrid, Spain
Email: {qing.wang, balaji.rengarajan}@imdea.org

***Abstract*—To meet the increasing demand for wireless capacity, future networks are likely to consist of dense layouts of small cells. Thus, the number of concurrent users served by each base station (BS) is likely to be small which results in diminished gains from opportunistic scheduling, particularly under dynamic traffic loads. We propose user-initiated *BS-transparent* traffic spreading that leverages user-to-user communication to increase BS scheduling flexibility. The proposed scheme is able to increase opportunistic gains and improve user performance. For a specified tradeoff between performance and power expenditure, we characterize the optimal policy by modeling the system as a Markov decision process and also present a heuristic algorithm that yields significant performance gains. Our simulations show that, in the performance-centric case, average file transfer delays are lowered by up to 20% even in homogeneous scenarios, and up to 50% with heterogeneous users. Further, we show that the bulk of the performance improvement can be achieved with a small increase in power expenditure, e.g., in an energy-sensitive case, up to 78% of the performance improvement can be typically achieved at only 20% of the power expenditure of the performance-centric case.**

I. Introduction

Opportunistic scheduling [1, 2] was proposed for multiuser wireless communication networks to exploit fluctuating channel conditions, aiming to improve performance. In cellular networks, opportunistic schedulers use knowledge of the channels between the base station (BS) and users to schedule those with favorable channel conditions, thus improving overall throughput.

The performance of opportunistic scheduling algorithms has been commonly investigated under the assumption of a static user population with infinitely backlogged queues [2], i.e., the BS always has data to transmit to each user. However, a more realistic setting is one with a time-varying user population and stochastic traffic loads. In such a setting, the performance of opportunistic scheduling algorithms can be very different [3, 4]. The impact of a time-varying user population is eased when cell sizes are large since the BS is likely to always have a large number of users to choose from for scheduling purposes. However, as cell sizes in future wireless networks shrink in response to increasing demands for capacity [5], the average number of users served by a BS will decrease and the burstiness will increase. Since opportunistic gain scales as a concave function of the user population [6], presently used scheduling algorithms are prone to losing effectiveness in small cells with dynamic traffic load. An approach that has been proposed for systems with dynamic traffic load is to take into account each user's backlog during BS scheduling. Authors in [7–9] propose BS schedulers that try to maximize opportunistic gain as well as balance users' backlogs. Among these, [7, 8] propose the throughput-optimal MaxWeight and Exponential rules, respectively, and [9] proposes a policy named log rule to improve delay performance. All these policies necessitate changes at the BS and have not been deployed yet.

In this paper, we propose an alternate user-initiated *BS-transparent* (i.e., without changes at the BS) methodology. We focus on the downlink case which accounts for most of the traffic in a cellular network [10], and on best-effort traffic, web browsing or http live streaming where mobile devices request files (chunks of content) which are then sent to users after some fetching and queueing delay. We leverage the multiple radio interfaces (e.g. 3G, WiFi) available in most smartphones, to spread traffic among users. Users keep track of their backlogs at the BS and balance traffic requests across users, aiming to maximize the BS's long-term scheduling options and hence increase opportunistic gains. Note that this strategy does not exploit current/short-term channel conditions, nor is it a relaying scheme.

The proposed methodology certainly incurs additional power expenditure due to forwarding requests and files among users. Mobile devices with scarce energy resources necessitate careful power management because excessive traffic spreading can result in unacceptably high penalties in terms of power expenditure. Thus, the degree of spreading has to be carefully chosen, and the tradeoff between performance improvement and additional power expenditure must be taken into account. In this paper, we develop an energy-aware traffic spreading policy that can be calibrated based on desired tradeoff between performance and power expenditure. We summarize our main contributions as follows:

1) We propose a novel traffic spreading policy to increase opportunistic gains by energy-aware user cooperation.
2) We formulate the problem of determining the optimal spreading policy under a specified tradeoff between performance and energy as a Markov decision problem, and study properties of the corresponding optimal

978-1-4673-5828-6/13/$31.00 ©2013 IEEE

policy in a two-user scenario.

3) We propose a tractable heuristic for the multi-user case that uses the two-user solution as a building block.
4) Based on realistic Rayleigh fading channels, we provide simulation results that demonstrate file transfer delays be reduced by up to 50% using the proposed methodology; and that significant gains (up to 78% of the gain) are typically achieved at only 20% of the power expenditure of the performance-centric case.

The rest of this paper is organized as follows: related work is summarized in Sec. II, followed by the traffic spreading policy, system model, and the dynamic programming formulation in Sec. III, IV and V respectively. The setup for numerical evaluation and simulation is presented in Sec. VI. The properties of the optimal traffic spreading policy are described in Sec. VII, and a tractable heuristic for multi-user scenarios is developed in Sec. VIII. Simulation results and evaluation of the heuristic are presented in Sec. IX. Finally, our conclusions and future work are presented in Sec. X.

II. Related Work

As we describe in Sec. IV, we formulate the problem of determining the optimal traffic spreading policy as a dispatching problem. Here, we discuss some related work on this topic. A dispatching system typically consists of a dispatcher and several servers. The role of the dispatcher is to route new jobs to a server based on dispatching policies. The dispatching problem has received a lot of attention since the landmark work in [11]. The author considers a homogeneous model with Poisson arrivals and exponentially distributed job size, and show that when the queue lengths of the servers (number of jobs) are known, Join the Shortest Queue (JSQ) minimizes the average waiting time in the queues. When queue lengths are unavailable, [12] shows that Round Robin is optimal. A number of papers focus on the settings where the queue lengths are not observed, while the size of each arrival job is available, e.g., authors in [13, 14] work on policies that dispatch jobs according to job size.

In contrast to above papers, our model only has one shared server whose service rate is affected by the dispatching policy. The model in [15] includes the case of a shared server and is the closest to ours. However, this paper like the others makes the assumption that the service rate is constant and does not depend on the instantaneous queue states. In our work, the service rate depends on the channel states as well as queue states, making the problem more complex. The emphasis in all the above papers is on performance, whereas in our case we additionally consider the implications of the dispatching decisions on energy cost which adds a facet that has not been explored before.

III. Traffic Spreading

The traffic spreading we propose includes a dispatcher that resides on each mobile device. If the dispatcher detects that traffic spreading can be beneficial, it uses the user-to-user link to forward the new request to a chosen user who then forwards it to the BS. The chosen user, upon receiving the file from the BS (at some future time), forwards it to the user who originated the request through the user-user link.

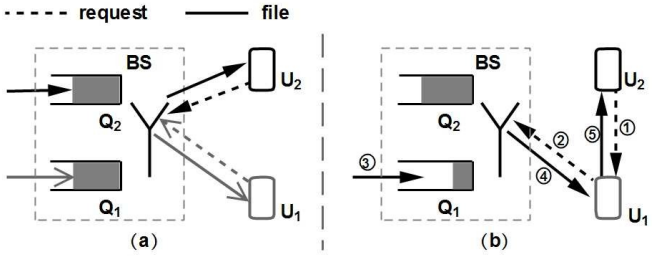


Figure 1: An example of traffic spreading: (a) No traffic spreading; (b) Traffic spreading from U_2 to U_1.

Let us consider the example shown in Fig. 1. We depict a scenario with two users, U_1 and U_2 being served by the BS. The queues, Q_1 and Q_2, depict the number of files waiting to be sent to each user at the BS. In this scenario, the users perceive similar channel statistics and we consider below the case where they generate similar traffic loads to illustrate the spreading mechanism. In Fig. 1(a), since the queues at the BS are balanced, the dispatchers of the users would ideally detect that traffic spreading is not beneficial. Thus users send their new requests to the BS directly. In Fig. 1(b), there are many more files in Q_2 than in Q_1 which is nearly empty. Under this case, the dispatcher of U_2 would ideally detect that traffic spreading is beneficial since in the near term the risk that the BS has no files to send to U_1 is high. Thus when a new request is generated by U_2, it forwards the request to U_1, who will send it to the BS. When U_1 receives the corresponding file from the BS, it forwards the file to U_2 through the user-to-user link.

The criteria used by dispatchers is that they are aware of the channel statistics of all the users, and can estimate/infer users' backlogs at the BS. Note that dispatchers do not exploit current channel conditions, and can not predict when a new request will be served or the instantaneous channel conditions at that time. Each user keeps track of the number of files it is waiting to receive directly from the BS, and shares this information periodically with other users. Users also measure and exchange their perceived channel statistics with other users. Moreover, users are aware of, or can easily infer the BS scheduling policy.

IV. System Model

We model the system in continuous time with N users attached to a single BS, where the set of users is denoted by $\mathcal{I} = \{1, 2, ..., N\}$. The arrival requests of users are modeled as Poisson processes with mean arrival rate vector $\boldsymbol{\lambda} = \{\lambda_1, \lambda_2, ..., \lambda_N\}$, and are assumed to be independent across users. The requested file sizes of users are exponentially distributed, with mean file size vector $\boldsymbol{\theta} = \{\theta_1, \theta_2, ..., \theta_N\}$.

Channel model: The wireless channel is assumed to be time-varying and the channel instance at time t between the BS and user i is denoted by $S^i(t)$, which can take values from the set $\mathcal{S} = \{c_1, c_2, \cdots, c_K\}$. Further, we assume channels perceived by two different users are independent. We denote by $p^i_{c_k}, c_k \in \mathcal{S}$, the probability that user i perceives channel c_k at any time t. Each user i shares its channel probability vector $\boldsymbol{p}^i = \{p^i_{c_1}, p^i_{c_2}, ..., p^i_{c_K}\}$ with all the other users, where we assume all users are within the transmission range of each other, which is expected to be the case in picocell/femtocell [16, 17]. We denote by $\boldsymbol{c}(t) = \{c^i(t), c^i(t) \in \mathcal{S}\}$, the vector of current channel states of users at time t and by $\boldsymbol{\mathcal{S}}$, the set of possible vector channels. Conditional on the channel state being $c_k \in \mathcal{S}$, we define a non-negative value $R_i^{c_k}$, which denotes the data rate (rate supported by the channel) in bits/second of user i.

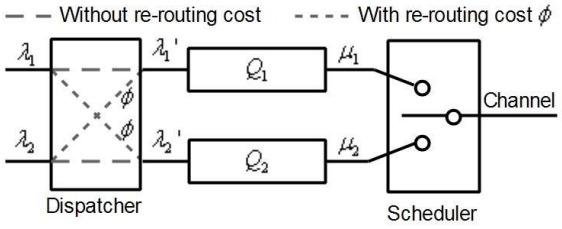


Figure 2: Components of the system model

Fig. 2 depicts our system model consisting of three main components, i.e., the BS scheduler, the queues at the BS, and a dispatcher that models the joint behavior of all the mobile devices. The BS maintains a separate queue corresponding to each user, and we denote by $\boldsymbol{Q}(t) \equiv (Q_i(t), i \in \mathcal{I}) \in \mathbb{Z}_+^N$, the number of files waiting to be sent by the BS to each user at time t, i.e., the number of unsatisfied requests.

Scheduling policy: We model the BS scheduling policy through $\xi_i(\boldsymbol{q}, \boldsymbol{c})$, denoting the probability that user i is selected to be served by the scheduler, conditional on the queues being in state $\boldsymbol{q}$, and channel vector being $\boldsymbol{c}$. Under the scheduling policy $\boldsymbol{\xi}_i$, the average state-dependent service rate of user i is denoted as $\mu_i(\boldsymbol{q})$.

Dispatching policy: The dispatching policy used across all users is defined through the probability matrix $\boldsymbol{\sigma}(\boldsymbol{q})$ as:

$$\boldsymbol{\sigma}(\boldsymbol{q}) = \begin{bmatrix} \sigma_1^1(\boldsymbol{q}) & \cdots & \sigma_1^N(\boldsymbol{q}) \\ \cdots & \cdots & \cdots \\ \sigma_N^1(\boldsymbol{q}) & \cdots & \sigma_N^N(\boldsymbol{q}) \end{bmatrix}, \tag{1}$$

where $\sigma_j^i(\boldsymbol{q})$ denotes the probability of dispatching user j's request to user i, conditional on the queues being in state $\boldsymbol{q}$. The set of all the possible dispatching policies is defined as

$$\mathcal{C} \equiv \{\boldsymbol{\sigma}(\boldsymbol{q}) : \sum_{i \in \mathcal{I}} \sigma_j^i(\boldsymbol{q}) = 1,\ 0 \le \sigma_j^i(\boldsymbol{q}) \le 1,\ j \in \mathcal{I}\}\ . \tag{2}$$

The rate at which files arrive to user i's queue at the BS is denoted by $\lambda_i'(\boldsymbol{q}, \boldsymbol{\sigma}(\boldsymbol{q}))$, which corresponds to the rate of requests sent by user i to the BS (including forwarded requests from other users) .

Performance metrics: The metrics we use are average file transfer delay and *the re-routing cost*, i.e., additional power expenditure induced by traffic spreading. We assume there is no queueing delay for the user-to-user transfer, but each transfer certainly adds additional forwarding delay and power expenditure. We model these additional delay and power expenditure incurred for a single file of user j as a function of its mean file size θ_j, i.e., $\eta_j^i(\theta_j)$ and $\phi_j^i(\theta_j)$, and define $\eta_j^j(\theta_j) = 0$, $\phi_j^j(\theta_j) = 0$. Note that a very high $\phi_j^i(\theta_j)$ can be used to model the case where the large distance between users makes the communication between them infeasible. Moreover, let $\bar{\eta}$ and $\bar{\phi}$ denote the average additional forwarding delay and power expenditure, respectively. The objective function we seek to minimize is

$$\left(\bar{D} + \bar{\eta}\right) + w \cdot \bar{\phi}\ , \tag{3}$$

where $\bar{D}$ is the average BS-user delay and w is a weight associated with the power expenditure that determines the tradeoff between the delay and power expenditure.

V. Dynamic Programming Formulation

Consider the process $(\boldsymbol{Q}(t), t \ge 0)$ initiated in state $\boldsymbol{Q}(0) = \boldsymbol{q}(0)$ and evolving under a dispatching policy $\boldsymbol{\sigma}$ and a scheduling policy $\boldsymbol{\xi}$. We define the vector $\boldsymbol{\mu}(\boldsymbol{q}) \equiv (\mu_i(\boldsymbol{q}), i \in \mathcal{I})$ as the average service rate of users, conditional on the queues being in state $\boldsymbol{q}$. Clearly, $\boldsymbol{\mu}(\boldsymbol{q})$ depends on the scheduling policy used at the BS. For a given scheduling policy, the average service rate $\mu_i(\boldsymbol{q})$ is given as

$$\mu_i(\boldsymbol{q}) = \sum_{\boldsymbol{c} \in \boldsymbol{\mathcal{S}}} \left(\prod_{i=1}^{N} p_{c^i} \right) \xi_i(\boldsymbol{q}, \boldsymbol{c}) \cdot R_i^{c^i}\ . \tag{4}$$

We assume over an epoch, each queue $i \in \mathcal{I}$ is served at constant service rate $\mu_i(\boldsymbol{q})$. Since the channel varies much faster than the queue dynamics, we use the service rate averaged across channel fluctuations at each queue state. A rigorous justification of the service rate with consideration of packet or file dynamics can be found in [18]. Conditional on the process being in state $\boldsymbol{q}$ and under the dispatching policy $\boldsymbol{\sigma}$, the file arrival rate to Q_i is given by

$$\lambda_i'(\boldsymbol{q}, \boldsymbol{\sigma}(\boldsymbol{q})) = \sum_{j \in \mathcal{I}} \sigma_j^i(\boldsymbol{q}) \lambda_j, \quad i \in \mathcal{I}\ . \tag{5}$$

Our objective is to find the right $\boldsymbol{\sigma}(\boldsymbol{q}) \in \mathcal{C}$ for each state $\boldsymbol{q}$ that minimizes (3). Using Little's law, (3) becomes

$$\left(\frac{|\ E[\boldsymbol{Q}]\ |}{|\ \boldsymbol{\lambda}\ |} + \bar{\eta} \right) + w \cdot \bar{\phi}\ , \tag{6}$$

where $|\cdot|$ denotes the L_1 norm.

Under a fixed policy $\boldsymbol{\sigma}$, the process $(\boldsymbol{Q}(t), t \ge 0)$ is a Markov process on $\mathbb{Z}_+^N$ with state-dependent (depends on both channel and queue states) transition rate. For convenience, we uniformize $\boldsymbol{Q}(t)$ following [19]. For any $\boldsymbol{q} \in \mathbb{Z}_+^N$, we make the following definitions:

$$D_i \boldsymbol{q} \equiv [\boldsymbol{q} - \boldsymbol{e}_i]^+\ , \qquad A_i \boldsymbol{q} \equiv \boldsymbol{q} + \boldsymbol{e}_i\ , \tag{7}$$

where $\boldsymbol{e}_i$ is a $1\times N$ zero-valued vector except the i^{th} element is 1 and $\boldsymbol{q}^+ \equiv (y|y_n = \max\{0, q_n\})$. The D_i in (7) denotes a file is successfully transmitted from the BS to user i, and A_i means a file arrives to Q_i at the BS.

Let $\varphi \geq |\boldsymbol{\lambda}| + \max_{\boldsymbol{q}} |\boldsymbol{\mu}(\boldsymbol{q})|$. Let τ_k denote the time of the k^{th} transition of $\boldsymbol{Q}(t)$ and $\tau_0 = 0$. Also, let $\boldsymbol{Q}_k = \lim_{t\downarrow\tau_k} \boldsymbol{Q}(t)$. Then, under policy $\boldsymbol{\sigma}(\boldsymbol{Q})$, the process $\boldsymbol{Q}(t)$ can be viewed as having a state-independent event transition rate of φ, and the transition probabilities are given by

$$P\big(\boldsymbol{Q}_{k+1} = A_i\boldsymbol{q} \mid \boldsymbol{Q}_k = \boldsymbol{q}\big) = \lambda_i'(\boldsymbol{q}, \boldsymbol{\sigma}(\boldsymbol{q}))/\varphi \ , \tag{8}$$

$$P\big(\boldsymbol{Q}_{k+1} = D_i\boldsymbol{q} \mid \boldsymbol{Q}_k = \boldsymbol{q}\big) = \mu_i(\boldsymbol{q})/\varphi \ , \tag{9}$$

$$P\big(\boldsymbol{Q}_{k+1} = \boldsymbol{q} \mid \boldsymbol{Q}_k = \boldsymbol{q}\big) = 1 - \big(|\boldsymbol{\lambda}| + |\boldsymbol{\mu}(\boldsymbol{q})|\big)/\varphi. \tag{10}$$

Moreover, we define the function $\boldsymbol{f}(\cdot)$ as

$$\boldsymbol{f}\left(\boldsymbol{q}, \boldsymbol{\sigma}(\boldsymbol{q})\right) = \sum_{j\in\mathcal{I}} \sum_{i\in\mathcal{I}} \sigma_j^i(\boldsymbol{q}) \cdot \left[\eta_j^i(\theta_j) + w \cdot \phi_j^i(\theta_j)\right] \ . \tag{11}$$

The cost (our objective in (6)) under policy $\boldsymbol{\sigma}$ over $[0, \tau_k)$ when starting from an initial queue state $\boldsymbol{q}$ is

$$\mathbb{E}_{\boldsymbol{q}}^{\boldsymbol{\sigma}}\left[\int_0^{\tau_k} \left[\frac{|\boldsymbol{Q}(t)|}{|\boldsymbol{\lambda}|} + \boldsymbol{f}\big(\boldsymbol{Q}(t), \boldsymbol{\sigma}(\boldsymbol{Q}(t))\big)\right] dt\right] \ , \tag{12}$$

which, by ignoring the constant multiplier φ^{-1}, is equal to:

$$V_k^{\boldsymbol{\sigma}}(\boldsymbol{q}) \equiv \mathbb{E}_{\boldsymbol{q}}^{\boldsymbol{\sigma}}\left[\sum_{l=0}^{k-1}\left(\frac{|\boldsymbol{Q}_l|}{|\boldsymbol{\lambda}|} + \boldsymbol{f}\left(\boldsymbol{Q}_l, \boldsymbol{\sigma}(\boldsymbol{Q}_l)\right)\right)\right] \ . \tag{13}$$

Then the average cost under policy $\boldsymbol{\sigma}$ when starting from state $\boldsymbol{q}$ is given as follows:

$$J_{\boldsymbol{q}}^{\boldsymbol{\sigma}} = \lim_{k\to\infty} \sup \frac{1}{k} V_k^{\boldsymbol{\sigma}}(\boldsymbol{q}) \ . \tag{14}$$

The objective function given in (6) seeks to find the minimal average cost J^* and the corresponding optimal control $\boldsymbol{\sigma}^*(\boldsymbol{q})$, which fits the classical dynamic programming [19]. Under all possible dispatching probabilities $\boldsymbol{\sigma}(\boldsymbol{q}) \in \mathcal{C}$, J^* is well-defined, independent of the initial state $\boldsymbol{q}$, and satisfies Bellman's equation:

$$\begin{aligned} J^* = \min_{\boldsymbol{\sigma}(\boldsymbol{q})\in\mathcal{C}} \Bigg\{ & \frac{|\boldsymbol{q}|}{|\boldsymbol{\lambda}|} + \boldsymbol{f}(\boldsymbol{q}, \boldsymbol{\sigma}(\boldsymbol{q})) \\ & + \mathbb{E}^{\boldsymbol{\sigma}(\boldsymbol{q})}\big\{\big[h\big(\boldsymbol{Q}_{k+1}\big) - h\big(\boldsymbol{Q}_k\big)\big] \mid \boldsymbol{Q}_k = \boldsymbol{q}\big\}\Bigg\} \\ = & \frac{|\boldsymbol{q}|}{|\boldsymbol{\lambda}|} + \sum_{i\in\mathcal{I}} \frac{\mu_i(\boldsymbol{q})}{\varphi}\Big[h\big(D_i\boldsymbol{q}\big) - h\big(\boldsymbol{q}\big)\Big] + \min_{\boldsymbol{\sigma}(\boldsymbol{q})\in\mathcal{C}} \sum_{j\in\mathcal{I}} \sum_{i\in\mathcal{I}} \\ & \frac{\lambda_j \sigma_j^i(\boldsymbol{q})}{\varphi}\Big\{\eta_j^i(\theta_j) + w\cdot\phi_j^i(\theta_j) + \big[h\big(A_i\boldsymbol{q}\big) - h\big(\boldsymbol{q}\big)\big]\Big\} \ , \end{aligned} \tag{15}$$

where $h(\boldsymbol{q}) = J(\boldsymbol{q}) - J(\boldsymbol{q}_s)$ is a relative function with $\boldsymbol{q}_s$ being a reference state. The optimal dispatching policy $\boldsymbol{\sigma}^*(\boldsymbol{q})$ that minimize (15) can be calculated through methods such as the value iteration or policy iteration from the dynamic programming framework [19].

VI. Numerical Evaluation & Simulation Setup

In this section, we present the parameters for numerical evaluation and simulation. We consider two channel-aware scheduling policies: a queue-unaware policy where $\xi_i(\boldsymbol{q}, \boldsymbol{c})$ only depends on the set of non-zero elements in $\boldsymbol{q}$, and a queue-aware policy that have a stronger dependence on $\boldsymbol{q}$:

1) Queue-unaware, greedy scheduling policy: Queue-unaware means the scheduler is unaware of the queue length, but knows whether a queue is empty or not. At any time t, a greedy scheduler chooses a non-empty queue i to serve if user i has the largest instantaneous data rate.

2) Queue-aware, the log rule scheduling policy [9]: Queue-aware means the scheduler is aware of the queue length. At time t, a log rule scheduler makes decisions based on current channel state and the logarithm of queue length.

In the simulation, we assume the mean file size θ_j=1MB, under which the additional power expenditure $\phi_j^i(\theta_j)$ is 1 Joule for all $i \neq j, i, j \in \mathcal{I}$. The channel is a Rayleigh fading channel and the Signal-to-Noise-Ratio (SNR) is assumed to be constant during a time slot. We denote the channel bandwidth as B, and the distance between user i and the BS as d_i. Conditional on the channel being in state $c_k \in \mathcal{S}$, the data rate $R_i^{c_k}$ is given by the Shannon formula with a 3dB SNR backoff (to model achievable data rate):

$$R_i^{c_k} = B \cdot \log_2(1 + \text{SNR}_i(c_k)/2) \ . \tag{16}$$

The settings of the channel parameters are listed in Table I (Note that the results are not sensitive to these settings).

Table I: Channel parameters

Parameters	Value
Bandwidth (Min bandwidth in LTE)	1.4 MHz
BS Tx power spectral density	0.1/1.4 W/MHz
Noise spectral density	10^{-8}/1.4 W/MHz
Path loss exponent (Urban Area)	3
Slot time	10 ms
Doppler shift (ITU Pedestrian A)	5 Hz

We ignore the user-to-user delay (i.e., $\eta_j^i = 0, \forall i, j \in \mathcal{I}$) since the bandwidth of user-to-user link would be much higher than the cellular link. We estimate the average file transfer delay and additional power expenditure within a relative error of 2%, at a confidence interval of 95%.

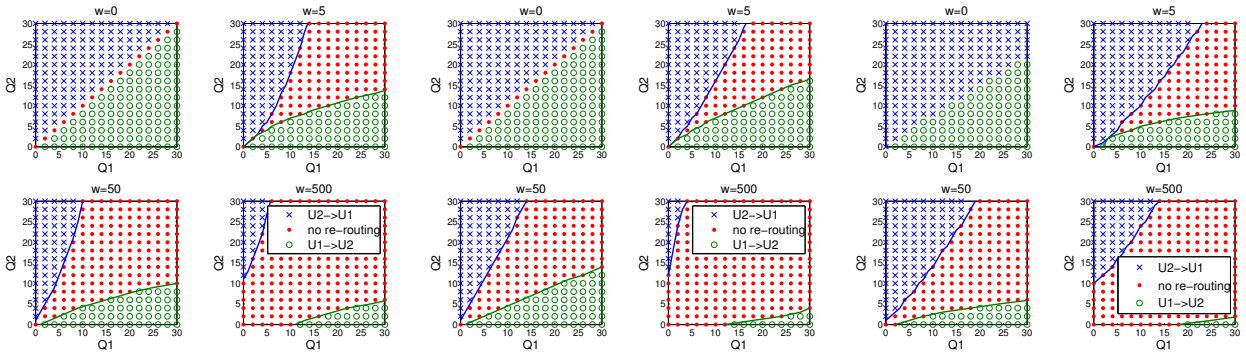
VII. Properties of the Optimal Policy

In this section, we present some properties of the optimal dispatching policy $\boldsymbol{\sigma}^*(\boldsymbol{q})$.

A. Restricting the Optimal Policy Space

The following theorem guarantees that the optimal value of the objective function can be achieved by non-randomized policies that apply deterministic rules for dispatching the arrivals at a given system state. This reduces the computational effort required to compute an optimal dispatching policy, and allows us to use value iteration in the sequel to study the structure of the optimal policy.

Theorem 1: There exists an optimal dispatching policy $\boldsymbol{\sigma}^*(\boldsymbol{q})$ such that each element ${\sigma^*}_j^i \in \{0, 1\}$.



(a) $\boldsymbol{\lambda} = \{0.2, 0.2\}$ arrivals/sec; (b) $\boldsymbol{\lambda} = \{0.2, 0.2\}$ arrivals/sec; (c) $\boldsymbol{\lambda} = \{0.19, 0.21\}$ arrivals/sec;

Figure 3: Optimal dispatching strategy as a function of queue backlogs: (a) Homogeneous scenario ($\boldsymbol{d} = \{100, 100\}$ m), greedy scheduler; (b) Homogeneous scenario ($\boldsymbol{d} = \{100, 100\}$ m), the log rule scheduler; (c) Heterogeneous scenario ($\boldsymbol{d} = \{92, 100\}$ m), greedy scheduler.

Proof: The proof is omitted (can be found in [20]. ■

B. A Two-user System

From Theorem 1, we know that under the two-user model and at any queue state $\boldsymbol{q}$, there are three reasonable controls: *i)* $\sigma(\boldsymbol{q}) = [1\ 0; 0\ 1]$; *ii)* $\sigma(\boldsymbol{q}) = [1\ 0; 1\ 0]$; *iii)* $\sigma(\boldsymbol{q}) = [0\ 1; 0\ 1]$. In the rest of this paper, we refer to these controls as *no re-routing*, $U_2 \rightarrow U_1$ and $U_1 \rightarrow U_2$, respectively. The optimal dispatching policies evaluated numerically under different scenarios are shown in Fig. 3. The axes correspond to the number of files in the users' queues, and the figure depicts the optimal dispatching strategy at each state. From this figure, we observe the following properties:

Existence of switching curves: As Fig. 3 shows, the optimal policy in all the above cases consists of a set of switching curves. Here, switching curves refer to the boundaries between contiguous regions where the same control is used in each state of the region. We conjecture that an optimum policy can be described by threshold values q_2^a and q_2^b corresponding to each value of q_1, such that

$$\sigma^*(q_1, q_2) = \begin{cases} U_1 \rightarrow U_2, & \text{if } q_2 \leq q_2^a \\ \textit{No re-routing}, & \text{if } q_2^a \leq q_2 \leq q_2^b \\ U_2 \rightarrow U_1, & \text{if } q_2 \geq q_2^b \end{cases}$$

For the two-user scenario with homogeneous users under a two-state channel model, we can prove that the optimal policy indeed possesses this structure, as shown in [20].

Performance vs. Energy consumption: Choosing a weight of 0 implies that the optimal policy is one which minimizes average file transfer delay. In the case of the homogeneous scenarios of Fig. 3 (a) and (b), this corresponds to dispatching arrivals to the shortest queue, as described in [11]. In the heterogeneous case of Fig. 3 (c), arrivals are dispatched rather to the queue with less backlog, taking into account the difference in average service rates. A higher value of the weight, w, implies that delay performance is sacrificed in order to reduce the excess power expenditure due to traffic spreading. We observe that the regions corresponding to *re-routing areas* ($U_1 \rightarrow U_2$ and $U_2 \rightarrow U_1$) diminish progressively as the weight attached to power expenditure increases. At very high values of w, traffic spreading is initiated only when the imbalance between the user queues is very large.

Switching curve shape: We observe from the results in Fig. 3 that the level of imbalance between the queues that is required for arrival re-routing to be the optimal strategy increases as the overall backlog increases. For instance, the threshold on the queue length of U_2 beyond which arrivals are re-routed to U_1 appears to be a convex, increasing function of the backlog in Q_1. The intuition behind this is that when the backlog in both queues is large, the time interval for a queue to empty out is likely to be long, and the shorter queue might yet see many arrivals even without re-routing. In such a case, the gain from dispatching requests to other users to balance the queues does not justify the associated power expenditure.

Dispatching as a function of the scheduling policy: We observe that the optimal dispatching policy under the log rule scheduler (Fig. 3(b)) consists of switching curves that favor more re-routing, especially at larger queue-lengths. For example, the threshold q_2^b prompting re-routing is lower, and does not increase as rapidly with total queue length as under the greedy scheduler (Fig. 3(a)). The log rule scheduler itself reacts to imbalance in queues, sacrificing opportunistic gain in order to balance the queues. The optimal dispatcher takes this into account, resulting in the above policy. However, as we will see in the sequel, the dispatcher complements the log rule scheduler well and the combination does achieve better performance at lower power expenditure compared to the queue-unaware greedy scheduler.

Impact of the request arrival rate: Switching curves of the optimal dispatching policy under the greedy scheduler for different request arrival rates are shown in Fig. 4. We can observe that with the increase of arrival rate, the re-routing areas decrease. One reason for this is that at lower arrival rates, the shorter queue could be emptied before the next

arrival to either user. Thus, the optimal dispatcher is more aggressive at re-routing requests even when queue lengths are larger.

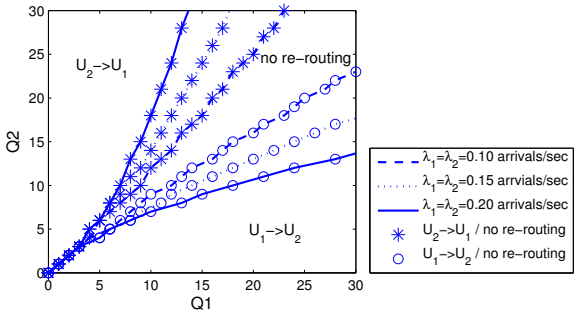


Figure 4: Switching curves of the optimal dispatching policy under different request arrival rates where $\boldsymbol{d} = \{100, 100\}$ m and $w = 5$.

VIII. A Heuristic Algorithm for Multi-user System

In scenarios with many users, numerically computing the optimal dispatching policy by solving the dynamic programming formulation (15) becomes intractable. Thus we propose a heuristic algorithm to determine the dispatching policy for a multi-user system that uses the dynamic programming solution of two-user scenarios as a building block. Here, we focus on currently deployed queue-unaware schedulers such as the greedy policy. The heuristic used to compute dispatching decisions at user i is specified below in Algorithm 1. We denote by $dp(\tilde{\boldsymbol{\lambda}}, \tilde{\boldsymbol{\mu}}, \tilde{\phi})$, the optimal dispatching policy for the two-user model, where $\tilde{\boldsymbol{\lambda}} = \{\tilde{\lambda}_1, \tilde{\lambda}_2\}$ is the vector of arrival rates , $\tilde{\boldsymbol{\mu}}(\tilde{\boldsymbol{q}}) = \{\tilde{\boldsymbol{\mu}}_1(\tilde{\boldsymbol{q}}), \tilde{\boldsymbol{\mu}}_2(\tilde{\boldsymbol{q}})\}, \tilde{\boldsymbol{q}} \in \mathbb{Z}_+^2$ specifies the queue-state dependent service rates and $\tilde{\phi} = \{\tilde{\phi}_1^2(\theta_1), \tilde{\phi}_2^1(\theta_2)\}$ is the re-routing power expenditure.

The proposed heuristic considers two options for a new request, i.e., forwarding it directly to the BS or dispatching it to the user with the least amount of work (workload) in its BS queue. To this end, all users other than the one with the least workload are treated as a single combined user, and their queues are also treated as a single combined queue. A series of two-user dynamic programming formulations are solved, where the two users are the combined user and the user with least workload. In order to determine the parameters of the two-user dynamic program, we map states where the combined queue is non-empty to states where all the component queues are non-empty in the multi-user system. The service rate of the combined queue at a state is calculated as the sum of the service rates of the component queues in the corresponding state (steps 4-6).

The users are examined in sequence, and the dispatching strategy is decided in order of decreasing workload. The arrival rates to the combined queue reflects the dispatching decisions made at all the users with higher workload than the one currently under consideration (step 9). The dynamic programming solution is computed taking into account the power expenditure associated with dispatching from the current user to the one with least workload. The state considered in the reduced dynamic program is always one where the combined user queue length is the sum of the queue lengths of the component queues. Arrivals to the current user are dispatched to the user with least workload in the multi-user system if the optimal policy in the reduced scenario is to re-route from the combined queue to the other. Note that the worst-case time-complexity of the above heuristic to obtain the dispatching decision for a new request is $O(N-1)$.

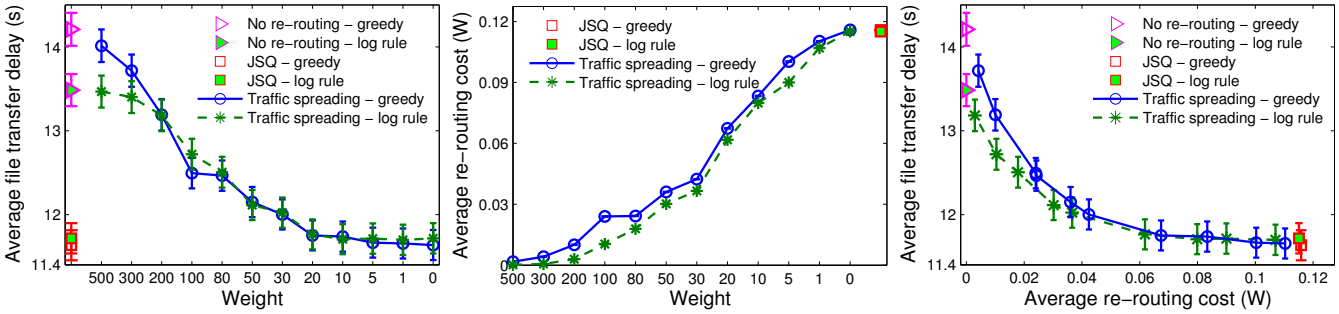
Algorithm 1 A Heuristic Algorithm for N-user System

// Heuristic to dispatch a new arrival at user i.
// **Definitions:** 1) $\boldsymbol{e}_l \equiv$ a $1 \times N$ zero-valued vector except the l^{th} element is 1; 2) $\mathbf{1}_N \equiv \sum_{l=1}^{N} \boldsymbol{e}_l$.
Input: $\boldsymbol{\lambda}$: a $1 \times N$ vector of arrival rates
$\boldsymbol{q}$: a $1 \times N$ vector of queue state
$\boldsymbol{\mu}(\boldsymbol{q})$: a matrix of queue-state dependent service rates
Output: dispatching decision at user i

1: $j \leftarrow \arg\min_{l \in \mathcal{I}} \{q_l / \mu_l(\boldsymbol{e}_l)$.
2: **if** $j \neq i$ **then**
3: $\quad \mathcal{Y}_S \leftarrow \{j\}$; $\mathcal{Y}_B \leftarrow \mathcal{I} \setminus \mathcal{Y}_S$; $\mathcal{Y}_P \leftarrow \emptyset$
4: $\quad \tilde{q}_1 \leftarrow \sum_{l \in \mathcal{I} \setminus \mathcal{Y}_S} q_l$; $\tilde{q}_2 \leftarrow q_j$
5: $\quad \tilde{\boldsymbol{\mu}}_1(\tilde{\boldsymbol{q}}) \leftarrow \begin{cases} 0, & \tilde{q}_1 = 0 \\ \sum_{l \neq j} \mu_l(\mathbf{1}_N - \boldsymbol{e}_j), & \tilde{q}_1 > 0, \tilde{q}_2 = 0 \\ \sum_{l \neq j} \mu_l(\mathbf{1}_N), & \tilde{q}_1 > 0, \tilde{q}_2 > 0 \end{cases}$
6: $\quad \tilde{\boldsymbol{\mu}}_2(\tilde{\boldsymbol{q}}) \leftarrow \begin{cases} 0, & \tilde{q}_2 = 0 \\ \mu_j(e_j), & \tilde{q}_1 = 0, \tilde{q}_2 > 0 \\ \mu_j(\mathbf{1}_N), & \tilde{q}_1 > 0, \tilde{q}_2 > 0 \end{cases}$
7: $\quad$ **while** $\mathcal{Y}_B \neq \emptyset$ **do**
8: $\quad\quad i^* \leftarrow \arg\max_{l \in \mathcal{Y}_B} \{q_l / \mu_l(\boldsymbol{e}_l)\}$
9: $\quad\quad \tilde{\lambda}_1 \leftarrow \sum_{l \in \mathcal{I} \setminus \mathcal{Y}_S} \lambda_l$; $\tilde{\lambda}_2 \leftarrow \sum_{l \in \mathcal{Y}_S} \lambda_l$
10: $\quad\quad \tilde{\phi} \leftarrow \{\phi_{i^*}^{j}(\theta_{i^*}), \phi_{j}^{i^*}(\theta_j)\}$
11: $\quad\quad \boldsymbol{\sigma} \leftarrow dp(\tilde{\boldsymbol{\lambda}}, \tilde{\boldsymbol{\mu}}, \tilde{\phi})$
12: $\quad\quad$ **if** $\boldsymbol{\sigma}(\tilde{q}_1, \tilde{q}_2) = U_1 \rightarrow U_2$ **then**
13: $\quad\quad\quad$ **if** $i^* = i$ **then**
14: $\quad\quad\quad\quad$ **return** dispatch the new request to user j
15: $\quad\quad\quad$ **else**
16: $\quad\quad\quad\quad \mathcal{Y}_B \leftarrow \mathcal{Y}_B \setminus i^*$; $\mathcal{Y}_S \leftarrow \mathcal{Y}_S \cup i^*$
17: $\quad\quad\quad$ **end if**
18: $\quad\quad$ **else if** $i^* = i$ **then**
19: $\quad\quad\quad$ **return** send the new request directly to the BS
20: $\quad\quad$ **else**
21: $\quad\quad\quad \mathcal{Y}_B \leftarrow \mathcal{Y}_B \setminus i^*$; $\mathcal{Y}_P \leftarrow \mathcal{Y}_P \cup i^*$
22: $\quad\quad$ **end if**
23: $\quad$ **end while**
24: **else**
25: $\quad$ **return** send the new request directly to the BS
26: **end if**

IX. Performance Evaluation

In this section, we evaluate our proposed traffic spreading policy through simulations and demonstrate the tradeoff between performance improvement and additional power



(a) Average delay versus weight (b) Average re-routing cost versus weight (c) Average delay versus re-routing cost

Figure 5: Performance under two-user homogeneous scenarios where $\boldsymbol{\lambda} = \{0.2, 0.2\}$ arrivals/sec and $\boldsymbol{d} = \{100, 100\}$ m.

expenditure resulting from the dynamic programming formulation and our multi-user heuristic. The parameters used are summarized in Sec. VI. To evaluate the proposed traffic spreading, we compare it with two dispatching policies:

1) No re-routing: Under this dispatching policy, when a request of user i is generated, user i sends it directly to the BS. Thus there is no additional power expenditure.

2) Join the Shortest Queue (JSQ): Under this dispatching policy, when there is a new request generated by user i, user i sends it to us er j that has the least amount of work left. If $i \neq j$, then additional power expenditure occurs.

A. The Two-user Scenario

Homogeneous scenarios: The simulation results under the greedy and log rule scheduling policies are shown in Fig. 5 (a)-(c) for a homogeneous scenario where both users are at exactly the same distance from the BS and have identical traffic demands. Fig. 5 (a) shows that JSQ results in the lowest average delay independent of the scheduling policy, as expected. Under performance-centric case ($w = 0$), traffic spreading can reduce the average delay as much as JSQ does, independent of the scheduling policies. Both JSQ and the traffic spreading have delay improvement up to 18% (greedy) and 14% (log rule) compared to no re-routing. The results in Fig. 5(b) show that these strategies also correspond to the highest power expenditure. Traffic spreading re-routes as much as JSQ does when $w = 0$. Under energy-sensitive cases ($w > 0$), increasing the weight of re-routing cost results in the energy consumption decreasing rapidly along with increasing average file transfer delay.

Moreover, we observe from Fig. 5 (a) that under large weight ($w \geq 200$), the delay performance under the log rule scheduler is better than under the greedy scheduler. However, in the performance-centric scenarios, traffic spreading under the greedy scheduler does achieve similar delay performance. In general, the rate of re-routing and the power expenditure is lower under the log rule scheduler. This is because the log rule scheduling policy already tries to balance the queues, while the greedy scheduler does not.

The tradeoff between the average delay and re-routing cost under traffic spreading can be seen from Fig. 5 (c). We see again that traffic spreading is generally able to achieve the same delay performance at lower power expenditure compared to the greedy scheduler. A very interesting observation indicated by this figure is that most of the delay performance gain can be achieved with a small increase in power expenditure. For example, when $w = 0$, the (maximal) delay performance gain is 18%, and the (maximal) average re-routing cost is $0.12W$. However, when $w = 100$, the delay performance gain is 14% and the re-routing cost is about $0.025W$. This means under the traffic spreading 78% of the maximal performance gain can be achieved with only 20% of the maximal re-routing cost.

The impact of arrival rates on performance is shown in Fig. 6, where we scale $\boldsymbol{\lambda}$, while keeping $\boldsymbol{d}$ and w unchanged. Even at low loads, traffic spreading results in performance gains, with the gains increasing as $\boldsymbol{\lambda}$ increases. For example, when $\boldsymbol{\lambda} = \{0.175, 0.175\}$ arrivals/sec, the gain is 16%; and it increases to 23% when $\boldsymbol{\lambda} = \{0.215, 0.215\}$ arrivals/sec. Note that, with the chosen weight, traffic spreading achieves similar delay performance to JSQ at all traffic loads while consuming less than half the extra energy.

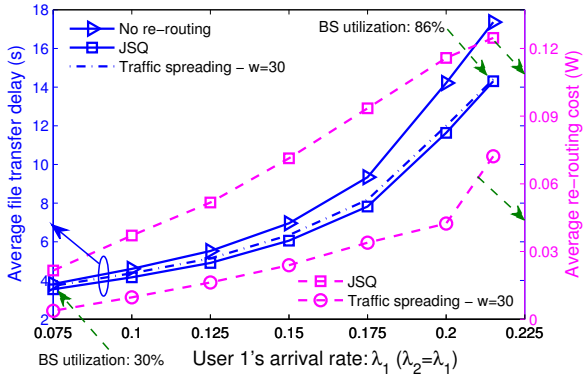


Figure 6: Performance vs. arrival rate under two-user homogeneous scenarios where $\boldsymbol{d} = \{100, 100\}$ m and $w = 30$ (confidence intervals are not shown for clarity).

Heterogeneous scenarios: Fig. 7 depicts the delay performance vs. power expenditure tradeoff achieved by traffic spreading in a scenario where one of the users has lower offered traffic as well as a better average channel to the BS.

The maximal delay performance gain under traffic spreading is up to 27% (greedy) and 18% (log rule), compared to no re-routing. Similarly to the homogeneous scenario, the average re-routing power expenditure reduces rapidly as the weight is increased from 0 while the average file transfer delay increases much slower. For example, up to 95% ($w = 10$) of the maximal performance gain can be achieved at only 40% of the maximal re-routing cost. We see again that queue-aware scheduling is indeed beneficial and the combination of the log rule scheduler and traffic spreading is more effective, especially in energy-sensitive scenarios.

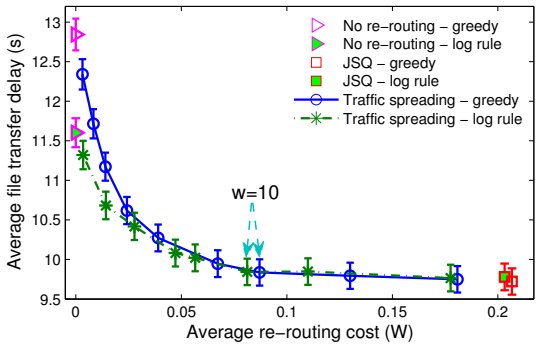


Figure 7: Performance vs. re-routing cost under two-user heterogeneous scenarios where $\boldsymbol{\lambda} = \{0.19, 0.21\}$ arrivals/sec and $\boldsymbol{d} = \{92, 100\}$ m.

Fig. 8 depicts the overall re-routing rate as well as the split between users. Clearly, the performance gain does not originate from simple relaying. In fact, the user with the better average channel and lower traffic (U_1) also forwards traffic to the user with the worse channel (U_2). U_1 does contribute to the bulk of the performance improvement, however we see that U_2 forwards a significant amount of traffic for U_1 as well. For example, 40% of the total re-routed traffic is from U_1 to U_2 in the performance-centric scenario with $w = 0$.

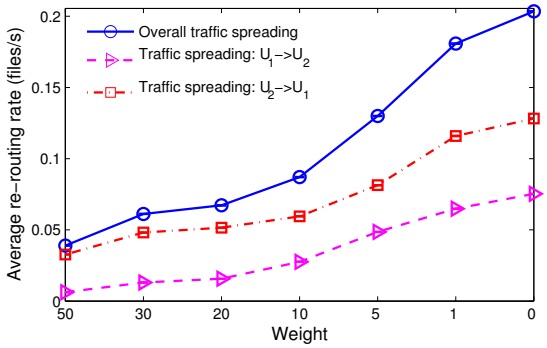


Figure 8: Average re-routing rate of each user under greedy scheduler where $\boldsymbol{\lambda} = \{0.19, 0.21\}$ arrivals/sec and $\boldsymbol{d} = \{92, 100\}$m.

B. Multi-user Scenarios

In this subsection, we present the performance evaluation results of the proposed heuristic under the greedy scheduler.

Dynamic programming vs. heuristic performance: We first consider a three-user homogeneous scenario.We evaluate the performance of the proposed heuristic against that of the optimal dispatching policy obtained from solving the dynamic programming formulation. We observe that the performance of our proposed heuristic is almost as good as the optimal dispatching policy, with both able to achieve near identical performance vs. energy consumption tradeoffs. The average file transfer delay can be reduced by 18% in this scenario. Similar to the cases considered earlier, we see that up to 60% ($w = 30$) of the maximal performance gain can be achieved at only 20% of the maximal re-routing cost. These results demonstrate that the proposed heuristic is indeed successful in multi-user cases.

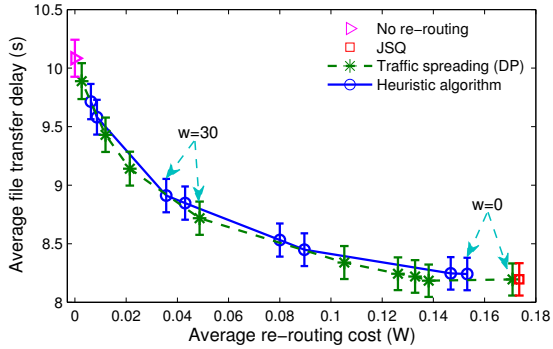


Figure 9: Dynamic programming vs. heuristic performance under a three-user scenario where $\sum_{i=1}^{3} \lambda_i = 0.4$ arrivals/sec, $\lambda_i = \lambda_j$ and $d_i = 100$m, $i, j \in \{1, 2, 3\}$.

Performance scaling with number of users: The scaling of the average file transfer delay with the number of users is shown in Fig. 10 under traffic spreading with different choices of weight, w. Here, all users are at the same distance from the BS and offer the same traffic, and the sum arrival rate across all users is fixed to 0.4 arrivals/sec. As we would expect, the average delay decreases with the increase number of users due to the increase in overall opportunistic gain. We can also observe that traffic spreading is able to improve the delay performance by 17% to 19%, compared to no re-routing. As the number of users increases, we observe that the gain from traffic spreading first increases slightly and then decreases as the user population increases further. This is due to the fact that opportunistic gain grows slower than linearly with the size of the user population. Note that even in multi-user scenarios, traffic spreading does result in significant performance gains. The power expenditure trends with increasing weight are similar to those of earlier scenarios, with power expenditure reducing steeply with small increase in file transfer delays when the weight, w, is increased (energy consumption curves not shown due to space limitations).

Users distributed randomly in a cell of radius 100m: We present simulation results for a scenario where a BS

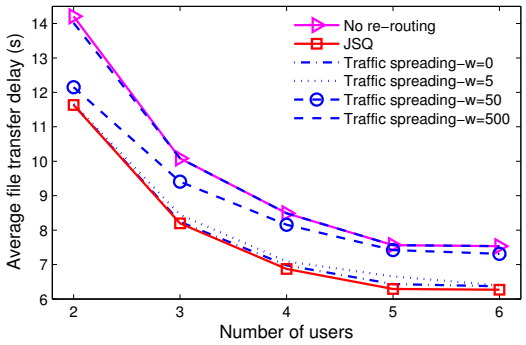


Figure 10: Performance versus N users where $\sum_{i=1}^{N} \lambda_i = 0.4$ arrivals/sec, $\lambda_i = \lambda_j$ and $d_i = 100$m, $i, j \in \{1, 2, ..., N\}$.

serves users in a service area of radius 100m. We consider instances with four users, each of them distributed uniformly at distances ranging from 10 to 100m (corresponding to average capacity to the BS between 16.0407 to 3.0163Mbps). The rate at which users generate requests is also heterogeneous and chosen uniformly in a range of $0.2 \pm 10\%$ arrivals/sec. We evaluate 50 random instances for each choice of weight (i.e. performance-energy tradeoff), and depict the average as well as the 95^{th} and 5^{th} percentile of the file transfer delay in Fig. 11. We observe that the average delay performance gain is up to 50%, which is nearly as good as JSQ. This delay performance can be achieved at a re-routing cost that is around half of that under JSQ. Depending on the user requirements, a different tradeoff between power expenditure and delay performance can be chosen. When we focus on the 95^{th} percentile, the performance gain observed is up to 56%. This demonstrates that traffic spreading is very helpful in improving user performance in instances where overall system performance is poor, which is a very important practical consideration.

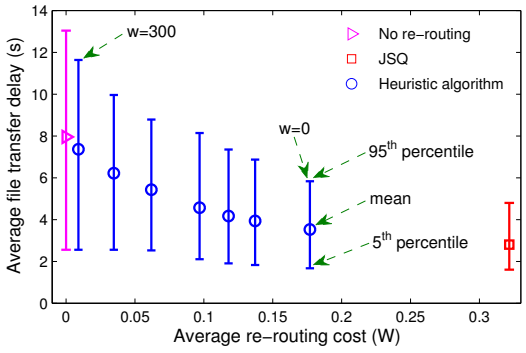


Figure 11: Performance under four-user heterogeneous scenarios where $\lambda_i = 0.2 \pm 10\%, d_i \in [10, 100], i \in \{1, 2, 3, 4\}$.

X. Conclusion and Future Work

In this paper, we presented a user-initiated traffic spreading approach, that is transparent to the BS, to improve the downlink delay performance in small cells. We formulated the problem of choosing the optimal dispatching policy as a Markov decision process and studied its properties in a two-user scenario. We also proposed a heuristic algorithm for multi-user scenarios. Our simulation results showed that the proposed approach can improve the delay performance greatly and the bulk of the performance can be achieved with a small increase in power expenditure. Moreover, even in the future when queue-aware scheduling policies are implemented at the BS, our proposed approach can still complement them and improve user performance. As for future work, we plan to extend our work to larger cells, as well as to consider fairness in power expenditure and delay performance among users.

References

[1] P. Bender, P. Black, M. Grob, R. Padovani, N. Sindhushyana, and S. Viterbi, "CDMA/HDR: a bandwidth efficient high speed wireless data service for nomadic users," *IEEE Communications Magazine*, vol. 38, no. 7, pp. 70–77, Jul. 2000.

[2] X. Liu, E. K. P. Chong, and N. B. Shroff, "A framework for opportunistic scheduling in wireless networks," *Computer Networks*, vol. 41, pp. 451–474, 2003.

[3] S. Borst, "User-level performance of channel-aware scheduling algorithms in wireless data networks," *IEEE/ACM Transaction on Networking*, vol. 13, no. 3, pp. 636–647, Jun. 2005.

[4] M. Andrews, "Instability of the proportional fair scheduling algorithm for HDR," *IEEE Transactions on Wireless Communications*, vol. 3, no. 5, pp. 1422–1426, 2004.

[5] R. Webb, "Femtocells, Small Cells&WiFi," Infonetics Research, 2012.

[6] A. Jalali, R. Padovani, and R. Pankaj, "Data throughput of CDMA-HDR a high efficiency-high data rate personal communication wireless system," in *Proc. IEEE VTC 2000-Spring,*, 2000, pp. 1854–1858.

[7] M. Andrews, K. Kumaran, K. Ramanan, A. Stolyar, and etc., "Scheduling in a queuing system with asynchronously varying service rates," *Probab. Eng. Inf. Sci.*, vol. 18, no. 2, pp. 191–217, Apr. 2004.

[8] S. Shakkottai and A. L. Stolyar, "Scheduling for multiple flows sharing a time-varying channel: The exponential rule," *American Mathematical Society Translations*, vol. 2, 2000.

[9] B. Sadiq, S. J. Baek, and G. De Veciana, "Delay-optimal opportunistic scheduling and approximations: the log rule," *IEEE/ACM Transaction on Networking*, vol. 19, no. 2, pp. 405–418, Apr. 2011.

[10] H. Falaki, D. Lymberopoulos, R. Mahajan, and etc., "A first look at traffic on smartphones," in *Proc. ACM IMC*, 2010, pp. 281–287.

[11] W. Winston, "Optimality of the shortest line discipline," *Journal of Applied Probability*, vol. 14, no. 1, pp. 181–189, 1977.

[12] A. Ephremides, P. Varaiya, and J. Walrand, "A simple dynamic routing problem," *IEEE Transactions on Automatic Control*, vol. 25, no. 4, pp. 690 – 693, Aug. 1980.

[13] M. E. Crovella, M. Harchol-Balter, and C. D. Murta, "Task assignment in a distributed system (extended abstract): improving performance by unbalancing load," in *Proc. ACM SIGMETRICS*, 1998, pp. 268–269.

[14] H.-b. Mor, E. C. Mark, and D. M. Cristina, "On choosing a task assignment policy for a distributed server system," *IEEE Journal of Parallel and Distributed Computing*, vol. 59, pp. 231–242, 1999.

[15] B. Hajek, "Optimal control of two interacting service stations," *IEEE Transactions on Automatic Control*, vol. 29, pp. 491–499, Jun. 1984.

[16] www.visionmobile.com/blog/2007/12/do-we-really-need-femto-cells/.

[17] IEEE Std 802.11n, "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 5: Enhancements for Higher Throughput," pp. 1–565, 2009.

[18] R. Prakash and V. V. Veeravalli, "Centralized wireless data networks with user arrivals and departures," *IEEE Transactions on Information Theory*, vol. 53, no. 2, pp. 695–713, 2007.

[19] D. P. Bertsekas, *Dynamic Programming and Optimal Control*. Athena Scientific, 2005.

[20] Q. Wang and B. Rengarajan, "Recouping opportunistic gain in dense base station layouts through energy-aware user cooperation," Tech. Rep., 2012. [Online]. Available: eprints.networks.imdea.org/365/