SCHEDULING POLICIES IN A CELLULAR NETWORK WITH DATA QUEUEING FOR USERS

Nomesh Bolia*

nomesh@mech.iitd.ac.in

Department of Mechanical Engineering, IIT Delhi

Vidyadhar Kulkarni

vkulkarn@email.unc.edu

Department of Statistics and Operations Research, UNC Chapel Hill

Running Head: Scheduling in Cellular Networks

Abstract *We consider scheduling policies for data transfer between the base station and users where data for every user is allowed to queue up for transmission within a cell of a wireless telecommunication network. We consider a slotted system where at most one user can be served in a slot and available data rate available for transfer depends on the varies with user and time. We derive an index policy that is stable and attempts to minimize congestion. We compare the performance of our index policy with existing policies through simulation.*

*Contact Author

1 Introduction

Data transfer over the internet has become an important application of wireless cellular systems. To ensure effective utilization of the available resources, we require effective algorithms to enable data flow between users and the base station in an efficient and fair manner.

In this paper we consider the problem of scheduling users to be served for data transfer in one cell of a cellular network. All users in the cell are served by a single base station. To begin with we consider a fixed number N of users. Later, in section 4.4 we look at the case where users can arrive in and depart from the cell. Time is slotted and exactly one user is served in a given time slot n as long as there is data to serve for at least one user. We assume data is quantized in packets of equal size and use "data" and "number of packets" interchangeably. The data rate (packets per time slot) available to any user during the time slot n depends on its dynamic "channel conditions" such as the distance of the user from the base station, the topography and the weather conditions faced by the user, etc. We concentrate on the downlink channel, i.e. data transfer from the base station to users. Let R_u^n $(u = 1, 2, \ldots, N;\ n = 0, 1, \ldots)$ be the data rate available to user u during time slot n, i.e., the number of packets that can potentially be transmitted to user u during time slot n from the base station. The base station chooses the user to serve during each slot n using the data rate vector $R^n = [R_1^n, R_2^n, \ldots, R_N^n]$ that is obtained using engineering systems already in place and their details are irrelevant to the algorithms presented in this paper. An example of such a system in place is CDMA2000 1xEV-DO system [6]. We simply assume that $\{R^n,\ n \geq 0\}$ is a stochastic process that accounts for the dynamic channel conditions.

A stable scheduling algorithm restricts the queue lengths from increasing without any bounds and induces a stationary distribution of data queue length for all users. The throughput of any stable algorithm equals its input data rate provided the input rates are in the system's stability region. To see this, consider the data queue of any user: in steady state, whatever data enters

the queue has to leave it, thus implying a fixed throughput that depends on the average rate of data arrival. In the absence of throughput maximization, a useful measure of overall customer satisfaction is the total length of the data queues of all users. The service provider attempts to empty all data queues as quickly as possible, so that it can serve all the demand of data with the smallest delay possible. Thus a reasonable objective is to minimize the total weighted data queue length.

It is well known that Markov Decision Processes (MDP) can be used to determine optimal policies (that is, which user to serve in this case) in this setting. Further, previous experience [16] suggests that one can develop index policies [21, 10] that are nearly optimal and easy to implement. An index policy in the context of MDPs is a policy that gives a prescription of the action to take based on an *index* for every action. The index of an action is a closed form expression associated with that action and depends on the current state of the MDP. We thus have as many indices as the number of actions in the action space, and the *index policy* is simply to choose the action whose index is maximum or minimum depending on the objective of the MDP. The index we develop in this paper is related to our past work [7] that deals with infinitely backlogged data queues in a similar setting. It should be noted that our index policy is different from the one analyzed by Whittle [22] for the multiarmed bandits problem - the Gittins index policy in [22] considers discounted rewards and no state changes when user is not served, whereas we have average rewards, and users changing states regardless of service status.

In the setting of external data arrival considered in this paper however, stability of queues is a serious concern. We prove the stability of our recommended policy and our results are similar to the stability results of the **Max-Weight** type scheduling algorithms. The first results of this type were obtained by Tassiulas and Ephremides [20]. A significant amount of work has since appeared on proving the stability of algorithms similar to the **Max-Weight algorithm** (MWA)

such as Neely et al [14] and Andrews et al [3]. Andrews, Jung and Stoylar [2] prove the stability of the MWA for dynamic networks. It has also been studied in other situations such as scheduling input-queued crossbar switches [13] and maximizing the total utility of traffic injected into the network [9]. More recently Celik, Le and Modiano [8], for the first time, address the scheduling problem taking switchover delay into account.

The rest of this paper is organized as follows: In section 2 we describe some existing algorithms including the MWA for the scheduling problem. We formulate the problem as an MDP in section 3. We derive the index policy and briefly look at the case of users entering and leaving the cell in section 4. We prove the the stability of the index policy in section 5. In section 6 we briefly describe the performance analysis methodology to compare our recommended index policy with the MWA. Finally, section 7 contains results of a simulation experiment comparing the performance of our index policy with exisiting policies.

2 Existing Algorithms

In this section we describe an existing algorithm to solve the scheduling problem described in section 1. For this and the following sections, let $v(n)$ be the user chosen for service, Q_u^n be the number of packets waiting to be served in the queue for user u and A_u^n be the number of packets that arrive externally for user u during time slot n.

1. Max-Weight Algorithm (MWA): This algorithm was first introduced by Tassiulas and Ephremides in [20] and Awerbuch and Leighton in [5]. In time slot n, it serves the user u ($u \in \mathcal{A}$) for whom the product of the available data rate and the total data available to transmit is maximized. Mathematically,

$$v(n) \in \arg\max_u R_u^n(Q_u^n + A_u^n). \tag{2.1}$$

The MWA minimizes delay (sum of queue lengths) in symmetric systems and a stability condition, i.e., condition(s) under which queue lengths are bounded, for this algorithm is derived in [20].

2. The Exponential Rule: See Shakkottai and Stolyar [18]. Let γ_u, a_u, $u = 1, 2, \ldots, N$ and β, $\eta \in (0, 1)$ be positive constants. Then the exponential rule serves the user $v(n)$ in time slot n such that

$$v(n) \in \arg\max_u \gamma_u R_u^n exp\left(\frac{a_u Q_u^n}{\beta + (\bar{Q}^n)^\eta}\right), \tag{2.2}$$

where $\bar{Q}^n = (1/N)\sum_u a_u Q_u^n$. This algorithm has the property that for each time n, it minimizes $\max_u a_u Q_u^n$ [19]. The constants β, η and the arbitrary parameters a_u, γ_u give more flexibility to the scheduling algorithm. However, no method is specified to select these parameters.

Note that the above algorithms are instances of index policies with indices given by (2.1) and (2.2). The scheduling algorithm based on each of the above indices is also proven to be stable. We derive our index policy based on an MDP formulation that attempts to minimize the long run total data queue lengths. The policy based on this index turns out to be similar to the MWA. We evaluate its performance using simulation. Further, we do not actually solve the MDP to optimality; the MDP formulation merely gives us a starting point and a sound theoretical framework for the derivation of the final index policy. Furthermore, we prove that the index computed using our approach generates a stable scheduling policy.

3 MDP Formulation

We formulate the problem as an MDP in this section. We start with a model for the data rate process $\{R^n, n \geq 0\}$, the data arrival process $\{A^n, n \geq 0\}$ and the queue length process $\{Q^n, n \geq 0\}$ of the users. In systems currently used in practice such as CDMA2000 1xEV-DO system [6], the base station serves users at one of the M data rates $r_1, r_2, \ldots, r_M$. Each of these M data rates

corresponds to the "channel state" of a given user. This "channel state" of the user takes into account factors such as distance from the base station and topography. Let X_u^n be the channel state of user u during the time slot n. We use the following model for X_u^n: $\{X_u^n,\ n \geq 0\}$ is an irreducible Discrete Time Markov Chain (DTMC) on state space $\Omega = \{1, 2, \ldots, M\}$ with Transition Probability Matrix (TPM) $P^u = [p^u_{i_u,j_u}]$. For example, a set of $M = 11$ fixed data rates is available to users in an actual system [6]. During any time slot n, for every user u the underlying DTMC $\{X_u^n,\ n \geq 0\}$ determines the data rate R_u^n as follows: for $k \in \Omega$ $X_u^n = k \Longrightarrow R_u^n = r_k$. Further, let $X^n = [X_1^n, \ldots, X_N^n]$ be the channel state vector of all the users. We assume that all users are independent of each other and thus each component of $\{X^n, n \geq 0\}$ is an independent DTMC on Ω. Hence it is clear that $\{X^n,\ n \geq 0\}$ itself is a DTMC on Ω^N.

Next we consider the data model for each user. The base station maintains a separate queue for every user u. Let A_u^n be the number of packets that arrive at the base station for user u during time slot n. We assume that the arrival process $\{A_k^n, n \geq 0\}$ is independent of $\{A_l^n, n \geq 0\}$ for $k \neq l$. Further we assume that for every u, $\{A_u^n,\ n \geq 0\}$ is an irreducible DTMC on state space $Z = \{0, 1, 2, \ldots\}$ with TPM $H^u = [h^u_{i_u,j_u}]$. Since each component of $\{A^n, n \geq 0\}$ is an independent DTMC on Z, $\{A^n,\ n \geq 0\}$ itself is a DTMC on Z^N. An example of a Markovian packet arrival process is when $\{A_u^n, n \geq 0\}$ is a sequence of independent and identically distributed (i.i.d.) random variables with the probability distribution $h^u = [h_0^u, h_1^u, \ldots]$. In this case every row of the TPM H^u is equal to h^u. In this and the following sections we also refer to A_u^n as the "packet arrival state" of user u in time slot n.

We can now describe the evolution of the queue length process $\{Q^n, n \geq 0\}$. Recall that, for every user u, Q_u^n is the number of data packets in the queue for user u at the beginning of time

slot n. Therefore $\{Q_u^n, n \geq 0\}$ changes according to

$$Q_u^{n+1} = \begin{cases} Q_u^n + A_u^n & \text{if } u \neq v(n) \\ Q_u^n + A_u^n - \min(R_u^n, Q_u^n + A_u^n) & \text{if } u = v(n). \end{cases} \tag{3.1}$$

The "state" of user $u \in \mathcal{A}$, given by the vector $[X_u^n, Q_u^n, A_u^n] \in \Omega \times Z \times Z$, thus has three components: the "channel state" X_u^n, the "queue length state" Q_u^n and the "data arrival state" A_u^n. The "state of the system" at time n is then given by $[X^n, Q^n, A^n] \in \Omega^N \times Z^N \times Z^N$. It is thus a vector of $3N$ components and we assume that it is known to the base station in each time slot.

Unless the data queues of all users are empty, the base station serves exactly one user in every time slot after observing the system state in that time slot. We need a cost structure to make this decision optimally and we propose it below. We pay a cost K_u to hold one packet for one time slot for any user $v \in \mathcal{A}$. The parameters K_u are a mechanism to distinguish users, so their relative values are more important than absolute esimates. Therefore, if a particular class of users are high-end customers (more expensive data plan, for instance) K_u for them will be higher and so on. The higher K_u is, the more important user u is, so the service provider can assign K_u for different users depending on their relative importance to its revenues. If user u is served, the queue length of every user $l \neq u$ is $Q_l^n + A_l^n$. For user u, however, the queue length (after transmission) equals $(Q_u^n + A_u^n - R_u^n)^+$ where $(x)^+ = \max(x, 0)$ for any real number x. The total holding cost W_u^n after serving user u in time slot n is then given by

$$\begin{aligned} W_u^n &= \sum_{l \neq u} K_l \cdot (Q_l^n + A_l^n) + K_u \cdot (Q_u^n + A_u^n - R_u^n)^+ \\ &= \sum_{l} K_l \cdot (Q_l^n + A_l^n) - K_u \min\left(Q_u^n + A_u^n, R_u^n\right). \end{aligned} \tag{3.2}$$

Having described the system state, its evolution and the cost structure, we can now model the problem of scheduling a user to serve in a given time slot as an MDP. The decision epochs are the time slots $\{1, 2, \ldots\}$. The state at time n is $[X^n, Q^n, A^n]$ with Markovian evolution described above.

The action space in every state is $\mathcal{A} = \{1, 2, \ldots, N\}$ where action u corresponds to serving user u. The cost in state $[X^n, Q^n, A^n]$ for action u is W_u^n. The transition probability $p((j,z,b)|(i,y,a),u)$ under action u from (i,y,a) to (j,z,b) $(i,j \in \Omega^N,\ y,z \in Z^N,\ a,b \in Z^N)$ is given by

$$p((j,z,b)|(i,y,a),u) = \begin{cases} p^1_{i_1,j_1} \ldots p^N_{i_N,j_N} h^1_{a_1,b_1} \ldots h^N_{a_N,b_N} = p_{ij} h_{ab} & \text{if } z_u = (y_u + a_u - r_{i_u})^+ \text{ and} \\ & z_l = y_l + a_l \text{ for } l \neq u \\ 0 & \text{otherwise.} \end{cases} \tag{3.3}$$

Next we describe the value functions that form the basis of the Policy Improvement Algorithm (PIA). Let $V_D(i,y,a)$ be the optimal cost starting from state $[X^0, Q^0, A^0] = [i,y,a]$ at time 0 over time periods $0, 1, 2, \ldots, D-1$. If user u is served at time 0 the queue length vector in the next time slot is

$$(y+a,i)^u = (y_1 + a_1, \ldots, y_{u-1} + a_{u-1}, (y_u + a_u - r_{i_u})^+, y_{u+1} + a_{u+1}, \ldots, y_N + a_N). \tag{3.4}$$

For the sake of notational convenience, we define

$$W_u(i, y+a) = \sum_l K_l \cdot (y_l + a_l) - K_u \min\left(y_u + a_u, r_{i_u}\right). \tag{3.5}$$

A standard Dynamic Programming (DP) argument then yields the following Bellman equation

$$V_D(i,y,a) = \min_{u=1,2,\ldots,N} \left[W_u(i, y+a) + \sum_{(j,b)} p_{ij} h_{ab} V_{D-1}(j, (y+a,i)^u, b) \right]. \tag{3.6}$$

The goal of the scheduling policy is to determine the action $u = u(i,y,a)$ that minimizes the long run average cost $\lim_{D\to\infty} V_D(i,y,a)/D$ given the state $(i,y,a) \in \Omega^N \times Z^N \times Z^N$ of the system. It is well known from standard MDP theory [17] that such a policy $\{u(i,y,a)\}$ exists in the infinite horizon $(D \to \infty)$ if there is a constant g and a bias function $w(i,y,a)$ satisfying

$$g + w(i,y,a) = \min_u \{ W_u(i, y+a) + \sum_{(j,b)} p_{ij} h_{ab} w(j, (y+a,i)^u, b) \}. \tag{3.7}$$

Any u that minimizes $W_u(i, y+a) + \sum_{(j,b)} p_{ij} h_{ab} w(j, (y+a,i)^u)$ over all $u \in \{1, \ldots, N\}$ is an

optimal action $u(i, y, a)$ in state (i, y, a).

4 The Index Policy

Since solving the MDP to optimality is not feasible, we use the PIA to develop an easily implementable heuristic scheduling policy. The standard PIA [17] in this setting is given by:

1. Let π^0 be an arbitrary policy that chooses action $\pi^0(i, y, a) \in \mathcal{A}$ in state (i, y, a). Set $n = 0$.

2. Policy Evaluation Step: For $(i, y, a) \in \Omega^N \times Z^N \times Z^N$, solve the equations

$$g_n + w_n(i, y, a) = W_u(i, y + a) + \sum_{(j,b)} p_{ij} h_{ab} w_n(j, (y + a, i)^u, b),$$

   for g_n and $\{w_n(i, y, a) : i \in \Omega^N, y \in Z^N\, a \in Z^N\}$ where $u = \pi^n(i, y, a)$ and n denotes the number of iterations so far.

3. Policy Improvement Step: Let

$$\pi^{n+1}(i, y, a) \in \arg\min_{u \in \mathcal{A}} \{W_u(i, y + a) + \sum_{(j,b)} p_{ij} h_{ab} w_n(j, (y + a, i)^u, b)\}. \tag{4.1}$$

   If $\pi^n(i, y, a)$ minimizes the Right Hand Side (RHS), choose $\pi^{n+1}(i, y, a) = \pi^n(i, y, a)$.

4. If $\pi^{n+1} \neq \pi^n$, set $n = n + 1$ and go to step 2. Else, STOP. π^{n+1} is the optimal policy.

Under certain conditions [17] one can show that this algorithm terminates in a finite number of steps.

We use the policy π^1 (derived from one iteration of the PIA) as an approximation for the optimal policy. We shall show that π^1, after some approximations, is simple to implement and validate by simulation that it improves upon the existing max-weight algorithm. The PIA approach does not require π^1 to be computed in every time slot, computing it once yields a closed form expression for an index for each user $u \in \mathcal{A}$ that is valid in every time slot. This index is solely based on its state (i_u, y_u, a_u) and the means of the packet arrival process $\{A_u^n, n \geq 0\}$ and the packet departure

process $\{R_u^n, n \geq 0\}$ of user u. The heuristic policy π^1 that we recommend is to serve the user whose index is minimized. Although we use the current Markovian structure for the evolution of the channel state of users, the final index is independent of this Markovian structure. Derivation of these indices involves choosing an initial policy and then using one step of the PIA. We discuss such an initial policy in the next subsection.

4.1 Initial Policy

We consider the following state independent stationary policy as the initial policy π^0: Serve user u with probability q_u in any time slot, where $q_1, q_2, \ldots, q_N$ are fixed numbers such that for each $u \in \mathcal{A}$

$$q_u > 0, \qquad \sum_u q_u = 1. \tag{4.2}$$

Let $q = [q_1, q_2, \ldots, q_N]$, g_q be the long run cost and $w_q = \{w(i,y,a) : (i,y,a) \in \Omega^N \times Z^N \times Z^N\}$ be the bias vector for this policy satisfying the equation

$$g_q + w_q(i,y,a) = \sum_{u=1}^{N} q_u \left\{ W_u(i, y+a) + \sum_{(j,b)} p_{ij} h_{ab} w_q(j, (y+a,i)^u, b) \right\}. \tag{4.3}$$

We refer to this policy as the *randomized policy* or the *q-policy*. The equation (4.3) assumes stability of the queues, i.e., $\{(X^n, Q^n, A^n), n \geq 0\}$ is positive recurrent. We describe necessary and sufficient conditions for stability of the q-policy below.

Let $\pi^u = [\pi_1^u, \ldots, \pi_M^u]$ be the steady state distribution of the Markov Chain $\{X_u^n : n \geq 0\}$, $u = 1, 2, \ldots, N$. Then, since M is finite and the DTMC is irreducible, it is well known [11] that π^u exists and is the unique solution to

$$\pi^u = \pi^u P^u, \quad \sum_{m=1}^{M} \pi_m^u = 1.$$

Then we know from standard DTMC theory [11] that the long run average of $\{R_u^n, n \geq 0\}$ is

$$\bar{R}_u = \sum_{k=1}^{M} r_k \pi_k^u. \tag{4.4}$$

We further assume that for all $u \in \mathcal{A}$ the steady state distribution of the Markov Chain $\{A_u^n : n \geq 0\}$ exists and is given by $\theta^u = [\theta_0^u, \theta_1^u, \ldots]$. Let λ_u be the long run average of the number of data packets that arrive for user u in one time slot. Then, from standard DTMC theory [11]

$$\lambda_u = \sum_{k=0}^{\infty} k\theta_k^u. \tag{4.5}$$

A straightforward extension of the results on Matrix-Geometric methods for stochastic models [15] can be used to show that the randomized q-policy is stable (i.e., each user's queue is stable) if

$$\lambda_u < q_u \bar{R}_u, \qquad u = 1, 2, \ldots, N. \tag{4.6}$$

To get an idea of the physical significance of g_q and $w_q(i, y, a)$ let us look at the long term cost under this policy. Let $V_D^q(i, y, a)$ be the total cost in periods 0 through $D - 1$ starting in state (i, y, a) under this initial q-policy. We use $o(D)$ to denote terms that go to zero as D approaches ∞. Then using standard MDP theory [17] we have

$$V_D^q(i, y, a) = g_q D + w_q(i, y, a) + o(D). \tag{4.7}$$

Thus for a fixed q-policy and state (i, y, a) we can think of $V_D^q(i, y, a)$ as an asymptotically linear function of D with slope g_q and intercept $w_q(i, y, a)$.

Now consider the N queues $\{Q_u^n, n \geq 0\}$ for $u \in \mathcal{A}$ under the following policy: In every time slot serve user u with probability q_u independent of all other users. For $u \in \mathcal{A}$, the user u queue is fed by its independent arrival process $\{A_u^n, n \geq 0\}$ as above. We call this the q_u *policy* for user u. Under the policy q_u, let $V_D^{q_u}(i_u, y_u, a_u)$ be the total cost accumulated by user u starting from state $[X_u^0, Q_u^0, A_u^0] = [i_u, y_u, a_u]$ at time 0 over time periods $0, 1, 2, \ldots, D - 1$. The mean cost incurred by user u in time slot 0 is $q_u K_u(y_u + a_u - r_{i_u})^+ + (1 - q_u)K_u(y_u + a_u)$. Thus the standard DTMC

theory [11] yields

$$V_D^{q_u}(i_u, y_u, a_u) = q_u K_u(y_u + a_u - r_{i_u})^+ + (1 - q_u)K_u(y_u + a_u)$$

$$+ \sum_{j_u, b_u} q_u p^u_{i_u, j_u} h^u_{a_u, b_u} V^{q_u}_{D-1}(j_u, (y_u + a_u - r_{i_u})^+, b_u) + \sum_{j_u, b_u} (1 - q_u) p^u_{i_u, j_u} h^u_{a_u, b_u} V^{q_u}_{D-1}(j_u, y_u + a_u, b_u).$$

Let g_{q_u} be the long run cost and $w_{q_u} = \{w_u(i_u, y_u, a_u) : (i_u, y_u, a_u) \in \Omega \times Z \times Z\}$ be the bias vector for the policy q_u. It is known to satisfy the equation

$$g_{q_u} + w_{q_u}(i_u, y_u, a_u) = q_u K_u(y_u + a_u - r_{i_u})^+ + q_u \sum_{(j_u, b_u)} p^u_{i_u j_u} h^u_{a_u b_u} w_{q_u}(j_u, (y_u + a_u - r_{i_u})^+, b_u)$$

$$+ (1 - q_u)[K_u(y_u + a_u) + \sum_{(j_u, b_u)} p^u_{i_u j_u} h^u_{a_u b_u} w_{q_u}(j_u, y_u + a_u, b_u)]. \tag{4.8}$$

Standard MDP theory [17] yields

$$V_D^{q_u}(i_u, y_u, a_u) = g_{q_u} D + w_{q_u}(i_u, y_u, a_u) + o(D), \tag{4.9}$$

which is the counterpart of (4.7) for user u. Then the following theorem gives an intuitive and useful relation between g_q, $w_q(i, y, a)$ and g_{q_u}, $w_{q_u}(i_u, y_u, a_u)$. The proof of the theorem is given in the appendix.

Theorem 4.1. *Let g_q be a constant and $\{w_q(i, y, a) : i \in \Omega^N, y \in Z^N\, a \in Z^N\}$ a function satisfying (4.3). Then for all $u \in \mathcal{A}$, there exist constants g_{q_u} and functions $\{w_{q_u}(i_u, y_u, a_u) : (i_u, y_u, a_u) \in \Omega \times Z \times Z\}$ that satisfy (4.8) and*

$$g_q = \sum_u g_{q_u} \tag{4.10}$$

$$w_q(i, y, a) = \sum_u w_{q_u}(i_u, y_u, a_u). \tag{4.11}$$

4.2 Policy Evaluation

From (4.1) of the PIA the policy improvement step seeks to minimize

$$\min(W_u(i, y + a) + \sum_{(j,b)} p_{ij} h_{ab} w_q(j, (y + a, i)^u, b) \tag{4.12}$$

over all $u \in \mathcal{A}$. As seen in the next subsection, we do not need to compute $w_q(i, y, a)$, computing the difference $w_q(j, (y + a, i)^u, b) - w_q(j, y + a, b)$ is sufficient. Computing this is a key step in the derivation of the index since the computation of $w_q(i, y, a)$ itself is intractable. We will need the following notations to compute $w_q(j, (y + a, i)^u, b) - w_q(j, y + a, b)$. Let

$$Z_u = \min\{m \geq 0 : Q_u^m = 0\}, \quad u \in \mathcal{A}, \tag{4.13}$$

$$T_u(i_u, y_u, a_u) = \mathrm{E}[Z_u | X_u^0 = i_u, Q_u^0 = y_u, A_u^0 = a_u]. \tag{4.14}$$

For a stable randomized policy, $T_u(i_u, y_u, a_u) < \infty$ for all $(i_u, y_u, a_u) \in \Omega \times Z \times Z$ and all $u \in \mathcal{A}$.

Consider two sample paths $\{(X^{n,m}, Q^{n,m}, A^{n,m}), n \geq 0\}$, $m = 1, 2$, of the $\{(X^n, Q^n, A^n), n \geq 0\}$ process that are coupled as follows:

$$X^{0,1} = X^{0,2} = j, \quad A^{0,1} = A^{0,2} = a, \quad Q^{0,1} = (y + a, i)^u, \quad Q^{0,2} = y + a \tag{4.15}$$

$$A^{n,1} = A^{n,2}, \quad X^{n,1} = X^{n,2} \qquad \text{for } n \geq 0. \tag{4.16}$$

Let $v^m(n)$ be the user served in slot n along sample path m, and

$$v^1(n) = v^2(n) \quad \text{for } n \geq 0. \tag{4.17}$$

Now we introduce the notation $w_q^{\Delta}(j, y, (y + a, i)^u, b) = w_q(j, (y + a, i)^u, b) - w_q(j, y + a, b)$. It can be thought of as the expected difference of the cost accumulated along the two sample paths $\{(X^{n,m}, Q^{n,m}, A^{n,m}), n \geq 0\}$, $m = 1, 2$ described above. The queue length trajectory along both paths is complicated and hence it is not possible to compute exact closed form expression for $w_q^{\Delta}(j, y, (y+a, i)^u, b)$. The next theorem, however, presents closed form expressions for a lower and an upper bound of $w_q^{\Delta}(j, y, (y + a, i)^u, b)$.

Theorem 4.2. *For a stable randomized policy,*

$$-K_u \min(r_{i_u}, y_u + a_u) T_u(i_u, (y_u + a_u - r_{i_u})^+, a_u) \geq w_q^{\Delta}(j, y, (y + a, i)^u, b)$$

$$-K_u \min(r_{i_u}, y_u + a_u) T_u(i_u, y_u + a_u, a_u) \leq w_q^{\Delta}(j, y, (y + a, i)^u, b). \tag{4.18}$$

The proof is given in the appendix. Having computed the bias difference $w_q^{\Delta}(j, y, (y + a, i)^u, b)$, we now derive the one-step improved policy in the next subsection.

4.3 Policy Improvement Step

We apply one step of the PIA and use some further approximations to derive an index for every user. Now minimizing the expression in (4.12) over all $u \in \mathcal{A}$ is equivalent to minimizing

$$I_u^{'''}(i, y, a) = W_u(i, y + a) - \sum_l K_l(y_l + a_l) + \sum_{(j,b)} p_{ij} h_{ab} \left[w_q(j, (y + a, i)^u, b) - w_q(j, y + a, b) \right] \tag{4.19}$$

over all $u \in \mathcal{A}$ since for a given (i, y, a), the additional term $-\sum_l K_l(y_l + a_l) + \sum_{(j,b)} p_{ij} h_{ab} w_q(j, y + a, b)$ does not depend on u. The improved policy (our recommended policy) then serves the user with the highest index $I_u^{'''}(i, y, a)$. The next theorem gives an upper and lower bound for $I_u^{'''}(i, y, a)$.

Theorem 4.3. *$I_u^{'''}(i, y, a)$ given by (4.19) is a function only of the state of user u, i.e.,*

$$I_u^{'''}(i, y, a) = I_u^{''}(i_u, y_u, a_u), \tag{4.20}$$

$$-K_u \min(r_{i_u}, y_u + a_u) T_u(i_u, y_u, a_u) \geq I_u^{''}(i_u, y_u, a_u) \geq -K_u \min(r_{i_u}, y_u + a_u) T_u(i_u, y_u + r_{i_u}, a_u) \tag{4.21}$$

The proof can be found in the appendix. Thus we know the upper and lower bound for the index

$I_u^{''}(i_u, y_u, a_u)$. One can use these bounds to define the Average Index (AI) given by

$$I_u^{'}(i_u, y_u, a_u) = K_u \min(r_{i_u}, y_u + a_u) \left[T_u(i_u, y_u + r_{i_u}, a_u) + T_u(i_u, y_u, a_u)\right] \tag{4.22}$$

Note that $I_u^{'}(i_u, y_u, a_u)$ has been obtained by taking the average of the upper and lower bound of $I_u^{''}(i_u, y_u, a_u)$ and multiplying the result by -1. Hence whereas the index policy based on $I_u^{'''}(i_u, y_u, a_u)$ is to serve the user u for whom $I_u^{''}(i_u, y_u, a_u)$ is *minimized*, the index policy based on $I_u^{'}(i_u, y_u, a_u)$ is to serve the user u for whom $I_u^{'}(i_u, y_u, a_u)$ is *maximized*. We will see in the simulation results that this index policy performs better than the MWA for the entire traffic range.

Exact values of the first passage times $T_u(i_u, y_u + r_{i_u}, a_u)$ and $T_u(i_u, y_u, a_u)$ can be obtained only by numerically solving a set of countably infinite first passage time [11]. Hence, we use the following to approximate the first passage times. Consider the queue of user u. On an average, the number of net departures per time slot is $q_u \bar{R}_u - \lambda_u$. Hence, starting with y_u packets in the queue, the expected time when the queue length Q_u^n first hits zero can be approximated by

$$T_u(i_u, y_u, a_u) \approx \frac{y_u}{q_u \bar{R}_u - \lambda_u}. \tag{4.23}$$

Using the approximation (4.23) yields the following expressions for AI:

$$I_u(i_u, y_u, a_u) = \frac{K_u}{q_u \bar{R}_u - \lambda_u} (2y_u + r_{i_u}) \min(r_{i_u}, y_u + a_u). \tag{4.24}$$

When all users are identical, K_u, q_u, $\bar{R}_u$ and λ_u don't depend on u we recommended using $q_u = 1/N$ yielding the index

$$I_u(i_u, y_u, a_u) = (2y_u + r_{i_u}) \min(r_{i_u}, y_u + a_u). \tag{4.25}$$

The general expression for the index in (4.24) immediately begs the question of the choice of

$q_u, u \in \mathcal{A}$. We choose q_u so that the resulting index policy is stable as discussed in section 5 in general and (5.3) in particular. We will also see in section 5 that $q_u = 1/N$ yields stable index policies for stochastically identical users. Substituting this in (4.24) we get the Uniform Index (UI)

$$I_u(i_u, y_u, a_u) = \frac{K_u}{\bar{R}_u/N - \lambda_u}(2y_u + r_{i_u}) \min(r_{i_u}, y_u + a_u). \tag{4.26}$$

We can now describe our Index Policy (IP) as follows: In time slot n, given state (i, y, a) of the system, the user $v_{IP}(n)$ served according to the IP is

$$\text{IP}: \quad v_{IP}(n) \in \arg\max_u \frac{K_u}{q_u \bar{R}_u - \lambda_u} \min(R_u^n, Q_u^n + A_u^n) \left[2Q_u^n + R_u^n\right]. \tag{4.27}$$

In particular we use the UI given by (4.26) to yield the Uniform Index Policy (UIP). Thus, the user $v_{UIP}(n)$ served in the n^{th} time slot according to the UIP is

$$\text{UIP}: \quad v_{UIP}(n) \in \arg\max_u \frac{K_u}{\bar{R}_u/N - \lambda_u} \min(R_u^n, Q_u^n + A_u^n) \left[2Q_u^n + R_u^n\right]. \tag{4.28}$$

4.4 Variable Number of Users

In the analysis so far, no user is allowed to arrive in or depart from the cell. We call such a cell '*static*'. Next we describe a version of IP for the '*dynamic*' cell that allows for both user arrivals and departures. We do not intend to analyze the dynamic cell rigorously in this paper, i.e., we do not derive a scheduling policy taking the "dynamic" nature of the cell into the account. Rather, we present the results of our policy for the dynamic cell merely as an extension of what happens in this setting, if in every slot n, we use the IP (derived for the static cell) by replacing the constant N of the static cell with the number of users $N(n)$ in slot n in the dynamic cell. Wr refer the reader to [12] for a more rigorous treatment of scheduling in the presence of such flow level dynamics. In

this section, to derive $N(n)$, we assume users arrive according to a Poisson process with rate λ. Once in the cell, sojourn time of a user is generally distributed with mean a. Thus, in steady state the number of users is a Poisson random variable with mean $N_{\text{avg}} = \lambda a$. Then $\{N(n), n \geq 0\}$ is the number of customers in an $M/G/\infty$ queue and the index given by equation (4.26) is modified for this case as follows:

$$I_u(i_u, y_u, a_u) = \frac{K_u}{\bar{R}_u/N(n) - \lambda_u} \min(r_{i_u}, y_u + a_u) \left[2y_u + r_{i_u}\right]. \tag{4.29}$$

However, since we only look at the case of stochastically identical users in section 7, the UIP in this case reduces to maximizing $I_u(i_u, y_u, a_u)$ given by (4.25) over all users that are present in the cell. The stability of this system follows from the fact that every user leaves the cell eventually bringing the corresponding queue length to zero. In the next section we discuss the stability of the index policy for a static cell.

5 Stability of the Index Policy

A desirable condition that should be satisfied by any policy implementable in practice is its stability, namely the queue lengths do not grow out of bounds with time. We discuss the stability of the IP in this section.We refer to the entire set of N inequalities given by $q_u \bar{R}_u > \lambda_u, u = 1, 2, \ldots, N$ as the Individual Stability Conditions (ISCS). Consider instead the Aggregate Stability Condition (ASC) given by

$$\sum_u \frac{\lambda_u}{\bar{R}_u} < 1. \tag{5.1}$$

The ASC is clearly less restrictive than the ISCS, because any q-policy that satisfies the ISCS will satisfy (5.1). The ASC is also more useful in real life applications than the ISCS because it doesn't depend on the choice of the initial q-policy (which can be arbitrary). However, since the index in

(4.27) depends on q-policy, we explore the connection between the ASC and the ISCS in the lemma below.

Lemma 5.1. *If the ASC (5.1) holds, $\exists$ an initial q-policy that satisfies the ISCS (4.6) for all $u \in \mathcal{A}$.*

Proof. Let $\kappa_u = \lambda_u / \bar{R}_u$. Then (5.1) implies

$$\sum_u \kappa_u < 1. \tag{5.2}$$

Consider the q-policy given by

$$q_u = \frac{\kappa_u}{\sum_u \kappa_u}. \tag{5.3}$$

Clearly this is a valid q-policy satisfying (4.2). Further, for all $u \in \mathcal{A}$, using (5.3)

$$q_u \bar{R}_u = \frac{\lambda_u}{\sum_u \kappa_u}, \tag{5.4}$$

which reduces to the ISCS (4.6) using equation 5.2. □

It should be noted here that under the assumption of stochastically identical users, κ_u is the same for every user and the initial q-policy (5.3) reduces to $q_u = 1/N$.

We now state the stability conditions and prove the stability of the IP in the next theorem. We use a well developed theory of stability using Lyapunov drift [4, 20].

Theorem 5.2. *If (5.1) holds, then there exists a $q = [q_1, q_2, \ldots, q_N]$ such that the IP using index in (4.27) is stable.*

Proof. Lemma 5.1 implies there exists a q given by (5.3) such that the ISCS (4.6) hold. Now we follow a technique similar to the one used in [1]. Define a Lyapunov function $G(Q)$ on the set of queue length vectors $Q \in Z^N$ as follows: $G(Q) = \sum_{u=1}^{N} \delta_u Q_u^2$. Then it is enough to prove that

$G(Q)$ has negative drift except in a finite set $\Lambda \in Z^N$ for some $\delta_u > 0$, $u \in \mathcal{A}$. Choose

$$\delta_u = \frac{K_u}{q_u \bar{R}_u - \lambda_u}. \tag{5.5}$$

Then from (4.6) $\delta_u > 0$ for $u = 1, 2, \ldots, N$. Recall that

$$v(n) \in \arg\max_u \delta_u \min(R_u^n, Q_u^n + A_u^n) \left[2Q_u^n + R_u^n\right]. \tag{5.6}$$

For $u = 1, 2, \ldots, N$ define

$$J_u^n = \min(R_u^n, Q_u^n + A_u^n)\mathbf{1}(u = v(n)), \tag{5.7}$$

where $\mathbf{1}(u = v(n)) = 1$ if $u = v(n)$ and 0 otherwise. Therefore, (3.1) implies

$$Q_u^{n+1} = Q_u^n + A_u^n - J_u^n. \tag{5.8}$$

Squaring (5.8), then adding and subtracting $R_u^n(A_u^n - J_u^n)$ on the Right Hand Side (RHS) of the resulting equation and finally multiplying both sides by δ_u yields

$$\begin{aligned}\delta_u(Q_u^{n+1})^2 - \delta_u(Q_u^n)^2 = \delta_u J_u^n(R_u^n + J_u^n) - \delta_u A_u^n(R_u^n + J_u^n) + \delta_u(A_u^n)^2 - \delta_u A_u^n J_u^n \\ + \delta_u(2Q_u^n + R_u^n)(A_u^n - J_u^n)\end{aligned} \tag{5.9}$$

Let $R^* = \max_u r_u$. Clearly $J_u^n \leq R^*$, and since $A_u^n, R_u^n, J_u^n \geq 0$, summing (5.9) over all $u \in \mathcal{A}$,

$$\sum_u \left[\delta_u(Q_u^{n+1})^2 - \delta_u(Q_u^n)^2\right] \leq \sum_u \delta_u \left[2(R^*)^2 + (A_u^n)^2\right] + \sum_u \left[\delta_u(2Q_u^n + R_u^n)(A_u^n - J_u^n)\right]. \tag{5.10}$$

Now let ζ_u^n be the number of packets served to user u and $v_q(n)$ be the user served in time slot n

under the randomized policy. Then

$$\begin{aligned}
\sum_u \delta_u(2Q_u^n + R_u^n)J_u^n &= \delta_{v(n)}(2Q_{v(n)}^n + R_{v(n)}^n)\min(R_{v(n)}^n, Q_{v(n)}^n + A_{v(n)}^n) && (5.11)\\
\sum_u \delta_u(2Q_u^n + R_u^n)\zeta_u^n &= \delta_{v_q(n)}(2Q_{v_q(n)}^n + R_{v_q(n)}^n)\min(R_{v_q(n)}^n, Q_{v_q(n)}^n + A_{v_q(n)}^n) && (5.12)\\
\sum_u \delta_u(2Q_u^n + R_u^n)J_u^n &\geq \sum_u \delta_u(2Q_u^n + R_u^n)\zeta_u^n, && (5.13)
\end{aligned}$$

where (5.13) follows from (5.11), (5.12), (5.6) and (5.7). Equation (5.13) implies

$$\begin{aligned}
\sum_u \left[\delta_u(Q_u^{n+1})^2 - \delta_u(Q_u^n)^2\right] &\leq \sum_u \delta_u\left[2(R^*)^2 + (A_u^n)^2\right] + \sum_u \left[\delta_u(2Q_u^n + R_u^n)(A_u^n - \zeta_u^n)\right]\\
&\leq \sum_u \delta_u\left[2(R^*)^2 + (A_u^n)^2 + R_u^n A_u^n\right] - \sum_u \left[\delta_u(2Q_u^n)(\zeta_u^n - A_u^n)\right]
\end{aligned} \tag{5.14}$$

Since $\{A_u^n, n \geq 0\}$ and $\{R_u^n, n \geq 0\}$ are independent of each other, $\mathrm{E}[\zeta_u^n] = q_u\bar{R}_u$ and $\mathrm{E}[A_u^n] = \lambda_u$, it follows from (5.14) that

$$\mathrm{E}\left[\sum_u \delta_u(Q_u^{n+1})^2 - \sum_u \delta_u(Q_u^n)^2 | Q^n\right] \leq \omega - 2\sum_u \left[Q_u^n\delta_u(q_u\bar{R}_u - \lambda_u)\right], \tag{5.15}$$

where the constant $\omega = 2(R^*)^2\sum_u \delta_u + \sum_u \delta\mathrm{E}(A_u^n)^2 + \sum_u \delta_u\bar{R}_u\lambda_u$. Furthermore, letting $K^\# = \min_u K_u$ and using (5.5) in (5.15) we have

$$\mathrm{E}\left[\sum_u \delta_u(Q_u^{n+1})^2 - \sum_u \delta_u(Q_u^n)^2 | Q^n\right] \leq \omega - 2K^\# \sum_u Q_u^n. \tag{5.16}$$

Thus we have proved that for any $\alpha > 0$, the expected drift $\mathrm{E}\left[\sum_u \delta_u(Q_u^{n+1})^2 - \sum_u \delta_u(Q_u^n)^2 | Q^n\right] < -\alpha$ except in the finite set Λ given by

$$\Lambda = \left\{Q^n \in Z^N : \sum_u Q_u^n \leq \frac{\omega + \alpha}{2K^\#}\right\}, \tag{5.17}$$

as required. □

Thus our approach is to use the initial policy (5.3) to compute our index policy (4.27) and the theorem above guarantees its stability. We further note that the stability region of this system is given by the region $\mathcal{S} = \{\lambda_u < \sum_{m=1}^{M} r_k \pi_m^u q_u^m, u = 1, 2, \ldots, N\}$ for some channel aware probability (depends on the channel state m of the user) q_u^m of serving user u [18]. It is easy to see that IP is stable in $\mathcal{S}$ as well. To see this, note that the proof above works out in this setting by replacing q_u with q_u^n that can depend on (the state in) time slot n. This will make δ also dependent on n. However, the only condition on δ needed in the proof is $\delta > 0$ which will be true in the stability region $\mathcal{S}$.

5.1 Implementation and Estimation Issues

The IP and UIP given by equations (4.27) and (4.28) respectively require estimates of $\bar{R}_u$ and λ_u for their implementation. We therefore recommend the use of an index with a coefficient different from the one used in AI (equation 4.24) and show how it is equivalent to using the AI itself, incorporating the optimality and stability properties of the AI. For $\delta_u > 0$, define Robust Index (RI) as:

$$I_u(i_u, y_u, a_u) = \delta_u(2y_u + r_{i_u}) \min(r_{i_u}, y_u + a_u). \tag{5.18}$$

For each user $u \in \{1, 2, \ldots, N\}$, the coefficient δ_u should be chosen depending on the priority (for service) to be given to that user - higher priority implying a higher value of δ_u. There are several advantages of using this index. Firstly, there is no need for any rate or arrival parameter estimates. Second, as long as $\delta_u > 0$, stability of queues is guaranteed according to Theorem 5.2. Finally, equating the two coefficients δ_u and $K_u/(q_u\bar{R}_u - \lambda_u)$, we see that using δ_u as the coefficient is

equivalent to using a q_u given by:

$$q_u = \frac{1}{\bar{R}_u}\left(\lambda_u + \frac{K_u}{\delta_u}\right) \tag{5.19}$$

Clearly, $q_u\bar{R}_u > \lambda_u$ implying that the ISCS hold and the system is stabilizable, as required. This also means that the ASC holds, i.e., $\sum_u \lambda_u/\bar{R}_u < 1$. Summing equation (5.19) over all u, we get $1 = \sum_u q_u = \sum_u \lambda_u/\bar{R}_u + \sum_u K_u/(\delta_u\bar{R}_u)$. Equivalently, $\sum_u K_u/(\delta_u\bar{R}_u) = 1 - \sum_u \lambda_u/\bar{R}_u$. Since the ASC holds, $1 - \sum_u \lambda_u/\bar{R}_u$ is a positive constant, say c. Hence for a given mobility and data arrival profile $(\bar{R}_u, \lambda_u)$, choosing a higher value of δ_u (to ensure $\sum_u K_u/(\delta_u\bar{R}_u)$ is a constant) is equivalent to increasing K_u, i.e., giving more priority to user u. Define the Robust Index Policy (RIP) as the policy serving the user $v_{RIP}(n)$ in slot n such that

$$\text{RIP}: \quad v_{RIP}(n) \in \arg\max_u \delta_u \min(R_u^n, Q_u^n + A_u^n)\,[2Q_u^n + R_u^n]\,. \tag{5.20}$$

The policy that we finally recommend for implementation is the RIP since it is equivalent to the IP: just like IP, it is an outcome of one step of policy improvisation of a randomized policy whose choice of q_u guarantees stabilizability. Its throughout optimality follows from Theorem 5.2 and thus we have a policy that is intuitive (increasing δ_u increases the service priority of a user), incorporates the optimality and stability properties of the IP as explained above and is robust, i.e., does not require any estimation as was implied in the use of IP. When users are identical, we recommend using $\delta_u = 1$ yielding the index of equation (4.25). We present results corresponding to this policy in section 7.2.

Thus to be implemented in real systems, first of all we need to compute R_u^n for each user $u = 1, 2, \ldots, N$ from its channel state X_u^n. As mentioned in the introduction, current systems already do that through the pilot signal, so the technology to do that is already developed and there is no additional design modification and effort required. Further, since even in current systems, base

stations do manage the data for all users, even Q_u^n is known without additional effort. Therefore, the proposed algorithm does not pose any additional technological or data management challenges to the existing system.

6 Performance Analysis

In this section we compare our algorithm to the MWA. First we show that the space and time complexity of the index policies and the MWA are the same. The state vector of a user at time n in the MWA algorithm is given by $[R_u^n, Q_u^n, A_u^n]$. The computation of the MWA index of (2.1) takes a constant time. The actual scheduling step involves a maxima over N entities, which is an $O(N)$ operation. In fact, it can be easily argued that there is a matching lower bound of $\Omega(N)$ because any algorithm must examine all of the inputs to be sure that it actually finds the maximum value. Thus the time complexity of the MWA is constant + $O(N)$ while space complexity is $3N$. Similarly, the time complexity of each of the index policy RIP too (as well as UIP and the more general, IP) is constant + $O(N)$ while space complexity is $3N$ *which are the same as that of the MWA*. Further, since the expected throughput $B = \sum_u \lambda_u$ for any stable policy, we concentrate on ξ = Long run expected sum of queue lengths of all users as the performance measure. We compare the RIP with MWA by comparing their achieved ξ values for a wide range of arriving packet load (λ_u). We assume that all the users have the same TPM P with limiting distribution $\pi = [\pi_1, \ldots \pi_M]$, and that $\{X^n : n \geq 0\}$ is aperiodic. Furthermore, we assume that $K_u = K$ for all $u \in 1, 2, \ldots, N$ yielding $\delta_u = 1$ for RIP.

7 Simulation Results

We first use simulation to compare the performance of the RIP with respect to the optimal on a small problem. We estimate ξ for the RIP and the optimal policy. Then, we use simulation to estimate ξ for the RIP and the MWA on a realistic problem of large size. The code for all simulation has been written using the C programming language. We use standard statistical estimators and refer to our earlier paper [7] for details.

7.1 Simulation Parameters

We run every path of the Markov Chain for 10^6 time slots and use a warmup period of $5 * 10^5$ time slots. We use the following TPM P of the channel state Markov Chain $\{X_u^n, n \geq 0\}$ for all users $u \in \mathcal{A}$:

$$P = \alpha I + \frac{1-\alpha}{M-1}(D - I),$$

where I is an $M \times M$ identity matrix and D is an $M \times M$ matrix with all entries equal to 1. This implies that the Markov Chain stays in a given state for $Geometric(1-\alpha)$ number of time slots and then moves to one of the remaining $M-1$ states with equal probability.

Due to curse of dimensionality, we can not compare our policy with the optimal policy for a full blown system with large number of users and channel states. However, just to get a flavor of the extent of non-optimality, albeit on a small problem, we present the following comparison before we launch into comparing our policy (RIP) with existing algorithms on a large example. In this illustrative example, we consider $N = 2$ users and $M = 2$ available data rates, i.e., $r = \{1, 2\}$. We assume a simple arrival process: for each user, in every slot, 1 packet arrives with a probability p and no packet arrives with probability $1 - p$. The optimal policy was derived using a matlab code. The queue-length of each user increases at a very fast rate with increase in p, so we present

the comparative performance (with respect to the optimal) of the RIP for $p \in \{0.1, 0.2, 0.3\}$. The computational effort required for higher values of p becomes prohibitively large, since the buffer sizes (and consequently the size of the state space) required to hold the queues blow up. In table 1, we present simulation results comparing the ξ of RIP with ξ of the optimal policy.

[Table 1 about here.]

Now, to compare the RIP with existing algorithms, we consider a realistic scenario with $M = 11$ data rates. We use the following set of available data rates (kbps) [6]: r ={1, 2, 3, 4, 5, 8, 16, 24, 32, 48, 64}. Since P is doubly stochastic, $\pi_k = 1/M = 1/11$ yielding $\bar{R}_u = 18.8$ for all $u = 1, 2, \ldots, N$. Further, the length of a time slot is $1.67 * 10^{-3}$ seconds. We choose $\alpha = 0.9999$ implying that on the average the Markov Chain stays in one state for 10^4 time slots, i.e., 16.7 seconds, before changing states.

7.2 Constant Number of Users

We use $L = 100$ sample paths of the Markov Chain $\{X_u^n, n \geq 0\}$. Since all users are assumed stochastically identical, we use the same $\lambda_u = \lambda^p$ for all $u = 1, 2, \ldots, N$. In the notation λ^p we use the subscript p to indicate the mean arrival rate of packets. We use λ in sections 4.4 and 7.3 to denote the arrival rate of users themselves. We vary λ^p from 0.1 to 1.7 and report the results in table 2. We use 1.7 as the maximum value of λ^p for reporting results, since there are 10 users and the total average departure rate is $\bar{R}_u = 18.8$.

[Table 2 about here.]

In table 2, ξ-RIP, ξ-MWA and ξ-EXP represent the total queue lengths under RIP, MWA and the exponential rule respectively. The Imp-MWA column represents the absolute improvement of RIP over MWA (= ξ-MWA - ξ-RIP) followed by percentage improvement (reported in parentheses, computed using % Imp = 100*Imp-MWA/ξ-MWA). Similarly Imp-EXP column displays the

corresponding improvement of RIP over the exponential rule. From the Table 2 we see that the RIP performs better (lower ξ) than the MWA across the entire range of λ^p from 0.1 (low traffic) to 1.7 (high traffic). The % Imp is more for lower values of λ^p. The similar performance of RIP and MWA for higher λ^p is expected because in heavy traffic $\min(r_{i_u}, y_u + a_u) = r_{i_u}$ for most time slots. Thus the index given by (4.25) becomes similar to the MWA index given by (2.1). It is worth noting, however, that although the % Imp decreases with increase in λ^p (except from 0.1 to 0.25), the absolute improvement increases monotonically with λ^p. As expected, the RIP performs significantly better than the Exponential Rule because the RIP is designed to minimize ξ whereas the Exponential Rule minimizes the maximum steady state (weighted) queue length among all users.

The above results come from high warmup period, and hence seem difficult to apply in real life settings. Further, for the algorithm to be useful, it should work for various values of α corresponding to various mobility scenarios. For instance, $\alpha = 0.9999$ implies an average of 10,000 slots between state change, while $\alpha = 0.999$ implies 1000 and $\alpha = 0.99$ implies 100 slots. We perform simulations for a much lower value of warmup, i.e., 5000 and $\alpha = 0.999, 0.99$ and observe that results look similar. We report the results in table 3 below:

[Table 3 about here.]

7.3 Poisson Arrival of Users

The user dyanmics is as given in section 4.4. We choose $a = 1$ minute and $N_{\text{avg}} = 10$.

We use $L = 100$, $T = 5 * 10^5$ (10^6 total time slots and $5 * 10^5$ warmup slots). We report the results in Table 4 using the same format as that for Table 2.

[Table 4 about here.]

It is clear from the Table 4 that the best improvement for UIP, close to 25 %, corresponds to $\lambda^p = 0.1$. The results follow a similar trend as the static cell. As in the static cell, the index of

the MWA and the UI are similar in the high traffic regime yielding similar results. Even for the dynamic cell, we have performed simulation for 5000 warmup slots and various values of α. As in static cell, these results have relative trends (across different policies) that are similar to the base mobility scenario ($\alpha = 0.9999$) of table 4; hence we omit them here to keep the exposition from being repetitive.

8 Conclusion

In this paper we have used the MDP formulation and the PIA to develop an Index policy for the data transfer problem in a single cell of a wireless telecommunication network. Our main contribution is the development of an explicit optimization based approach (using an MDP model, and use of Policy Improvement Algorithm in a creative way) that yields the recommended policy as an outcome. Exact computation of the index is intractable, so we use appropriate approximations to yield a simple, intuitive, closed form index that is somewhat similar to the MWA in implementation, and slightly better in performance over the existing MWA. The approximation 4.23 tends to reduce this performance improvement, and better approximations for 4.23 are guaranteed to result in more performance improvement. However, while we are working on it now, it is beyond the scope of this paper whose main purpose is to demonstrate that an MDP-based explicit optimization approach can yield well performing and easy to implement scheduling policies. The advantage of this approach, other than those that follow from it being based on a systematic optimization procedure, is that it can accommodate a variety of scenarios, for example different user types, giving more or less weight to the queuing of data packets (by changing the penalty function in the one step reward from linear to something else), incorporating additional measures of rewards for service to different types of users and so on. Moreover, the framework can be extended to channel (condition) aware scheduling by having the initial policy (the q policy, i.e. the initial policy of the policy improvement

algorithm) depend on the state.

Future extensions of this work can include developing effective policies under the setting of a non-Markovian evolution of the state of the users, using admission control to limit the number of arriving users to maintain target performance. The work can also be extended to evaluate the performance of the IP when the parameters λ^p and the TPM P are no longer assumed to be known, but estimated dynamically. This extension will get the IP really close to being implementable in real life situations where none of the packet arrival and departure parameters are known.

APPENDIX

Proof of Theorem 4.1 Let g_q, $w_q(i,y,a)$, $\{g_{q_u}, u = 1,2,\ldots,N\}$ and $\{w_{q_u}(i_u,y_u,a_u), u = 1,2,\ldots,N\}$ satisfy (4.3), (4.10) and (4.11). Using (4.8) in the RHS of (4.11) and rearranging, we have

$$\sum_{u=1}^{N} g_{q_u} + \sum_{u=1}^{N} w_{q_u}(i_u,y_u,a_u) = \sum_{u=1}^{N} \left[q_u K_u(y_u + a_u - r_{i_u})^+ + (1-q_u)(K_u(y_u+a_u)\right]$$
$$+ \sum_{u=1}^{N} q_u \sum_{(j_u,b_u)} p^u_{i_u j_u} h^u_{a_u b_u} w_{q_u}(j_u, (y_u + a_u - r_{i_u})^+, b_u) + \sum_{u=1}^{N}(1-q_u) \sum_{(j_u,b_u)} p^u_{i_u j_u} h^u_{a_u b_u} w_{q_u}(j_u, y_u + a_u, b_u). \tag{.1}$$

Algebraic rearrangement of some terms yields

$$\begin{aligned} \sum_{u=1}^{N}(1-q_u) \sum_{(j_u,b_u)} p^u_{i_u j_u} h^u_{a_u b_u} w_{q_u}(j_u, y_u + a_u, b_u) &= \sum_{u=1}^{N} q_u \sum_{l \neq u} \sum_{(j_l,b_l)} p^l_{i_l j_l} h^l_{a_l b_l} w_{q_l}(j_l, y_l + a_l, b_l), \\ \sum_{u=1}^{N}(1-q_u)\{K_u(y_u+a_u)\} &= \sum_{u=1}^{N} q_u \sum_{l \neq u} \{K_l(y_l + a_l)\}. \end{aligned} \tag{.2}$$

From (.1) and (.2), and using (3.2) for the definition of $W_u(i, y+a)$, we have

$$\sum_{u=1}^{N} g_{q_u} + \sum_{u=1}^{N} w_{q_u}(i_u, y_u, a_u) = \sum_{u=1}^{N} [q_u W_u(i, y+a)]$$
$$+ \sum_{u=1}^{N} q_u [(\sum_{(j_u, b_u)} p^u_{i_u j_u} h^u_{a_u b_u} w_{q_u}(j_u, (y_u + a_u - r_{i_u})^+, b_u)) + (\sum_{l \neq u} \sum_{(j_l, b_l)} p^l_{i_l j_l} h^l_{a_l b_l} w_{q_l}(j_l, y_l + a_l, b_l))] \tag{.3}$$

Since $\forall k, m \in \mathcal{A}, k \neq m$ $w_{q_m}(\cdot, \cdot, \cdot)$ is independent of $p^k_{i_k j_k} h^k_{a_k b_k}$, and from (3.3) for the definitions of p_{ij} and h_{ab}, and (3.4) for the definition of $(y+a, i)^u$, (.3) reduces to

$$\sum_{u=1}^{N} g_{q_u} + \sum_{u=1}^{N} w_{q_u}(i_u, y_u, a_u) = \sum_{u=1}^{N} q_u W_u(i, y+a) + \sum_{u=1}^{N} q_u \sum_{j,b} p_{ij} h_{ab} w_q(j, (y+a, i)^u, b), \tag{.4}$$

which yields (4.3) by setting

$$g_{q_u} = q_u K_u(y_u + a_u - r_{i_u})^+ + (1 - q_u)[K_u(y_u + a_u)] \quad \text{and}$$
$$g_q = \sum_{u=1}^{N} g_{q_u}, \quad w_q(i, y, a) = \sum_{u=1}^{N} w_{q_u}(i_u, y_u, a_u),$$

as required.

Proof of Theorem 4.2 From (4.11),

$$\begin{aligned} w^{\Delta}_q(j, y, (y+a, i)^u, b) &= \sum_l w_{q_l}(i_l, (y+a, i)^u_l, a_l) - \sum_l w_{q_l}(i_l, y_l + a_l, a_l) \\ &= w_{q_u}(i_u, (y_u + a_u - r_{i_u})^+, a_u) - w_{q_u}(i_u, y_u + a_u, a_u) \end{aligned} \tag{.5}$$

Therefore, we only need to consider user u for computing $w^{\Delta}_q(j, y, (y+a, i)^u, b)$. Now consider path 1 and path 2. We plot the queue lengths of user u along these paths in figure 8. Recall that $Z_u(\cdot, \cdot, \cdot)$ and $T_u(\cdot, \cdot, \cdot)$ are defined by equations (4.13) and (4.14) respectively. We define Z^m_u, $m = 1, 2$ to be

the first time when the queue length of user u in path m goes to zero. Then

$$Z_u^1 = Z_u(j_u, (y_u + a_u - r_{i_u})^+, a_u), \quad Z_u^2 = Z_u(j_u, y_u + a_u, a_u), \tag{.6}$$

$$T_u^1 = T_u(j_u, (y_u + a_u - r_{i_u})^+, a_u), \quad T_u^2 = T_u(j_u, y_u + a_u, a_u). \tag{.7}$$

[Figure 1 about here.]

Let C_n^m be the cost incurred by user u along path m in time slot n and let $C_n = C_n^1 - C_n^2$ be the corresponding difference in costs. Then from equation .5,

$$w_q^{\Delta}(j, y, (y + a, i)^u, b) = E\left(\sum_{n \geq 0} [C_n]\right). \tag{.8}$$

It is easy to see from the coupling of paths in (4.15) and (4.17) and figure 8 that

$$C_n = 0, \quad n \geq Z_u^2 \tag{.9}$$

Using (.5) through (.7) and (.9) in (.8)

$$w_q^{\Delta}(j, y, (y + a, i)^u, b) = E\left(\sum_{n < Z_u^2} C_n\right) \tag{.10}$$

Clearly,

$$C_n = K_u(Q_u^{n,1} - Q_u^{n,2}). \tag{.11}$$

Consider the queue lengths $Q_u^{n,1}$ and $Q_u^{n,2}$ in paths 1 and 2 respectively. Both get the same number of data packets in every time slot and serve the same number of packets to user u whenever enough data is available in both the queues. Thus for $n \leq Z_u^1$ the difference $Q_u^{n,2} - Q_u^{n,1}$ remains the same as $Q_u^{0,2} - Q_u^{0,1}$. For $Z_u^2 > n > Z_u^1$, the difference $Q_u^{n,2} - Q_u^{n,1}$ remains the same as

$Q_u^{n-1,2} - Q_u^{n-1,1}$ when either user u is not served or user u is served and $Q_u^{n,1} \geq r_{X_u^{n,1}}$, $Q_u^{n,2} \geq r_{X_u^{n,2}}$. Also $Q_u^{n,2} - Q_u^{n,1} < Q_u^{n-1,2} - Q_u^{n-1,1}$ if $Q_u^{n,1} < r_{X_u^{n,1}}$. Thus as indicated in figure 8,

$$Q_u^{n,2} - Q_u^{n,1} = \min(r_{i_u}, y_u + a_u), \quad n \leq Z_u^1, \tag{.12}$$

$$Q_u^{n,2} - Q_u^{n,1} < \min(r_{i_u}, y_u + a_u), \quad Z_u^2 > n > Z_u^1. \tag{.13}$$

Using (.11), (.12) and (.13) in (.10), and the definitions (4.14), (.6) and (.7) of T_u^1 and T_u^2 respectively yields (4.18), as required.

Proof of Theorem 4.3 Using (3.5) for the definition of $W_u(i, y + a)$ and (4.19) we have

$$I_u'''(i, y, a) = -K_u \min(r_{i_u}, y_u + a_u) + \sum_{(j,b)} p_{ij} h_{ab} \left[w_q(j, (y + a, i)^u, b) - w_q(j, y + a, b) \right] \tag{.14}$$

Thus (4.18) and (.14) yield,

$$-K_u \min(r_{i_u}, y_u + a_u)[1 + \sum_{(j_u,b_u)} p_{i_u j_u}^u h_{a_u b_u} T_u(j_u, (y_u + a_u - r_{i_u})^+, a_u)] \geq I_u'''(i, y, a) \tag{.15}$$

$$-K_u \min(r_{i_u}, y_u + a_u)[1 + \sum_{(j_u,b_u)} p_{i_u j_u}^u h_{a_u b_u} T_u(j_u, y_u + a_u, a_u)] \leq I_u'''(i, y, a). \tag{.16}$$

Further, we know that $\{(X_u^n, A_u^n), n \geq 0\}$ is a DTMC with state space $\Omega \times Z$ because both the components $\{X_u^n, n \geq 0\}$ and $\{A_u^n, n \geq 0\}$ are independent DTMC's with P^u and Q^u as the TPM's respectively. Therefore using standard DTMC theory [11]

$$T_u(i_u, y_u, a_u) = 1 + \sum_{(j_u,b_u)} p_{i_u j_u}^u h_{a_u b_u} T_u(j_u, (y_u + a_u - r_{i_u})^+, a_u) \tag{.17}$$

$$T_u(i_u, y_u + r_{i_u}, a_u) = 1 + \sum_{(j_u,b_u)} p_{i_u j_u}^u h_{a_u b_u} T_u(j_u, y_u + a_u, a_u). \tag{.18}$$

Using (.17) in (.15) and (.18) in (.16) yields (4.21) and (4.20), as required.

References

[1] M. Andrews. A survey of scheduling theory in wireless data networks. *IMA Volumes in Mathematics and its applications*, 143:1–18, 1999.

[2] M. Andrews, K. Jung, and A. Stoylar. Stability of the max-weight routing and scheduling protocol in dynamic networks and at critical loads abstract, 2007.

[3] M. Andrews, K. Kumaran, K. Ramanan, A.L. Stolyar, R. Vijayakumar, and P. Whiting. Scheduling in a queueing system with asynchronously varying service rates. *Probability in the Engineering and Informational Sciences*, 18:191–217, 2004.

[4] S. Asmussen. *Applied Probability and Queues*. John Wiley & Sons, Inc, 1987.

[5] B. Awerbuch and T. Leighton. A simple local-control approximation algorithm for multi-commodity flow. In *Proceedings of the 34th Annual Symposium on Foundations of Computer Science*, pages 459 – 468, 1993.

[6] P. Bender, P. Black, M. Grob, R. Padovani, N. Sindhushayana, and A. Viterbi. A bandwidth efficient high speed data service for nomadic users. *IEEE Communications Magazine*, 38:70–77, July 2000.

[7] N. Bolia and V. Kulkarni. Index policies for resource allocation in wireless networks. *IEEE Transactions on Vehicular Technology*, 58:1823–1835, 2009.

[8] G. D. Celik, Long B. Le, and E. Modiano. Scheduling in parallel queues with randomly varying connectivity and switchover delay. In *Proc. IEEE INFOCOM'11*, April 2011.

[9] A. Eryilmaz and R. Srikant. Fair resource allocation in wireless networks using queue-length based scheduling and congestion control. In *Proceedings of IEEE INFOCOM '05*, pages 1794–1803, 2005.

[10] K. Glazebrook, J. Nino-Mora, and P.S. Ansell. Index policies for a class of discounted restless bandits. *Advances in Applied Probability*, 34:754–774, December 2002.

[11] V. Kulkarni. *Modeling and Analysis of Stochastic Systems*. Chapman & Hall, Inc, New York, USA, 1995.

[12] S. Liu, Lei Ying, and R. Srikant. Throughput-optimal opportunistic scheduling in the presence of flow-level dynamics. *IEEE/ACM Transactions on Networking*, 19:1–15, August 2011.

[13] N. W. McKeown, V. Anantharam, and J. Walrand. Achieving 100% throughput in an input-queued switch. In *Proceedings of IEEE INFOCOM '02*, pages 1451–1460, 2002.

[14] M. J. Neely, E. Modiano, and C. E. Rohrs. Power and server allocation in a multi-beam satellite with time varying channels. In *Proceedings of IEEE INFOCOM '02*, pages 1451–1460, 2002.

[15] M. F. Neuts. *Matrix-Geometric Solutions in Stochastic Models - An Algorithmic Approach*. The Johns Hopkins University Press, 1981.

[16] M. Opp, K. Glazebrook, and V. Kulkarni. Outsourcing warranty repairs: Dynamic allocation. *Naval Research Logistics Quarterly*, 52:381–398, December 2005.

[17] M. Puterman. *Markov Decision Processes - Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc, New York, USA, 1994.

[18] S. Shakkottai and A. L. Stolyar. Scheduling for multiple flows sharing a time-varying channel: The exponential rule. *Analytic Methods in Applied Probability. In Memory of Fridrih Karpelevich. Yu. M. Suhov, Editor*, 207:185–202, 2002.

[19] S. Shakkottai, A. L. Stolyar, and R. Srikant. Pathwise optimality of the exponential scheduling rule for wireless channels. *Advances in Applied Probability*, 36:1021–1045, 2004.

[20] L. Tassiulas and A. Ephremides. Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. *IEEE Transactions on Automatic Control*, 37:1936–1948, December 1992.

[21] R. Weber. On the gittins index for multiarmed bandits. *The Annals of Applied Probability*, 2:1024–1033, November 1992.

[22] P. Whittle. Multi-armed bandits and the gittins index. *Journal of the Royal Statistical Society*, 42:143–149, 1980.

List of Figures

1 Queue lengths for user u along paths 1 and 2 until both hit zero 36

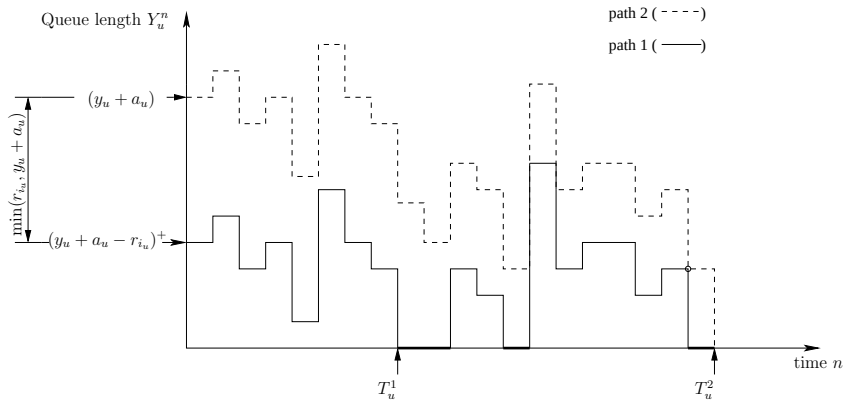

Figure 1: Queue lengths for user u along paths 1 and 2 for $0 \leq n \leq Z_u^2$. The queue length remains the same during a time slot. The difference $Q_u^{n,2} - Q_u^{n,1}$ remains equal to $\min(r_{i_u}, y_u + a_u)$ for $0 \leq n \leq Z_u^1$ because equal amount of data can be served along both paths. For $Z_u^2 > n > Z_u^1$, $0 \leq Q_u^{n,2} - Q_u^{n,1} \leq \min(r_{i_u}, y_u + a_u)$. The difference $Q_u^{n,2} - Q_u^{n,1}$ goes down in every slot in which there isn't enough data to serve for user u along path 1. For $n \geq Z_u^2$, $Q_u^{n,2} = Q_u^{n,1}$ and are therefore not shown in the figure.

List of Tables

1 Comparison of the RIP with the Optimal Policy cell 38
2 Performance of the index policy, the MWA and the Exponential Rule in the static cell 39
3 Performance of the index policy, the MWA and the Exponential Rule in the static cell 40
4 Performance of the index policy, the MWA and the Exponential Rule in the dynamic cell 41

p	ξ-optimal	ξ-RIP	% Difference
0.1	0.11	0.12	9.1
0.2	0.24	0.28	16.7
0.3	0.44	0.52	18.2

Table 1: Performance of the RIP, the optimal policy in the static cell: ξ for RIP, optimal policy and percentage increase of ξ for RIP with respect to the optimal policy.

λ^p	ξ-RIP	ξ-MWA	ξ-EXP	Imp-MWA	Imp-EXP
0.1	1.77	1.98	11.98	0.21 (10.61)	10.21 (85.22)
0.5	6320.02	6427.75	32023.15	107.73 (1.68)	25703.13 (80.26)
0.75	26245.1	26385.78	236538.67	140.68 (0.53)	210293.57 (88.90)
1.0	59211.02	59384.51	536662.19	173.49 (0.29)	477451.17 (88.97)
1.1	78763.46	78945.77	698849.4	182.31 (0.23)	620085.94 (88.73)
1.25	105715.01	105906.88	948331.77	191.87 (0.18)	842616.76 (88.85)
1.5	158305.29	158503.72	1438472.14	198.43 (0.13)	1280166.85 (89.0)
1.6	186283.35	186486.07	1629874.98	202.72 (0.11)	1443591.63 (88.57)
1.7	209638.7	209847.32	1840995.63	208.62 (0.10)	1631356.93 (88.61)

Table 2: Performance of the index policy, the MWA, the Exponential Rule in the static cell: ξ for RIP, MWA, Exponential Rule; absolute improvement and percentage improvement of RIP with respect to the MWA and the Exponential Rule.

λ^p	ξ-RIP	ξ-MWA	ξ-EXP	Imp-MWA	Imp-EXP
—	$\alpha = 0.9999$				—
0.1	1.76	1.97	11.99	0.21 (10.66)	10.23 (85.32)
1.0	57594.51	57765.83	475768.71	171.32 (0.3)	418174.2 (87.89)
1.5	152933.76	153131.43	1197014.63	197.67 (0.13)	1044080.87 (87.22)
1.7	205557.48	205760.48	1501240.82	203 (0.10)	1295683.34 (86.31)
—	$\alpha = 0.999$				—
0.1	1.76	1.85	11.94	0.09 (4.86)	10.18 (85.26)
1.0	5885.2	5988.86	54137.12	103.66 (1.73)	48251.92 (89.13)
1.5	15662.01	15852.29	146678.39	190.28 (1.2)	131016.38 (89.32)
1.7	20970.08	21172.55	194130.23	202.47 (0.96)	173160.15 (89.2)
—	$\alpha = 0.99$				—
0.1	1.7	1.85	10.63	0.15 (8.11)	8.93 (84.01)
1.0	658.91	698.38	4805.1	39.47 (5.65)	4146.19 (86.29)
1.5	1509.1	1565.18	14082.92	56.08 (3.58)	12573.82 (89.28)
1.7	1981.7	2050.84	19040.07	69.14 (3.37)	17058.37 (89.59)

Table 3: Performance of various policies for 5000 warmup slots and different mobility scenarios (different α values). The results are similar to those of table 2.

λ^p	ξ-UIP	ξ-MWA	ξ-EXP	Imp-MWA	Imp-EXP
0.1	14.26	19.06	322989908.9	4.8 (25.18)	322989894.6 (100.0)
0.5	5805.52	5897.56	1861730045.5	92.04 (1.56)	1861724239.98 (100.0)
0.75	17905.93	18010.79	2990814203.07	104.86 (0.58)	2990796297.14 (100.0)
1	37271.5	37393.96	4295218011.30	122.46 (0.33)	4294880739.80 (100.0)
1.1	46920.62	47048.08	4668219754.69	127.46 (0.27)	4668172834.07 (100.0)
1.25	62684.45	62812.31	5316113575.45	127.86 (0.20)	5316050891.00 (100.0)
1.5	84970.78	85090.1	7291982154.77	119.32 (0.14)	7291897183.99 (100.0)
1.6	103527.23	103654.22	7403769958.26	126.99 (0.12)	7403666431.03 (100.0)
1.7	109686.71	109812.72	8156028950.81	126.01 (0.11)	8155919264.10 (100.0)

Table 4: Performance of the index policy, the MWA and the Exponential Rule in the dynamic cell: ξ for UIP, MWA, Exponential Rule and Improvement (absolute and percentage) of UIP with respect to the MWA and the Exponential Rule.