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

Qing Wang
Institute IMDEA Networks
Carlos III University of Madrid

***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 is likely to be small resulting in diminished gains from opportunistic scheduling, particularly under dynamic traffic loads. We propose user-initiated traffic spreading, that is transparent to base stations, in order to extract higher opportunistic gain and improve downlink performance. For a specified tradeoff between energy consumption and performance, we characterize the optimal policy by modelling the system as a Markov decision process and also present a tractable heuristic that yields significant performance gains even in multi-user scenarios. Our simulations show that, in the performance-centric case, average delays can be lowered by up to 25% even in homogeneous scenarios where users have identical channel distribution, and up to 51% with heterogeneous users. Further, we show that the bulk of the performance improvement can be achieved with very small increase in energy consumption, e.g., in an energy-sensitive scenario, up to 73% of the performance improvement can typically be achieved at 14% of the energy cost of the performance-centric case.**

***Keywords*-Opportunistic scheduling, energy efficiency, user performance, file transfer delay, Markov decision processes**

I. Introduction

Opportunistic scheduling [1, 2] was proposed for multiuser wireless communication networks to exploit fluctuating channel conditions in order to improve performance. In cellular networks, opportunistic schedulers use knowledge of the channels between the base station (BS) and the users in order to schedule those with favorable channel conditions, thus improving overall throughput. Opportunistic scheduling has been widely adopted in practical cellular systems since Code Division Multiple Access (CDMA) 2000 1xEV-DO [3] and Universal Mobile Telecommunications System (UMTS) High Speed Downlink Packet Access (HSDPA) [4].

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 base station always has data to transmit to each user. However, a more realistic setting is one with a time-varying user population where file requests arrive from users according to a stochastic process. In such a setting, the performance of opportunistic scheduling algorithms can be very different from that observed under static scenarios [5] and can even result in instability [6]. The impact of a time-varying user population is eased when cell sizes are large since the base station 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 wireless capacity, the average number of users served by a base station will decrease and the burstiness will increase. Since opportunistic gain scales as a concave function of the number of users [7] the BS can choose from, presently used scheduling algorithms are prone to losing effectiveness in small cells with dynamic traffic loads.

In this paper, we propose traffic spreading among users as a means to extract higher opportunistic gain in dense networks. We focus on the downlink case which accounts for most of the traffic in a cellular network [8], and on best-effort like traffic such as web browsing or http live streaming where the mobile devices request a file (chunk of content) which is then served to the user by the base station after the delay incurred in fetching it from the Internet. One of the key features of our proposed methodology is that it is completely transparent to the base station. Thus, the improvement in performance can be realized without any modification to existing base stations which can be a difficult, high-cost process. We leverage the multiple radio interfaces (e. g. 3G, WiFi, Bluetooth) available in most smartphones , expected to be the dominant devices used to access future cellular networks, in order to spread traffic among proximate users. Mobile devices are assumed to be capable of connecting simultaneously to the cellular network as well as to other mobile devices using, for example, their WiFi radio interfaces. These mobile-to-mobile links are used to forward both requests and the corresponding files among users, and to increase the available multiuser diversity and thus opportunistic gain.

The proposed methodology certainly incurs additional energy costs due to forwarding requests and files between users. Mobile devices with scarce energy resources necessitate careful power management. Excessive traffic spreading can result in unacceptably high penalties in terms of energy consumption, thus ruling out schemes such as those that require every user to receive the files corresponding to all users. Thus, the degree of spreading has to be carefully

chosen, and the tradeoff between excess energy consumption and performance improvement must be taken into account. In this paper, we develop energy-aware traffic spreading policies that can be calibrated based on the desired tradeoff between performance and energy consumption. We summarize our main contributions as follows:

1) We propose a novel traffic spreading mechanism to increase opportunistic gain through user cooperation.
2) We formulate the problem of determining the optimal traffic spreading policy under a specified tradeoff between energy and performance as a Markov decision problem, and study the properties of the optimal policy in a two user scenario.
3) We show that the optimal traffic spreading policy consists of a set of switching curves in a homogeneous scenario with two users.
4) We propose a tractable heuristic for the multiuser case that uses the two-user problem as a building block.
5) We provide simulation results that demonstrate that file transfer delays can be reduced by up to 51% using the proposed methodology; and that significant gains (up to 73% of the gain) are typically achieved at 14% of the energy cost of the performance-centric case.

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

II. Related Work

Opportunistic gain decreases with the number of users the BS can choose from to serve [7], an effect that is not taken into account under the assumption of a static population with infinite backlog. An approach that has been proposed for systems with dynamic traffic loads is to take into account the backlog of each user during BS scheduling. [9–12] propose BS schedulers that try to maximize opportunistic gain while simultaneously balancing the backlogs of the users in the system. Among these, [9, 10] propose the MaxWeight rule and Exponential rule, respectively, which are guaranteed to be throughput optimal. [11] introduces the Longest Connected Queue (LCQ) policy with the assumption of symmetric arrivals and channel statistics. Here, the authors assume an *on/off* channel model and the policy which serves the user with an on channel who has the largest backlog is shown to be delay-optimal. Recently, the authors in [12] proposed the Log rule with a focus on improving delay performance. Clearly, all the above policies require changes at the BS. Here, we propose an alternate mechanism that improves performance through user cooperation without requiring any changes to the BS, and can also complement the schedulers above in order to extract even higher opportunistic gains.

As we will describe in Sec. IV, we formulate the problem of determining the optimal traffic spreading policy as a dispatching problem. Here, we discuss some of the related work on this topic. A dispatching system typically consists of a dispatcher and several servers. The role of the dispatcher is to route jobs upon arrival to a server based on the chosen dispatching policy. The dispatching problem has received a lot of attention since the landmark work in [13]. The author considers a homogeneous model where arrivals are according to a Poisson process and have exponentially distributed size, and show that when the queue lengths (number of jobs) are known, Join the Shortest Queue (JSQ) minimizes the average waiting time in the queues. When the queue lengths are unavailable, [14] shows that Round Robin is optimal. A number of papers focus on the settings where the number of jobs waiting in each queue is not observed, while the size of each arrival job is available, e.g., authors in [15, 16] work on policies that dispatch packets according to packet size. In contrast to the above papers, our model has only one shared server whose service rate is affected by the dispatching policy. The model in [17] includes the case of a shared server and is the closest to our own. However, this paper like the others make the assumption that the service rate is constant and does not depend on the instantaneous queue state. In our work, the service rate depends on the channel state as well as the queue state, making the problem more complex. The emphasis in all the above papers in on performance, whereas in our case we additionally have to consider the implications of the dispatching decisions on energy consumption which adds a facet that has not been explored before.

While [18] considers an energy-aware dispatching problem, the focus here is on reducing the energy consumption by switching some servers off. The authors assume that different servers run at different service rate and power, and the energy can be saved by turning off the servers that are idle. While the model is similar to ours, with dispatching decisions having energy implications, the results are not applicable to our problem.

III. Traffic Spreading

Consider a network consisting of a BS and several users. Normally, all users send file requests directly to the BS, that fetches the corresponding files from the backbone Internet, and then set them back to the users. However, through user cooperation, one user can ask other users to send the requests to the BS and then receive the corresponding files from those users. We refer to this type of user cooperation through spreading the file requests as *traffic spreading*.

Let us consider the example shown in Fig. 1 where the BS-user link is 3G, and the mobile-to-mobile link is WIFI. 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 backlog corresponding 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 not detect traffic spreading to be beneficial. Thus users send their new requests to the BS directly. In Fig. 1(b), there is much more data in Q_2 than in Q_1, waiting to be served. Under this case, the dispatcher of U_2 would ideally detect that traffic spreading is beneficial since the risk of the BS having no data to send to U_1 in the near term is high. Therefore 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 WIFI connection. In the sequel, we study the problem of determining the optimal traffic spreading policy.

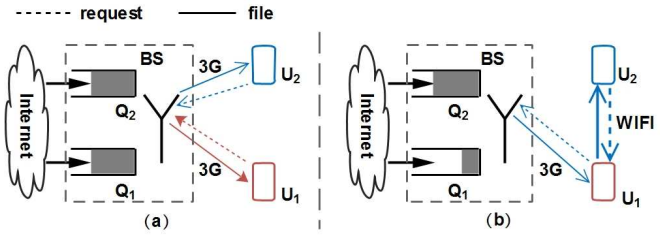


Figure 1: An example of traffic spreading within a small cell: (a) U_1 and U_2 dispatch their requests directly to the BS; (b) U_1 helps U_2 forward its request to the BS.

The traffic spreading policy we propose includes a dispatcher that resides on the mobile device and determines when traffic spreading is beneficial based on the channel statistics of the other users as well as the number of outstanding requests corresponding to each user. Users are assumed to keep track of the number of files they are waiting to receive from the BS, which corresponds to the number of pending requests. The users also measure and share the channel statistics they perceive amongst themselves. i.e., users are aware of the channel statistics corresponding to all users. Note that the dispatcher has no control over the scheduling and cannot predict when the request will be served or the channel conditions at that time. When the dispatcher detects that traffic spreading can be beneficial, it uses the mobile-to-mobile link to forward new requests to the chosen nearby user who forwards it in turn to the BS and is then also the target to receive the file from the BS. Upon receipt (at a future time), the file is forwarded to the user who originated the request using the orthogonal mobile-to-mobile link. The gain in performance derives from maintaining a higher population of users that the BS scheduler can choose from at any given time, thus boosting opportunistic gain.

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 $\mathcal{I} = \{1, 2, ..., N\}$. We assume that all the users are within transmission range of each other which is expected to be the case in femto cell scenarios. The scenarios where some user pairs cannot establish a mobile-to-mobile link is not considered here, and will be studied in the future. The arrival process of requests of user $i \in \mathcal{I}$ is modeled as a Poisson process in time with average arrival rate λ_i, and is assumed to be independent across users. The requested file sizes are exponentially distributed, with mean file size in bits denoted by θ.

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 $\mathcal{S}^i = \{c_1^i, c_2^i, \cdots, c_K^i\}$. We assume that at any time $t \neq t'$, the channel states $S^i(t)$ and $S^i(t')$ are independent for all i. Further, we assume that channels perceived by two different users are also independent. We denote by $p_{c_k}^i, c_k \in \mathcal{S}^i$, the probability that user i perceives channel c_k at any time t. We denote by $\boldsymbol{c}(t) = \{c^i, c^i \in \mathcal{S}^i\}$, the vector channel perceived at time t and by $\boldsymbol{\mathcal{S}}$, the set of possible vector channels. Conditional on the channel state being $c_k \in \mathcal{S}^i$, we define a non-negative value $R_i^{c_k}$, which denotes the achievable instantaneous service rate in bits/second to user i.

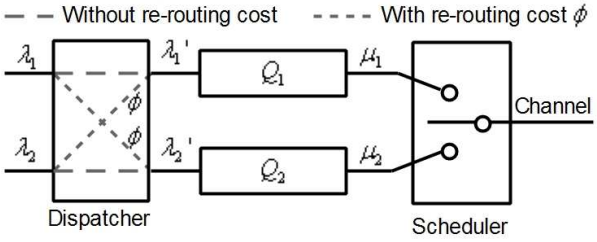


Figure 2: System model

Fig. 2 depicts our system model consisting of three main components, i.e., the BS scheduler, the queues corresponding to each user in 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 i at time t, i.e., the number of unsatisfied requests.

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

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

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 $\mathcal{C}$ of all the possible dispatching policies is defined as

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

The rate at which files arrive to user i's queue at the BS corresponds to the total rate of requests from user i (including forwarded requests from other users) is denoted by $\lambda_i'(\boldsymbol{q}, \boldsymbol{\sigma}(\boldsymbol{q}))$. When user j's dispatcher decides to ask user i to forward that request, there will be additional energy cost since user i has to receive the file from the BS and subsequently forward the file to the originating user j using the mobile-to-mobile (WIFI) link. We model the average additional cost ϕ incurred for a single file as a function of the average file size θ, namely, $\phi(\theta)$, which is obtained through averaging a per-bit cost over the file size distribution.

We model the scheduling policy at the BS through the probabilities $\xi_i(\boldsymbol{q}, \boldsymbol{c})$, denoting the probability that user i is selected by the scheduler, conditional on the queues being in state $\boldsymbol{q}$, and channel vector being $\boldsymbol{c}$. In our performance evaluation, we consider the following 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 lengths of the queues, but has the information whether a queue is empty or not. At any time t, a greedy scheduler randomly chooses a queue i to serve if it satisfies: a) $q_i(t) > 0$; b) The achievable instantaneous service rate to user i is positive at time t.

2) Queue-aware, LCQ scheduling policy: Queue-aware means the scheduler is aware of how many files are waiting to be served at each queue. At any time t, a LCQ scheduler randomly chooses a queue i to serve if it satisfies: a) Queue i has the largest amount of work, where the work means the queue length divided by the service rate of the queue; b) The achievable instantaneous service rate to user i is positive at time t.

The performance metrics we consider are the average file transfer delay and the excess energy consumption induced by the traffic spreading policy. The objective function we seek to minimize is the weighted sum of the average delay and energy cost:

$$\bar{D} + w \cdot \bar{e} \tag{2}$$

where $\bar{D}$ is the average file transfer delay and $\bar{e}$ is the average energy cost under the dispatching policy. Here, w is a weight associated with the energy cost that determines the tradeoff between performance and energy consumption. We ignore the additional time taken in order to forward files over the mobile-to-mobile links since the bandwidth of this link would generally be much higher than the cellular link with distances between users being much shorter than that between the BS and users.

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. 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

$$\mu_i(\boldsymbol{q}) = \sum_{\boldsymbol{c}\in\boldsymbol{\mathcal{S}}} \left(\prod_{i=1}^{N} p_{c^i}\right) \xi_i(\boldsymbol{q}, \boldsymbol{c}) \cdot R_i^{c^i} \tag{3}$$

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, all the n queues see queue-state-dependent service rate. A rigorous justification of the service rate with the consideration of packet or file dynamics can be found in [19]. A similar assumption is made in [12] to characterize the service rate in a continuous time system. On the other hand, 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

$$\lambda_i'(\boldsymbol{q}, \boldsymbol{\sigma}(\boldsymbol{q})) = \sum_{j\in\mathcal{I}} \sigma_j^i(\boldsymbol{q})\lambda_j, \quad i \in \mathcal{I} \tag{4}$$

Under the above assumptions, the objective to minimize the weight sum of average delay and additional energy cost is to find the right $\boldsymbol{\sigma}(\boldsymbol{q}) \in \mathcal{C}$ for each state $\boldsymbol{q}$. Moreover, from Little's law, any optimal policy that achieves the minimum average backlog achieves the minimum average delay as well. Therefore, (2) becomes

$$\frac{|\ E[\boldsymbol{Q}]\ |}{|\ \boldsymbol{\lambda}\ |} + w \cdot \bar{e} \tag{5}$$

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

Under a fix policy $\boldsymbol{\sigma}$, the process $(\boldsymbol{Q}(t), t \ge 0)$ forms a Markov chain on $\mathbb{Z}_+^N$ with state-dependent (depends on both channel and queue states) transition rate. For convenience, we uniformize $\boldsymbol{Q}(t)$ following [20]. For any $\boldsymbol{q} \in \mathbb{Z}_+^N$, we make the following definitions:

$$D_i\boldsymbol{q} \equiv [\boldsymbol{q} - \boldsymbol{e}_i]^+ \tag{6}$$

$$A_i\boldsymbol{q} \equiv \boldsymbol{q} + \boldsymbol{e}_i \tag{7}$$

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

Let $\varphi \ge |\boldsymbol{\lambda}| + \max_{\boldsymbol{q}} |\boldsymbol{\mu}(\boldsymbol{q})|$. Let τ_k denote the time of the kth 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

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

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

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

where $i, j \in \mathcal{I}$.

The average cost (our objective in (5)) under policy $\boldsymbol{\sigma}(\boldsymbol{Q})$ over $[0, \tau_k)$ when starting from an initial queue state $\boldsymbol{q}$ is $\mathbb{E}_{\boldsymbol{q}}^{\boldsymbol{\sigma}(\boldsymbol{Q})} \int_0^{\tau_k} \big[\frac{|\boldsymbol{Q}(t)|}{|\boldsymbol{\lambda}|} + w \cdot \boldsymbol{f}\big(\boldsymbol{Q}(t), \boldsymbol{\sigma}(\boldsymbol{Q}(t))\big)\big] dt$, which, by ignoring the constant multiplier φ^{-1}, is equal to:

$$\begin{aligned} V_k^{\boldsymbol{\sigma}(\boldsymbol{Q})}(\boldsymbol{q}) &\equiv \mathbb{E}_{\boldsymbol{q}}^{\boldsymbol{\sigma}(\boldsymbol{Q})} \left[\sum_{l=0}^{k-1} \boldsymbol{g}(\boldsymbol{Q}_l, \boldsymbol{\sigma}(\boldsymbol{Q}_l))\right] \\ &= \mathbb{E}_{\boldsymbol{q}}^{\boldsymbol{\sigma}(\boldsymbol{Q})} \left[\sum_{l=0}^{k-1} \left(\frac{|\,\boldsymbol{Q}_l\,|}{|\,\boldsymbol{\lambda}\,|} + w \cdot \boldsymbol{f}\left(\boldsymbol{Q}_l, \boldsymbol{\sigma}(\boldsymbol{Q}_l)\right)\right)\right] \end{aligned} \tag{11}$$

where the energy consumption function $\boldsymbol{f}(\cdot)$ is given by

$$\boldsymbol{f}\left(\boldsymbol{Q}_l, \boldsymbol{\sigma}(\boldsymbol{Q}_l)\right) = \sum_{i \neq j} 1 \cdot \sigma_j^i(\boldsymbol{Q}_l) \cdot \phi(\theta),\ j \in \mathcal{I} \tag{12}$$

where 1 denotes the file that arrives to user j during the lth transition time interval. Then the average cost under policy $\boldsymbol{\sigma}(\boldsymbol{Q})$ when starting from state $\boldsymbol{q}$ is as follows:

$$J_{\boldsymbol{q}}^{\boldsymbol{\sigma}(\boldsymbol{Q})} = \lim_{k \to \infty} \sup \frac{1}{k} V_k^{\boldsymbol{\sigma}(\boldsymbol{Q})}(\boldsymbol{q}) \tag{13}$$

The objective function given in (5) seeks to minimizes this average cost. The problem of finding the minimal average cost J^* and the corresponding optimal control $\boldsymbol{\sigma}^*(\boldsymbol{q})$ fits the classical dynamic programming [20]. Thus J^*, which is under all possible dispatching probabilities $\boldsymbol{\sigma}(\boldsymbol{q}) \in \mathcal{C}$, is well-defined, independent of the initial state $\boldsymbol{q}$, and satisfies the Bellman's equation:

$$\begin{aligned} J^* = &\min_{\boldsymbol{\sigma}(\boldsymbol{q}) \in \mathcal{C}} \Big\{ \frac{|\,\boldsymbol{q}\,|}{|\,\boldsymbol{\lambda}\,|} + w \cdot \boldsymbol{f}(\boldsymbol{q}, \boldsymbol{\sigma}(\boldsymbol{q})) \\ &\quad + \mathbb{E}^{\boldsymbol{\sigma}(\boldsymbol{q})}\big\{\big[h\big(\boldsymbol{Q}_{k+1}\big) - h\big(\boldsymbol{Q}_k\big)\big] \mid \boldsymbol{Q}_k = \boldsymbol{q}\big\}\Big\} \\ = &\frac{|\,\boldsymbol{q}\,|}{|\,\boldsymbol{\lambda}\,|} + \sum_{i \in \mathcal{I}} \frac{\mu_i(\boldsymbol{q})}{\varphi}\Big[h\big(D_i\boldsymbol{q}\big) - h\big(\boldsymbol{q}\big)\Big] \\ &\quad + \min_{\boldsymbol{\sigma}(\boldsymbol{q}) \in \mathcal{C}} \sum_{i \in \mathcal{I}} \frac{\lambda_i'(\boldsymbol{q}, \boldsymbol{\sigma}(\boldsymbol{q}))}{\varphi}\big[h\big(A_i\boldsymbol{q}\big) - h\big(\boldsymbol{q}\big)\big] \\ &\quad + w \cdot \min_{\boldsymbol{\sigma}(\boldsymbol{q}) \in \mathcal{C}} \sum_{j \in \mathcal{I}} \sum_{i \neq j} \frac{\lambda_j \sigma_j^i(\boldsymbol{q})}{\varphi} \phi(\theta) \\ = &\frac{|\,\boldsymbol{q}\,|}{|\,\boldsymbol{\lambda}\,|} + \sum_{i \in \mathcal{I}} \frac{\mu_i(\boldsymbol{q})}{\varphi}\Big[h\big(D_i\boldsymbol{q}\big) - h\big(\boldsymbol{q}\big)\Big] + \min_{\boldsymbol{\sigma}(\boldsymbol{q}) \in \mathcal{C}} \sum_{j \in \mathcal{I}} \sum_{i \in \mathcal{I}} \\ &\quad \frac{\lambda_j \sigma_j^i(\boldsymbol{q})}{\varphi}\Big\{w \cdot z_j^i \cdot \phi(\theta) + \big[h\big(A_i\boldsymbol{q}\big) - h\big(\boldsymbol{q}\big)\big]\Big\} \end{aligned} \tag{14}$$

where z_j^i is 1 if $i \neq j$ and 0 if $i = j$, and $h(\boldsymbol{q})$ is a relative function defined by

$$h(\boldsymbol{q}) = J(\boldsymbol{q}) - J(\boldsymbol{q}_s) \tag{15}$$

where $\boldsymbol{q}_s$ is a reference state. Then the optimal dispatching policy $\boldsymbol{\sigma}^*(\boldsymbol{q})$ that minimize (14) can be calculated through methods such as the value iteration or policy iteration from the dynamic programming framework.

VI. Properties of the Optimal Policy

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

A. Restricting the Policy Space

We use value iteration to calculate the numerical results of the optimal dispatching policy $\boldsymbol{\sigma}^*(\boldsymbol{q})$. To simplify the calculation, in this subsection we present a theorem that restricts the optimal policy space as follows:

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

Proof: From Sec. V, we know that a dispatching policy that minimizes (14) is an optimal policy. To minimize (14), it is sufficient to minimize the last part, namely,

$$\min_{\boldsymbol{\sigma}(\boldsymbol{q}) \in \mathcal{C}} \sum_{j \in \mathcal{I}} \sum_{i \in \mathcal{I}} \frac{\lambda_j \sigma_j^i(\boldsymbol{q})}{\varphi}\Big\{w \cdot z_j^i \cdot \phi(\theta) + \big[h\big(A_i\boldsymbol{q}\big) - h\big(\boldsymbol{q}\big)\big]\Big\}$$

Since the dispatching rule used by each user can be chosen independent of the others, we just have to minimize each part within the first sum, namely

$$\min_{\boldsymbol{\sigma}(\boldsymbol{q}) \in \mathcal{C}} \sum_{i \in \mathcal{I}} \sigma_j^i(\boldsymbol{q})\Big\{w \cdot z_j^i \cdot \phi(\theta) + \big[h\big(A_i\boldsymbol{q}\big) - h\big(\boldsymbol{q}\big)\big]\Big\}, j \in \mathcal{I}$$

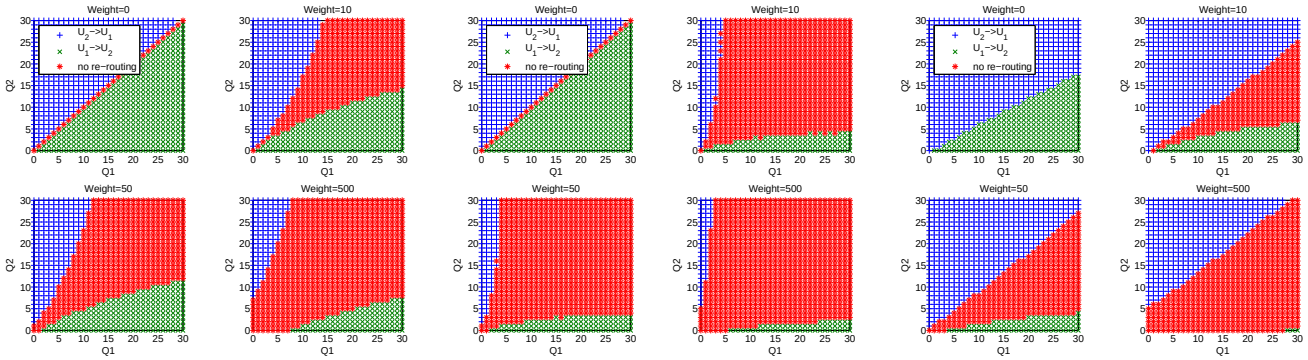
We consider a particular value of j in the proof below. Under queue state $\boldsymbol{q}$, we denote:

$$\alpha_i \equiv \sigma_j^i(\boldsymbol{q}), \quad \beta_i \equiv w \cdot z_j^i \cdot \phi(\theta) + \big[h\big(A_i\boldsymbol{q}\big) - h\big(\boldsymbol{q}\big)\big]$$

Furthermore, let $\boldsymbol{\alpha} = \{\alpha_i, i \in \mathcal{I}\}$ denote a stochastic vector. Recall that each row of matrix $\boldsymbol{\sigma}$ has non-negative elements that sum up to 1, thus $\boldsymbol{\alpha}$ is a stochastic vector. To prove the theorem, we have to show that for a given $\boldsymbol{\beta}$, the minimal value of $\boldsymbol{\alpha} \cdot \boldsymbol{\beta}$ can be achieved when $\alpha_{i^*} = 1$, where $i^* = \arg\min_{i \in \mathcal{I}}\{\beta_i\}$, augmented with a tie-breaking rule. For i^* and $\forall i \in \mathcal{I}$, we have

$$1 \cdot \beta_{i^*} = \alpha_{i^*}\beta_{i^*} + \sum_{i \neq i^*} \alpha_i\beta_{i^*} \leq \alpha_{i^*}\beta_{i^*} + \sum_{i \neq i^*} \alpha_i\beta_i \tag{16}$$

Therefore when the element α_{i^*} of vector $\boldsymbol{\alpha}$ is set to 1 and all the other elements of $\boldsymbol{\alpha}$ are set to 0, the minimal value of $\boldsymbol{\alpha} \cdot \boldsymbol{\beta}$ is achieved, which proves the theorem. ■



(a) (b) (c)

Figure 3: Optimal dispatching policy: (a) A homogeneous scenario with greedy scheduling policy: $p_1 = p_2 = 0.6; \lambda_1 = \lambda_2 = 0.12$; (b) A homogeneous scenario with LCQ scheduling policy: $p_1 = p_2 = 0.6; \lambda_1 = \lambda_2 = 0.12$; (c) A heterogeneous scenario with greedy scheduling policy: $p_1 = 0.8, p_2 = 0.6; \lambda_1 = 0.09, \lambda_2 = 0.15$

B. A Two-user System

Due to the high complexity of solving the dynamic program (14) for multi-user scenarios, in this subsection we focus on the two-user scenario. Currently we only consider the *on/off* channel, where we denote p_i as the probability that the channel is *on* for user i. When the channel is *on* for user i, the user i can be served at a constant positive service rate; while the instantaneous service rate of user i is 0 if the channel is *off* for it.

From Theorem 1, we know that under the two-user model and at any queue state $\boldsymbol{q}$, there are only 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 section, we refer to these controls as *no re-routing*, $U_2 \to U_1$ and $U_1 \to U_2$, respectively.

The optimal dispatching policy evaluated numerical under a homogeneous case with greedy scheduling policy, a homogeneous case with LCQ scheduling policy and a heterogeneous case with greedy scheduling policy are shown in Fig. 3. From this figure, we observe the following properties of the optimal policy:

Existence of switching curves: As Fig. 3 clearly shows, the optimal policy in all the cases considered is a set of switching curves. Here, switching curves refer to the boundaries between decision regions where the same control is used in every state belonging to the region. Thus, in this case, we observe that for any fixed value of q_1, there exist threshold values q_2^a and q_2^b such that

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

For the two-user homogeneous scenario with the LCQ scheduling policy, we show the existence of switching-curves of the optimal policy in the following theorem.

Theorem 2: There exists an optimal dispatching policy that has switching curves, under the two-user homogeneous scenarios with LCQ scheduling policy.

Proof: The proof is presented in Appendix-A. ■

Impact of the weight of re-routing cost: In the top-left sub-figures of Fig. 3(a) and Fig. 3(b) where the weight of re-routing cost is 0, the switch curves lie along the line where both of the queue lengths are equal, and the corresponding optimal policy is just the JSQ presented in [13]. We see from the left-top sub-figure of Fig. 3(c) that there is no control *no re-routing*. This is because the optimal policy that minimizes the average delay is JSQ (note that under heterogeneous case, JSQ means joining the queue with the least work left), thus unless carefully choosing the parameters, the queue lengths are always not integers when *no re-routing* is the optimal control. With the increase of weight, the re-routing areas ($U_1 \to U_2$ and $U_2 \to U_1$) decrease. This is because when the weight increases, the portion of energy consumption in the objective function (5) also increases; while the increased performance gain from re-routing is not as much as the increased energy cost.

Impact of the queue length: Another interesting observation from Fig. 3 is that the switching curves are concave. As the queue length values are scaled up, the level of imbalance between the queues that is required before re-routing is used also increases. The intuition behind this is when both the queues are very long, the probability that one of the queues will become empty in the near future is very low. There is also a high degree of uncertainty and the shorter queue might yet see many arrivals without any re-routing required. Thus there is no need to dispatch requests to other users to balance the queues at the cost of energy.

Impact of the scheduling policies: From Fig. 3(a) and Fig. 3(b), we can observe that compared to the scenario with greedy scheduling policy, the re-routing areas under

LCQ scheduling policy are much smaller when weight of the re-routing cost is the same. This is because with the LCQ scheduling policy, the scheduler already tries to balance the queues to keep as many non-empty queues in the future as possible, making the dispatcher not re-route as much as it does under the greedy scheduling policy.

Impact of the arrival rate: The boundaries between different decision regions of the optimal dispatching policy under different request arrival rates is shown in Fig. 4. The scheduling policy here is greedy. We can observe that with the increase of arrival rate, the re-routing areas decrease. One reason for this could be that at lower arrival rates, opportunities to use re-routing are also naturally lower. Thus, the optimal dispatcher is more aggressive at re-routing requests even when queue lengths are larger since the shorter queue could be emptied before the next arrival to either user.

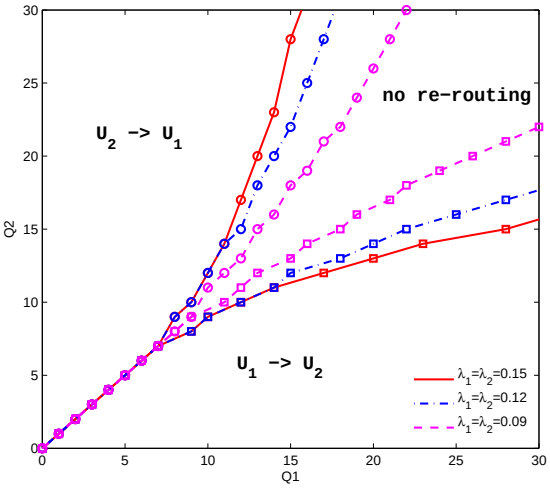


Figure 4: The boundaries between different decision regions of the optimal dispatching policy under different traffic load where $p_1 = p_2 = 0.6$; weight $= 1$

VII. A Heuristic Algorithm for Multi-user System

In the network with more than two users, using dynamic programming technique to obtain the optimal dispatching policy (14) becomes very difficult because of the computational complexity. Thus we propose a heuristic algorithm to solve the multi-user scenario, as shown in Algorithm 1, where the algorithm uses the dynamic programming of the two-user model as a building block. We denote $dp(\tilde{\boldsymbol{\lambda}}, \tilde{\boldsymbol{\mu}})$ as the optimal policy for the two-user dispatching problem, where $\tilde{\boldsymbol{\lambda}} = \{\tilde{\lambda}^1, \tilde{\lambda}^2\}$ is the vector of arrival rates and $\tilde{\boldsymbol{\mu}}(\tilde{\boldsymbol{q}}) = \{\tilde{\boldsymbol{\mu}}^1(\tilde{\boldsymbol{q}}), \tilde{\boldsymbol{\mu}}^2(\tilde{\boldsymbol{q}})\}, \tilde{\boldsymbol{q}} \in \mathbb{Z}_+^2$ specifies service rate at all queue states. We define a family of functions $g_j : \mathbb{Z}_+^2 \to \mathbb{Z}_+^N$, for $j = \{1, 2, ..., N\}$, so that if $\boldsymbol{q} = g_j(\tilde{\boldsymbol{q}})$, then $q_j = \tilde{q}^2$ and $q_l = \tilde{q}^1, l \neq j, l = \{1, 2, ..., N\}$.

Algorithm 1 A Heuristic Algorithm for N-user System

// The algorithm runs at the dispatcher of user $i \in \mathcal{I}$ where a new request is generated.
Input: $\boldsymbol{\lambda}$ **:** a $1 \times N$ vector of the arrival rates
$\boldsymbol{\mu} : \boldsymbol{\mu}(\boldsymbol{q}), \boldsymbol{q} \in \mathbb{Z}_+^N$, $1 \times N$ vector of the service rates at queue state $\boldsymbol{q}$
$\boldsymbol{q}$: a $1 \times N$ vector of the queue lengths at the time when a new request is generated at user i
Output: dispatching decision at user i
$j \leftarrow \arg\min_{l \in \mathcal{I}} \{q_l / \mu_l(\boldsymbol{e}_l)$, where $\boldsymbol{e}_l$ is a $1 \times N$ zero-valued vector except the lth element is 1.
if $j \neq i$ **then**
  $\mathcal{Y}_S \leftarrow \{j\}$
  $\mathcal{Y}_B \leftarrow \mathcal{I} \setminus \mathcal{Y}_S$
  while $\mathcal{Y}_B \neq \emptyset$ **do**
    $\tilde{\lambda}^1 \leftarrow \sum_{l \in \mathcal{Y}_B} \lambda_l$
    $\tilde{\lambda}^2 \leftarrow \sum_{l \in \mathcal{Y}_S} \lambda_l$
    $\tilde{\boldsymbol{\mu}}^1(\tilde{\boldsymbol{q}}) = \mu^j(g_j(\tilde{\boldsymbol{q}}))$
    $\tilde{\boldsymbol{\mu}}^2(\tilde{\boldsymbol{q}}) = \sum_{l \neq j} \mu^l(g_j(\tilde{\boldsymbol{q}}))$
    $\boldsymbol{\sigma} \leftarrow dp(\tilde{\boldsymbol{\lambda}}, \tilde{\boldsymbol{\mu}})$
    if $\boldsymbol{\sigma}(\sum_{l \in \mathcal{Y}_B} q_l, q_j) = U_1 \to U_2$ **then**
      $i^* \leftarrow \arg\max_{l \in \mathcal{I}} \{q_l / \mu_l(\boldsymbol{e}_l)\}$
      if $i^* = i$ **then**
        return dispatch the new request to user j
      else
        $\mathcal{Y}_B \leftarrow \mathcal{Y}_B \setminus i^*$
        $\mathcal{Y}_S \leftarrow \mathcal{Y}_S \cup i^*$
      end if
    else
      return send the new request directly to the BS
    end if
  end while
else
  return send the new request directly to the BS
end if

The intuition behind the heuristic algorithm is that users are more aggressive to send requests to the user with the least amount of work left at the BS. To use the dynamic programming of the two-user model, we put the user with the least amount of work into set $\mathcal{Y}_S$ and all the other users into set $\mathcal{Y}_B$. We combine all users in $\mathcal{Y}_B$ as a new user, of which the queue length, arrival rate and service rate is the sum of the queue lengths, arrival rates and service rates of the individual users in $\mathcal{Y}_B$, respectively. The new user and the user with least amount of work form a two-user model.

If the optimal policy $\boldsymbol{\sigma}$ is to send requests from the new user to the user with least amount of work at current queue state, the request of the user with the largest amount of work in $\mathcal{Y}_B$ is sent to the user in $\mathcal{Y}_S$ with the least amount of work. The intuition here is that users with the largest amount of work is more aggressive to send request to other users,

which is presented in Sec. VI. The algorithm continues to decide whether or not to send the request of the user with the second largest amount of work to the user with least amount of work in $\mathcal{Y}_S$, by excluding the user with largest amount of work from $\mathcal{Y}_B$, and adding the arrival rate of that user to the user with least amount of work in $\mathcal{Y}_S$.

However, if the optimal policy $\boldsymbol{\sigma}$ is not to send requests from the new user to the user with least amount of work, the algorithm stops with sending the requests of each user directly to the BS.

VIII. Performance Evaluation

In this section, we evaluate our proposed traffic spreading policy through simulation. We demonstrate the tradeoff between performance improvement and additional energy consumption, based on the numerical results presented in Sec. VI. The file size is exponentially distributed, and the mean size θ is 8Mbits, under which the re-routing energy cost $\phi(\theta)$ is assumed to be 1 Joule. The instantaneous service rate R_{on} of the channel is set to 3Mbits/s and $R_{off} = 0$. The maximum number of files each user requests is 10000, and we stop the simulation when the last file of any of the users reaches at the BS. The average file transmission delay and average re-routing cost are estimated at a confidential interval of 95%. To evaluate our policy, we first introduce two dispatching policies that we use as baselines.

1) No routing: : Under the no routing dispatching policy, at any time t when a request of user i is generated, user i sends it directly to the BS. Thus there is no additional energy consumption under this dispatching policy.

2) Join the Shortest Queue (JSQ): : Under this dispatching policy, at any time t when there is a new request generated by user i, it makes the decision whether to send the request directly to the BS or to an other user based on the current queue state $\boldsymbol{q}$. It chooses to send the request to user j that has the least amount of work left with a randomly choosing tie-breaking rule. If $i \neq j$, then additional energy consumption occurs.

Next we present the simulation results for the two-user model and multi-user model, respectively.

A. The Two-user Scenario

1) Greedy Scheduling Policy:

Homogeneous scenario: The simulation results under a homogeneous scenario are shown in Fig. 5(a). On the left y-axis, we plot the average file delay, and on the right y-axis, the average re-routing cost (additional energy consumption because of the forwarding of files among users). Firstly, we focus on the curves that represent the average file delay. Among the three policies, JSQ is the optimal policy that minimizes the average delay, which has been shown in [13]. However, under performance-centric case (where the weight $w = 0$), our proposed traffic spreading policy can reduce the average file delay as much as JSQ policy does. Both JSQ and the traffic spreading polices have delay performance improvement up to 21%, using no routing policy as the baseline. Under energy-sensitive cases (where $w > 0$), as we would expect, increasing the weight of re-routing cost results in the increase of average file delay. The reason behind this is that under the traffic spreading policy, changing the weight is the method to tradeoff the delay performance and additional energy consumption. A higher weight indicates that users face more serious energy problem, therefore they prefer to save energy other than improve the delay performance.

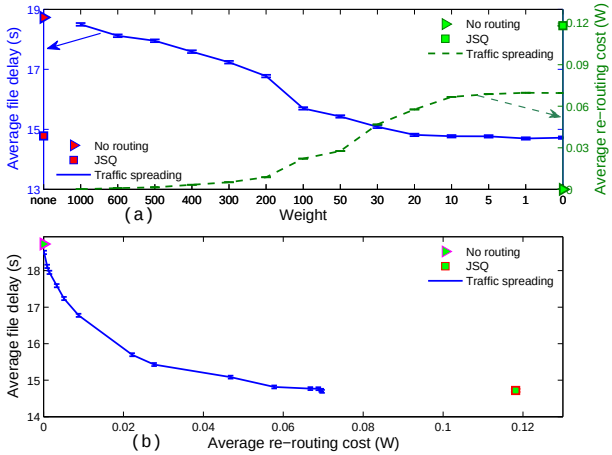


Figure 5: Performance under a homogeneous scenario where $p_1 = p_2 = 0.6; \lambda_1 = \lambda_2 = 0.12$. (a) Average file delay and re-routing cost versus the weight of re-routing cost; (b) Average file delay versus average re-routing cost.

Secondly, we focus on the curves representing the average re-routing cost. As shown in Fig. 5(a), the average re-routing cost is around $0.12W$ under JSQ policy. This indicates that the average re-routing rate is 0.12 files/s, which can be expected under the homogeneous scenario. However, under our proposed traffic spreading policy, the maximal re-routing cost is only around $0.07W$. Therefore the traffic spreading policy has a very important property – it can maintain the performance improvement at the maximal level, as well as save nearly half of the additional energy consumption at the same time when compared to JSQ. Furthermore, as the weight increases, the average re-routing rate decreases greatly.

The tradeoff between the performance improvement and the additional energy consumption under the traffic spreading policy can be seen clearly from Fig. 5(b). A very interesting observation indicated by this figure is that most of the performance gain can be achieved with not too much energy consumption. For example, when $w = 0$, the maximal performance gain is around 21%, and the maximal average re-routing cost is around $0.07W$. However, when $w = 50$, the re-routing cost is only about $0.025W$. This means under the traffic spreading policy, 85% of the

maximal performance gain can be achieved with only 35% of the maximal average re-routing cost.

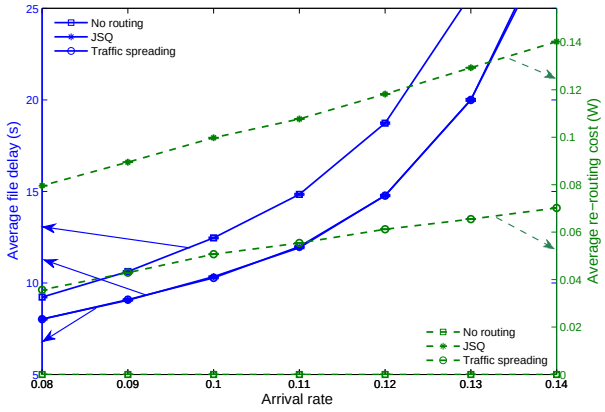


Figure 6: Performance under homogeneous scenarios with different traffic load where $p_1 = p_2 = 0.6$ and weight=10.

The impact of traffic load on the delay performance and additional energy consumption is shown in Fig. 6. In this figure we simulate different traffic load by changing the arrival rate $\boldsymbol{\lambda}$, while keeping $\boldsymbol{p}$, $\boldsymbol{\theta}$ and w be constant. From the curves that represent average file delay, we can observe the delays under JSQ and traffic spreading are always the same. This indicates again that the traffic spreading policy has similar delay performance as JSQ under small weights. With the increase of $\boldsymbol{\lambda}$, both the delays under three policies increase, and the performance gain of traffic spreading also increases, e.g., when $\boldsymbol{\lambda} = \{0.1, 0.1\}$, the performance gain is about 18%; while the performance gain increases to 25% when $\boldsymbol{\lambda} = \{0.14, 0.14\}$. As for the curves representing average re-routing cost, we have several observations. Firstly, the curve under JSQ is linear, and always equals to half of the total arrival rate of all users. The reason behind this is whenever a request is generated by user i, user i will ask user j ($i \neq j$) to send it to the BS with probability 0.5, under the two-user homogeneous scenario. Secondly, the average re-routing cost under traffic spreading policy is observed to be only half of that under JSQ. This indicates our proposed policy can save much energy under different traffic load settings, which is very important in cellar networks where most of the users face limited energy problem.

Heterogeneous scenario: The simulation results under a heterogeneous scenario are shown in Fig. 7 (a). We also plot the average file delay and re-routing cost on the left and right y-axis, respectively. The delay under traffic spreading decreases with the increase of weight, and when $w = 0$ the performance gain is up to 51% compared to the no routing policy. As for the re-routing energy cost, it reduces greatly even if we increase the weight a little, while at the same time the delay performance keeps at very high level. For example, up to 99% ($w = 5$) of the maximal performance gain ($w = 0$) can be achieved at only 50% of the maximal energy consumption. This shows that under this heterogeneous scenario, the traffic spreading policy can save half of the energy while almost maintains the maximal delay performance unchanged (which can be seen clearly from Fig. 7 (b)).

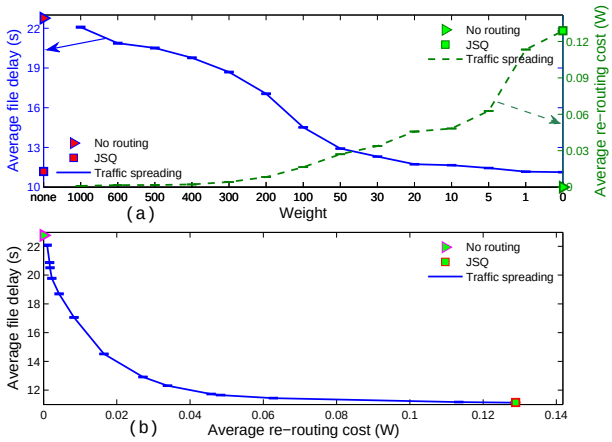


Figure 7: Performance under a heterogeneous scenario where $p_1 = 0.8, p_2 = 0.6; \lambda_1 = 0.1, \lambda_2 = 0.15$.

Another important observation under the heterogeneous scenario is that the performance gain comes from all users, namely, the user (U_2) with lower channel *on* probability also forwards files to the user (U_1) with higher channel *on* probability. This is indicated by Fig. 8, where we can see that U_2 re-routes 16% of the total re-routed files for user U_1. This value decreases with the increase of weight. Certainly, U_1 contributes to the bulk of the performance improvement.

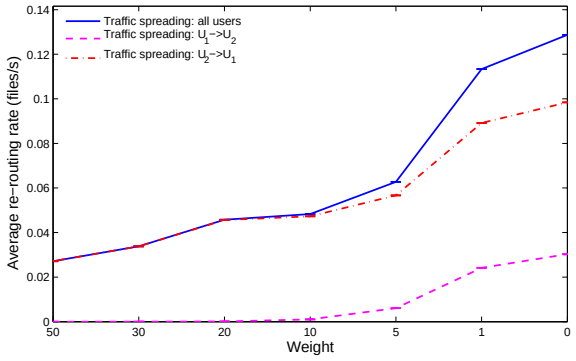


Figure 8: Average re-routing rate of per user under the settings where $p_1 = 0.8, p_2 = 0.6$ and $\lambda_1 = 0.1, \lambda_2 = 0.15$.

The impact of different traffic load on the performance are shown in Fig. 9, where $w = 10$. We vary the traffic load by changing λ_1 and fixing $\lambda_2 = \lambda_1 + 0.05$, while keeping $\boldsymbol{p}$ and $\boldsymbol{\theta}$ unchanged. As indicated from the curves that represent the

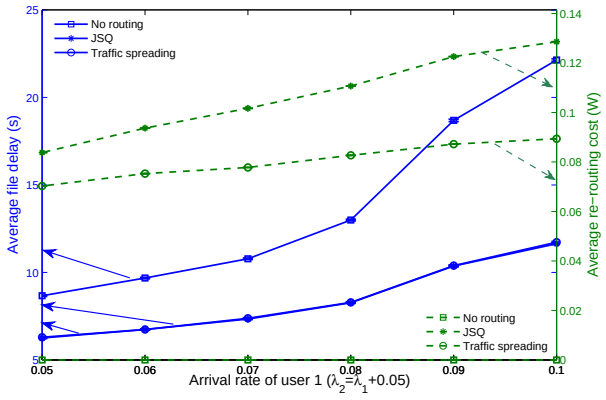


Figure 9: Performance under heterogeneous scenarios with different traffic load where $p_1 = 0.8, p_2 = 0.6$ and weight=10.

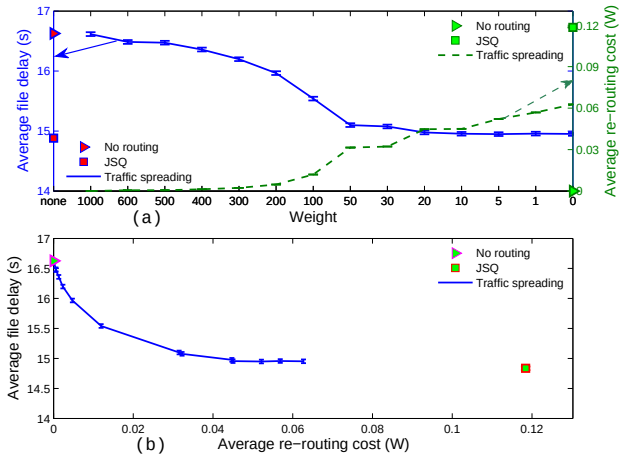


Figure 10: Performance under a homogeneous scenario where $p_1 = p_2 = 0.6; \lambda_1 = \lambda_2 = 0.12$.

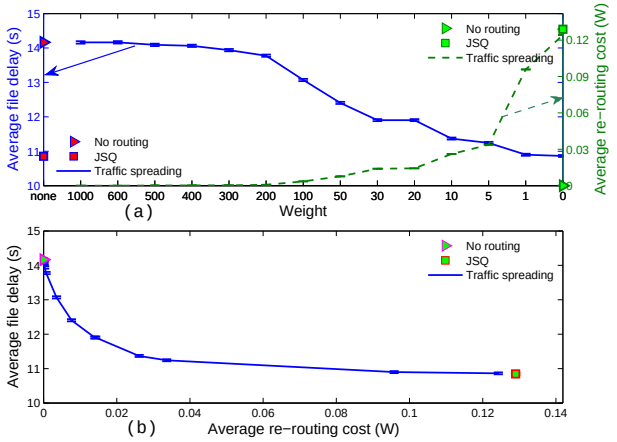


Figure 11: Performance under a heterogeneous scenario where $p_1 = 0.8, p_2 = 0.6; \lambda_1 = 0.1, \lambda_2 = 0.15$.

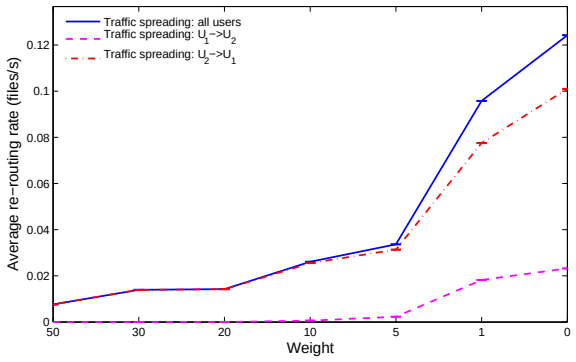


Figure 12: Average re-routing rate of per user under the settings where $p_1 = 0.8, p_2 = 0.6$ and $\lambda_1 = 0.1, \lambda_2 = 0.15$.

delay, the traffic spreading policy reduces the delays as much as JSQ does, same as that under homogeneous scenarios. Compared to no routing, the delay performance gain under traffic spreading increases with the increase of $\boldsymbol{\lambda}$, e.g., when $\boldsymbol{\lambda} = \{0.06, 0.11\}$, the performance gain is around 40%, while the gain increases to 50% when $\boldsymbol{\lambda} = \{0.1, 0.15\}$, both are much higher than that under homogeneous scenarios. As for the average re-routing cost, the curves are both linear, and the one under the traffic spreading policy is lower than that under JSQ, but not as much lower as that under homogeneous scenarios. Again, this also indicates that the traffic spreading policy can save much energy under heterogeneous scenarios, at the same time maintains the performance improvement at a high level.

2) LCQ scheduling policy:

Homogeneous scenario: The simulation results under a homogeneous scenarios are shown in Fig. 10 (a). From the curves representing average file delay, we observe that the delay under no routing & LCQ scheduling policies is around 16.8s, which is lower than 19s (Fig. 5(a)) that under no routing & greedy scheduling policies. This is the gain from the "smart" LCQ scheduling policy. However, even under this scheduling policy, the traffic spreading policy can still reduce the delay by 11% as much as JSQ does, and at the same time the average re-routing cost is only $0.06W$, half of that under JSQ ($0.12W$). Moreover, as indicated in Fig. 10 (b), up to 90% of the maximal gain can also be achieved at only 50% of the maximal additional energy consumption under the LCQ scheduling policy.

Heterogeneous scenario: The simulation results under a heterogeneous scenarios are shown in Fig. 11. Compared to no routing policy, the maximal delay performance gain under traffic spreading is 21%; and the maximal average re-routing cost is $0.12W$, which is the same as that under JSQ. When $w = 5$, we observe that it can achieves 90% of the maximal performance gain, at only 25% of the maximal average re-routing cost. Another observation under heterogeneous scenario is that U_2 also help U_1 forward the requests, as

shown in Fig. 12. we can see that U_2 re-routes up to 16% of the total re-routed files for U_1, and this value decreases with the increase of weight.

Therefore, in the future even when "smart" scheduling policies are implemented at the BS, the traffic spreading policy can still improve the downlink delay performance.

B. The Multi-user Scenario

In the following part we simulate the proposed heuristic algorithm for a three-user model. The performance under a homogeneous scenario for the three-user model is shown in Fig. 13(a). We plot the average delay and re-routing cost on the left and right y-axis, respectively. We first focus on the curves that represent average delay. One observation is that the maximal performance gain from the heuristic algorithm are not as much as that under the JSQ. This indicates our algorithm still have room to be optimized. Anyway, the proposed algorithm can reduce the average delay by 27% under this homogeneous settings. As we would expect, the performance decreases with the increase of weight. Secondly, from that curves that represent average re-routing cost, we can observe the maximal re-routing cost is $0.085W$ under the heuristic algorithm, which is only 42% of that under JSQ.

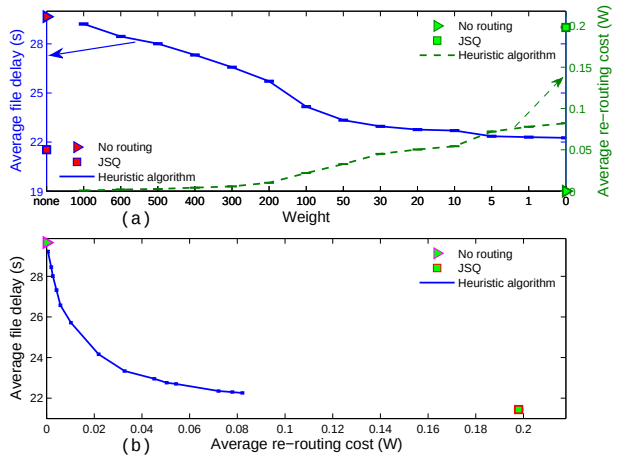


Figure 13: Performance under a homogeneous scenario (three users) where: $p_1 = p_2 = p_3 = 0.6; \lambda_1 = \lambda_2 = \lambda_3 = 0.1$.

IX. Conclusion and Future Work

A. Conclusion

In this paper, we presented a user-initiated traffic spreading approach to improve the downlink delay performance in small cells. By taking into account the additional energy consumed by the forwarding of files among users, we formulated the problem of choosing the optimal dispatching policy as a Markov decision process and studied the properties of the optimal policy 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 very small increase in energy consumption. Moreover, even in the future when "smart" scheduling policies are implemented at the BS, our proposed approach can still work well.

B. Future Work

As for the future work, we plan to extend our work from the following aspects:

1) Change the current *on/off* channel used in the numerical calculation and simulation parts to Rayleigh fading channel;
2) Find other methods to prove the existence of switching curves under the optimal policy for more general settings;
3) In this paper we assume all users within the BS can communicate with each other. In the future we will consider scenarios where some users are not in the transmission range of other users.
4) Currently, the energy cost of forwarding files only depends on the average file size. In the future we will consider energy cost depends both on the average file size, users and the distances between users.

Acknowledgment

This thesis is finished under the supervision of Dr. Balaji Rengarajan from Institute IMDEA Networks.

Appendix

A. Proof of Theorem 2

Before presenting the proof of Theorem 2, we introduce the value iteration and some notations. The value iteration is widely used to solve the Bellman equation. The most used version of the value iteration method for the average cost problem is to select an initial state and generate successively the corresponding optimal k-stage cost $J_k(\boldsymbol{q})$. As shown in [20], the ratios $J_k(\boldsymbol{q})/k$ converges to the optimal average cost per stage J^* as $k \to \infty$. Therefore, we can use induction upon the value iteration method, as the author does in [20], to prove our theorem. We first define the relative value $h_k(\boldsymbol{q})$ at stage k given by

$$h_k(\boldsymbol{q}) = J_k(\boldsymbol{q}) - J_k(\boldsymbol{q}_s) \tag{17}$$

where $\boldsymbol{q}_s$ is a reference state. Moreover, we define $\Delta_k(\boldsymbol{q})$ as follows:

$$\begin{aligned}\Delta_k(\boldsymbol{q}) =& h_k(\boldsymbol{q}+\boldsymbol{e}_1) - h_k(\boldsymbol{q}+\boldsymbol{e}_2)\\ =& J_k(\boldsymbol{q}+\boldsymbol{e}_1) - J_k(\boldsymbol{q}_s) - (J_k(\boldsymbol{q}+\boldsymbol{e}_2) - J_k(\boldsymbol{q}_s))\\ =& J_k(\boldsymbol{q}+\boldsymbol{e}_1) - J_k(\boldsymbol{q}+\boldsymbol{e}_2)\end{aligned} \tag{18}$$

To prove Theorem 2, we have to derive the service rate $\boldsymbol{\mu}(\boldsymbol{q})$ under the LCQ scheduling policy. From Sec. V-A, we can easily get the expression of $\boldsymbol{\mu}(\boldsymbol{q})$:

$$\mu_i(\boldsymbol{q}) = \begin{cases} p_i, & q_i < q_j \\ (p_i + p_j - p_i p_j)/2, & q_i = q_j \\ p_i(1-p_j), & q_i > q_j \end{cases} \tag{19}$$

where $i, j \in \{1, 2\}$ and $i \neq j$. Then we make the following notations:

$$\mu_1' \equiv p_1, \quad \mu_1'' \equiv p_1(1-p_2) \tag{20}$$

$$\mu_2' \equiv p_2(1-p_1), \quad \mu_2'' \equiv p_2 \tag{21}$$

According to (19)-(21), we know

$$\mu_1' \geq \mu_1'', \quad \mu_2' \leq \mu_2'', \quad \mu_1' + \mu_2' = \mu_1'' + \mu_2'' \tag{22}$$

And when $p_1 = p_2$, we have

$$\mu_1' = \mu_2'', \quad \mu_1'' = \mu_2' \tag{23}$$

The uniform version of the continuous problem is present in section V, with uniform rate φ and transition probabilities shown in (8)-(10). Then Bellman's equation takes the form

$$\begin{aligned} \varphi J_{k+1}(\boldsymbol{q}) = \varphi \frac{|\,\boldsymbol{q}\,|}{|\,\boldsymbol{\lambda}\,|} + \sum_{i\in\mathcal{I}} \mu_i(\boldsymbol{q}) J_k\big([D_i \boldsymbol{q}]^+\big) \\ + \min_{\boldsymbol{\sigma}(\boldsymbol{q})\in\mathcal{C}} \sum_{j\in\mathcal{I}} \sum_{i\in\mathcal{I}} \lambda_j \sigma_j^i(\boldsymbol{q}) \big[w \cdot z_j^i \cdot \phi + J_k\big(A_i \boldsymbol{q}\big)\big] \end{aligned} \tag{24}$$

We present two lemmas supporting the proof of Theorem 2.

Lemma 1: In the two-user model, the k-stage cost $J_k(\boldsymbol{q})$ always equals $J_k(\tilde{\boldsymbol{q}})$ under the homogeneous scenarios when the scheduling policy is LCQ, where $\tilde{\boldsymbol{q}}$ is defined as $\tilde{\boldsymbol{q}} \equiv \{q_2, q_1\}$ if $\boldsymbol{q} = \{q_1, q_2\}, q_1, q_2 \in \mathbb{Z}_+$.

Proof: We prove this lemma by induction.

Induction basis: Since $J_0(\boldsymbol{q}) = J_0(\tilde{\boldsymbol{q}}) = 0$ for any $\boldsymbol{q} \in \mathbb{Z}^2$, we get $J_0(\boldsymbol{q}) = J_0(\tilde{\boldsymbol{q}})$.

Induction step: Assume $J_k(\boldsymbol{q}) = J_k(\tilde{\boldsymbol{q}})$ for any $\boldsymbol{q} \in \mathbb{Z}^2$, then from (24) we have

$$\begin{aligned} \varphi J_{k+1}(\boldsymbol{q}) = \mu_1(\boldsymbol{q}) J_k([\boldsymbol{q} - \boldsymbol{e}_1]^+) + \mu_2(\boldsymbol{q}) J_k([\boldsymbol{q} - \boldsymbol{e}_2]^+) \\ + \varphi \frac{|\,\boldsymbol{q}\,|}{|\,\boldsymbol{\lambda}\,|} + \lambda_1 \min\big[J_k(\boldsymbol{q} + \boldsymbol{e}_1), w\phi + J_k(\boldsymbol{q} + \boldsymbol{e}_2)\big] \\ + \lambda_2 \min\big[w\phi + J_k(\boldsymbol{q} + \boldsymbol{e}_1), J_k(\boldsymbol{q} + \boldsymbol{e}_2)\big] \end{aligned} \tag{25}$$

and

$$\begin{aligned} \varphi J_{k+1}(\tilde{\boldsymbol{q}}) = \mu_1(\tilde{\boldsymbol{q}}) J_k([\tilde{\boldsymbol{q}} - \boldsymbol{e}_1]^+) + \mu_2(\tilde{\boldsymbol{q}}) J_k([\tilde{\boldsymbol{q}} - \boldsymbol{e}_2]^+) \\ + \varphi \frac{|\,\tilde{\boldsymbol{q}}\,|}{|\,\boldsymbol{\lambda}\,|} + \lambda_1 \min\big[J_k(\tilde{\boldsymbol{q}} + \boldsymbol{e}_1), w\phi + J_k(\tilde{\boldsymbol{q}} + \boldsymbol{e}_2)\big] \\ + \lambda_2 \min\big[w\phi + J_k(\tilde{\boldsymbol{q}} + \boldsymbol{e}_1), J_k(\tilde{\boldsymbol{q}} + \boldsymbol{e}_2)\big] \end{aligned} \tag{26}$$

We compare the right parts of (25) and (26):

1) From the definition of $\tilde{\boldsymbol{q}}$, we have $\varphi\frac{|\boldsymbol{q}|}{|\boldsymbol{\lambda}|} = \varphi\frac{|\tilde{\boldsymbol{q}}|}{|\boldsymbol{\lambda}|}$.
2) According to (19)-(21), we have $\mu_1(\boldsymbol{q}) = \mu_2(\tilde{\boldsymbol{q}})$ and $\mu_2(\boldsymbol{q}) = \mu_1(\tilde{\boldsymbol{q}})$. Furthermore, for the symmetric relationship between $\boldsymbol{q}$ and $\tilde{\boldsymbol{q}}$, we have $J_k([\boldsymbol{q} - \boldsymbol{e}_2]^+) = J_k([\tilde{\boldsymbol{q}} - \boldsymbol{e}_1]^+)$ and $J_k([\boldsymbol{q} - \boldsymbol{e}_1]^+) = J_k([\tilde{\boldsymbol{q}} - \boldsymbol{e}_1]^+)$ from the induction. Thus
$$\begin{aligned} &\mu_1(\boldsymbol{q}) J_k([\boldsymbol{q} - \boldsymbol{e}_1]^+) + \mu_2(\boldsymbol{q}) J_k([\boldsymbol{q} - \boldsymbol{e}_2]^+) \\ =&\mu_1(\tilde{\boldsymbol{q}}) J_k([\tilde{\boldsymbol{q}} - \boldsymbol{e}_1]^+) + \mu_2(\tilde{\boldsymbol{q}}) J_k([\tilde{\boldsymbol{q}} - \boldsymbol{e}_2]^+) \end{aligned}$$
3) Similarly, we have $J_k(\boldsymbol{q} + \boldsymbol{e}_1) = J_k(\tilde{\boldsymbol{q}} + \boldsymbol{e}_2)$ and $J_k(\boldsymbol{q} + \boldsymbol{e}_2) = J_k(\tilde{\boldsymbol{q}} + \boldsymbol{e}_1)$ from the induction. For $\lambda_1 = \lambda_2$ under the symmetric model, we get
$$\begin{aligned} \lambda_1 \min\big[J_k(\boldsymbol{q} + \boldsymbol{e}_1), w\phi + J_k(\boldsymbol{q} + \boldsymbol{e}_2)\big] = \\ \lambda_2 \min\big[w\phi + J_k(\tilde{\boldsymbol{q}} + \boldsymbol{e}_1), J_k(\tilde{\boldsymbol{q}} + \boldsymbol{e}_2)\big] \\ \lambda_2 \min\big[w\phi + J_k(\boldsymbol{q} + \boldsymbol{e}_1), J_k(\boldsymbol{q} + \boldsymbol{e}_2)\big] = \\ \lambda_1 \min\big[J_k(\tilde{\boldsymbol{q}} + \boldsymbol{e}_1), w\phi + J_k(\tilde{\boldsymbol{q}} + \boldsymbol{e}_2)\big] \end{aligned}$$

Therefore, (25) equals (26), which proves the lemma. ■

Lemma 2: The function $\Delta_k(\boldsymbol{q})$ is monotonically non-decreasing in q_1 for each fixed q_2.

Proof: We also prove this lemma using induction. First we make the following definitions:

$$\begin{aligned} f_1(\boldsymbol{q}) \equiv& \mu_1(\boldsymbol{q} + \boldsymbol{e}_1) J_k(\boldsymbol{q}) + \mu_2(\boldsymbol{q} + \boldsymbol{e}_1) J_k([\boldsymbol{q} + \boldsymbol{e}_1 - \boldsymbol{e}_2]^+) \\ &- \mu_1(\boldsymbol{q} + \boldsymbol{e}_2) J_k([\boldsymbol{q} - \boldsymbol{e}_1 + \boldsymbol{e}_2]^+) - \mu_2(\boldsymbol{q} + \boldsymbol{e}_2) J_k(\boldsymbol{q}) \\ f_2(\boldsymbol{q}) \equiv& \lambda_1 \big\{ \min\big[J_k(\boldsymbol{q} + 2\boldsymbol{e}_1), w\phi + J_k(\boldsymbol{q} + \boldsymbol{e}_1 + \boldsymbol{e}_2)\big] \\ &- \min\big[J_k(\boldsymbol{q} + \boldsymbol{e}_1 + \boldsymbol{e}_2), w\phi + J_k(\boldsymbol{q} + 2\boldsymbol{e}_2)\big] \big\} \\ f_3(\boldsymbol{q}) \equiv& \lambda_2 \big\{ \min\big[w\phi + J_k(\boldsymbol{q} + 2\boldsymbol{e}_1), J_k(\boldsymbol{q} + \boldsymbol{e}_1 + \boldsymbol{e}_2)\big] \\ &- \min\big[w\phi + J_k(\boldsymbol{q} + \boldsymbol{e}_1 + \boldsymbol{e}_2), J_k(\boldsymbol{q} + 2\boldsymbol{e}_2)\big] \big\} \end{aligned}$$

Induction basis: Since $J_0(\boldsymbol{q}) = 0$ for any $\boldsymbol{q} \in \mathbb{Z}^2$, we know $\Delta_0(\boldsymbol{q})$ is monotonically non-decreasing in q_1.

Induction step: Assume $\Delta_k(\boldsymbol{q})$ is monotonically non-decreasing in q_1 for each fixed q_2. By substituting (24) into (18), we have

$$\begin{aligned} \varphi \Delta_{k+1}(\boldsymbol{q}) =& \varphi\big(J_{k+1}(\boldsymbol{q} + \boldsymbol{e}_1) - J_{k+1}(\boldsymbol{q} + \boldsymbol{e}_2)\big) \\ =& f_1(\boldsymbol{q}) + f_2(\boldsymbol{q}) + f_3(\boldsymbol{q}) \end{aligned} \tag{27}$$

In the following we show that all $f_1(\boldsymbol{q}), f_2(\boldsymbol{q})$ and $f_3(\boldsymbol{q})$ are monotonically non-decreasing in q_1 for each fixed q_2.

$f_1(\boldsymbol{q})$: We have to consider different queue states around the diagonal, as shown in Fig 14. According to different queue state, we prove this part by five different cases.

Case 1: If $q_1 + 1 < q_2$, according to (19)-(21) we know $\mu_1(\boldsymbol{q} + \boldsymbol{e}_1) = \mu_1(\boldsymbol{q} + \boldsymbol{e}_2) = \mu_1''$ and $\mu_2(\boldsymbol{q} + \boldsymbol{e}_1) = \mu_2(\boldsymbol{q} + \boldsymbol{e}_2) = \mu_2''$. Thus $f_2(\boldsymbol{q})$ can be written as the following:

$$\begin{aligned} f_1(\boldsymbol{q}) \equiv& \mu_1''[J_k(\boldsymbol{q}) - J_k([\boldsymbol{q} - \boldsymbol{e}_1 + \boldsymbol{e}_2]^+)] \\ &+ \mu_2''[J_k([\boldsymbol{q} + \boldsymbol{e}_1 - \boldsymbol{e}_2]^+ - J_k(\boldsymbol{q})] \\ =& \mu_1'' \Delta_k([\boldsymbol{q} - \boldsymbol{e}_1]^+) + \mu_2'' \Delta_k([\boldsymbol{q} - \boldsymbol{e}_2]^+) \end{aligned} \tag{28}$$

which is non-decreasing in q_1 from the induction.

Case 2: If $q_1 + 1 = q_2$, we have $\mu_1(\boldsymbol{q} + \boldsymbol{e}_1) = \mu_1', \mu_1(\boldsymbol{q} + \boldsymbol{e}_2) = \mu_1''$ and $\mu_2(\boldsymbol{q} + \boldsymbol{e}_1) = \mu_2', \mu_2(\boldsymbol{q} + \boldsymbol{e}_2) = \mu_2''$ according to (19)-(21). Furthermore, since $q_1 + 1 = q_2$ and according

to lemma 1, we get $J_k(\boldsymbol{q}) = J_k(\boldsymbol{q}+\boldsymbol{e}_1-\boldsymbol{e}_2)$. Thus $f_2(\boldsymbol{q})$ can be written as:

$$\begin{aligned} f_1(\boldsymbol{q}) \equiv & \mu_1''[J_k(\boldsymbol{q}) - J_k([\boldsymbol{q}-\boldsymbol{e}_1+\boldsymbol{e}_2]^+)] \\ & + \mu_2''[J_k(\boldsymbol{q}+\boldsymbol{e}_1-\boldsymbol{e}_2) - J_k(\boldsymbol{q})] \\ = & \mu_1''\Delta_k([\boldsymbol{q}-\boldsymbol{e}_1]^+) + \mu_2''\Delta_k(\boldsymbol{q}-\boldsymbol{e}_2) \end{aligned} \tag{29}$$

which is non-decreasing in q_1 from the induction.

Case 3: If $q_1 = q_2$, we have $J_k([\boldsymbol{q}-\boldsymbol{e}_2]^+) = J_k([\boldsymbol{q}-\boldsymbol{e}_1]^+)$ according to lemma 1. Besides, we know $\mu_1(\boldsymbol{q}) = \mu_2(\boldsymbol{q}) = (\mu_1' + \mu_2')/2$. Thus $f_2(\boldsymbol{q}) - f_2([\boldsymbol{q}-\boldsymbol{e}_1]^+)$ is

$$\begin{aligned} f_1(\boldsymbol{q}) - f_1([\boldsymbol{q}-\boldsymbol{e}_1]^+) = & \ \mu_1' J_k(\boldsymbol{q}) + \mu_2' J_k([\boldsymbol{q}+\boldsymbol{e}_1-\boldsymbol{e}_2]^+) \\ & - \mu_1'' J_k([\boldsymbol{q}-\boldsymbol{e}_1+\boldsymbol{e}_2]^+) - \mu_2'' J_k(\boldsymbol{q}) \\ & - \frac{\mu_1''+\mu_2''}{2}\Big\{ J_k([\boldsymbol{q}-\boldsymbol{e}_1]^+) + J_k([\boldsymbol{q}-\boldsymbol{e}_2]^+)\Big\} \\ & + \mu_1'' J_k([\boldsymbol{q}-2\boldsymbol{e}_1+\boldsymbol{e}_2]^+) + \mu_2'' J_k([\boldsymbol{q}-\boldsymbol{e}_1]^+) \\ = & -\big\{\mu_1'' J_k([\boldsymbol{q}-\boldsymbol{e}_1]^+) - \mu_1'' J_k([\boldsymbol{q}-2\boldsymbol{e}_1+\boldsymbol{e}_2]^+)\big\} \\ = & -\Delta_k([\boldsymbol{q}-2\boldsymbol{e}_1]^+) \geq -\Delta_k(\boldsymbol{q}) = 0 \end{aligned} \tag{30}$$

Case 4: The proof under this case is similar to *Case 2*.

Case 5: The proof under this case is similar to *Case 1*.

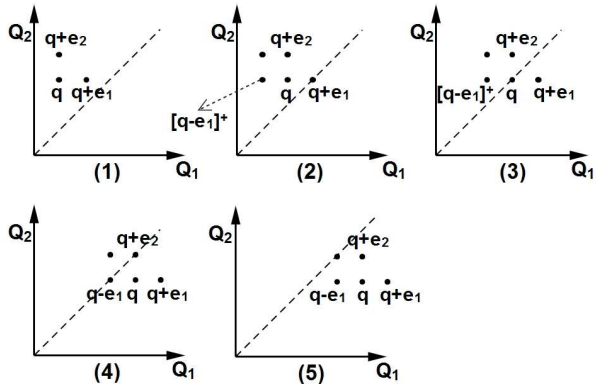


Figure 14: Different queue states $\boldsymbol{q} \in \mathbb{Z}_+^2$ around the diagonal.

$\underline{f_2(\boldsymbol{q})}$: $f_2(\boldsymbol{q})$ can be written as follows:

$$\begin{aligned} f_2(\boldsymbol{q}) = & \lambda_1\big\{ J_k(\boldsymbol{q}+\boldsymbol{e}_1+\boldsymbol{e}_2) - J_k(\boldsymbol{q}+\boldsymbol{e}_1+\boldsymbol{e}_2) \\ & + \min\big[J_k(\boldsymbol{q}+2\boldsymbol{e}_1) - J_k(\boldsymbol{q}+\boldsymbol{e}_1+\boldsymbol{e}_2), w\phi\big] \\ & - \min\big[0, w\phi + J_k(\boldsymbol{q}+2\boldsymbol{e}_2) - J_k(\boldsymbol{q}+\boldsymbol{e}_1+\boldsymbol{e}_2)\big]\big\} \\ = & \lambda_1\big\{ \min\big[\Delta_k(\boldsymbol{q}+\boldsymbol{e}_1), w\phi\big] + \max\big[0, \Delta_k(\boldsymbol{q}+\boldsymbol{e}_2) - w\phi\big]\big\} \\ = & \lambda_1\big\{ w\phi + \min\big[0, \Delta_k(\boldsymbol{q}+\boldsymbol{e}_1) - w\phi\big] \\ & + \max\big[0, \Delta_k(\boldsymbol{q}+\boldsymbol{e}_2) - w\phi\big]\big\} \end{aligned}$$

where $\Delta_k(\boldsymbol{q}+\boldsymbol{e}_1)$ and $\Delta_k(\boldsymbol{q}+\boldsymbol{e}_2)$ are monotonically non-decreasing in q_1 for each fixed q_2 from induction, resulting in $\min\big[0, \Delta_k(\boldsymbol{q}+\boldsymbol{e}_1) - w\phi\big]$ and $\Delta_k(\boldsymbol{q}+\boldsymbol{e}_2) - w\phi\big]$ are non-decreasing in q_1. Therefore, $f_2(\boldsymbol{q})$ is non-decreasing in q_1 for each fixed q_2.

$\underline{f_3(\boldsymbol{q})}$: Similarly, $f_3(\boldsymbol{q})$ can be written as follows:

$$\begin{aligned} f_3(\boldsymbol{q}) = \lambda_2\big\{ & -w\phi + \min\big[0, \Delta_k(\boldsymbol{q}+\boldsymbol{e}_1) + w\phi\big] \\ & + \max\big[0, \Delta_k(\boldsymbol{q}+\boldsymbol{e}_2) + w\phi\big]\big\} \end{aligned}$$

which is non-decreasing in q_1 for each fixed q_2.

Therefore, this lemma is proved since all $f_1(\boldsymbol{q})$, $f_2(\boldsymbol{q})$ and $f_3(\boldsymbol{q})$ are monotonically non-decreasing in q_1 for each fixed q_2. ■

Proof of Theorem 2: We consider the optimal dispatching policy characterized by Theorem 1. To show the optimal policy has switching curves for the two-user homogeneous scenarios, it is sufficient to show that $\Delta_k(\boldsymbol{q})$ is monotonically non-decreasing in q_i for each fixed q_j, $i, j \in \{1, 2\}$ and $i \neq j$ [20], which can be obtained from lemma 2. ■

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," *Communications Magazine, IEEE*, 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] 3GPP2, "Cdma2000 high rate packet data air interface specication," *C.S20024-A v. 1.0*, 2004.

[4] ——, "Dual-cell high speed downlink packet access (hsdpa) operation," *TR 25.825 v.1.0.0*, 2008.

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

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

[7] A. Jalali, R. Padovani, and R. Pankaj, "Data throughput of cdma-hdr a high efficiency-high data rate personal communication wireless system," in *IEEE VTC 2000-Spring,*, vol. 3, 2000, pp. 1854 –1858 vol.3.

[8] H. Falaki, D. Lymberopoulos, R. Mahajan, S. Kandula, and D. Estrin, "A first look at traffic on smartphones," in *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, ser. IMC '10. New York, NY, USA: ACM, 2010, pp. 281–287.

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

[10] S. Shakkottai and A. L. Stolyar, "Scheduling for multiple flows sharing a time-varying channel: The exponential rule," *American Mathematical Society Translations, Series*, vol. 2, p. 2002, 2000.

[11] L. Tassiulas and A. Ephremides, "Dynamic server allocation to parallel queues with randomly varying connectivity," *Information Theory, IEEE Transactions on*, vol. 39, no. 2, pp. 466 –478, Mar. 1993.

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

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

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

[15] M. E. Crovella, M. Harchol-Balter, and C. D. Murta, “Task assignment in a distributed system (extended abstract): improving performance by unbalancing load,” ser. SIGMETRICS’98. New York, NY, USA: ACM, 1998, pp. 268–269.

[16] 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.

[17] B. Hajek, “Optimal control of two interacting service stations,” *Automatic Control, IEEE Transactions on*, vol. 29, no. 6, pp. 491 – 499, Jun. 1984.

[18] A. Penttinen, E. Hyytia, and S. Aalto, “Energy-aware dispatching in parallel queues with on-off energy consumption,” in *Performance Computing and Communications Conference (IPCCC), 2011 IEEE 30th International*, nov. 2011, pp. 1 –8.

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

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