PAPER

# Optimal Buffer Partitioning on a Multiuser Wireless Link*

Ömur ÖZEL†, *Nonmember*, Elif UYSAL-BIYIKOGLU††a), *Member*, *and* Tolga GIRICI†††, *Nonmember*

**SUMMARY** A finite buffer shared by multiple packet queues is considered. Partitioning the buffer to maximize total throughput is formulated as a resource allocation problem, the solution is shown to be achieved by a greedy incremental algorithm in polynomial time. The optimal buffer allocation strategy is applied to different models for a wireless downlink. First, a set of parallel $M/M/1/m_i$ queues, corresponding to a downlink with orthogonal channels is considered. It is verified that at high load, optimal buffer partitioning can boost the throughput significantly with respect to complete sharing of the buffer. Next, the problem of optimal combined buffer allocation and channel assignment problems are shown to be *separable* in an outage scenario. Motivated by this observation, buffer allocation is considered in a system where users need to be multiplexed and scheduled based on channel state. It is observed that under finite buffers in the high load regime, scheduling simply with respect to channel state with a simply partitioned buffer achieves comparable throughput to combined channel and queue-aware scheduling.

***key words:*** *buffer partitioning, multiuser wireless communication, throughput optimal, finite buffer, complete sharing, complete partitioning, greedy allocation, scheduling, MaxWeight*

## 1. Introduction

Memory is a limited resource in communication devices. While communication, computation and memory capabilities continuously increase, with the advance of standards and systems such as 3G and broadband wireless MAN, there is also a substantial increase in the demand for bandwidth and memory. For example, a typical WiMax base station is supposed to serve a metropolitan area with hundreds of users demanding high speed multimedia applications. With a limited memory space, buffer management is necessary for maximum performance in such a multiuser system.

Sharing limited buffer space among multiple packet streams is a problem that previously attracted interest in the context of shared-memory switches [1] and wireline networks [2]. The two opposite extremes of buffer management are Complete Sharing (CS) and Complete Partitioning (CP). In Complete Sharing, packets that arrive are placed in the buffer as long as there is room, regardless of which session they belong to; whereas in CP, the buffer is divided into disjoint partitions dedicated to each active session. CS possesses a degree of flexibility, and can under some conditions achieve higher utilization of the buffer. However, it has the drawback that a high-rate session, or one which is highly bursty, could completely occupy the memory space, causing low-rate sessions to suffer packet drops, or be dropped altogether (for example, if they have delay constraints).

Another drawback of a CS architecture specific to a shared wireless link is the potential loss of multiuser diversity. Exploiting multiuser diversity, i.e., the increasing probability of finding good channels as the number of users increases [3] requires the base station to have packets to transmit to each user [4]. When some sessions "hog" the buffer, blocking others, potentially the full multiuser channel capacity region cannot be used, thus limiting throughput. Partitioning the buffer presents a sure remedy to the "hogging" problem, as it does not let users enter each other's space. While there may be obvious drawbacks of partitioning as well, such as its inflexibility, it performs extremely well in the high-load regime [1], which is the motivation for this work.

A multiuser wireless downlink may work in the overloaded regime for several reasons. Such a system typically serves various uncoordinated users, as in fixed wireless [5] Internet access, as well as in cellular systems. It is to be expected that sessions initiated by various user applications do not have correct estimates of the transmission rate available to them, as the total number of sessions is dynamic, as well as the channel itself. Under such uncertainties, operating close to instability may be preferable to occasionally idling and not fully utilizing the tight wireless resource, as consequent packet drops may be tolerated by higher-layer mechanisms (such as TCP). That is, perhaps the unstable regime is a practical reality in wireless systems.

While higher layer mechanisms can adjust arrival rate for stable data transmission, they do not obviate the need to address the overloaded regime because their response times are typically much longer than coherence times of outdoor channels [6], [7], and the system could easily become overloaded between congestion window updates.

Hence, we claim that optimal buffer partitioning can be used together with higher layer mechanisms in order to better utilize wireless resources. As an example, consider the situation depicted in Fig. 1 where the last hop along the net-

Manuscript received July 26, 2010.
Manuscript revised April 23, 2011.

†The author is with the Dept. of Electrical and Computer Engineering, University of Maryland, College Park, 20742 USA.
††The author is with the Department of Electrical and Electronics Engineering, Middle East Technical University, 06531 Ankara, Turkey.
†††The author is with the Department of Electrical and Electronics Engineering, TOBB University of Economics and Technology, 06560 Ankara, Turkey.

*This paper was presented at the Information Theory and Applications Workshop at the University of California San Diego on February 1, 2010.

a) E-mail: elif@eee.metu.edu.tr

DOI: 10.1587/transcom.E94.B.3399

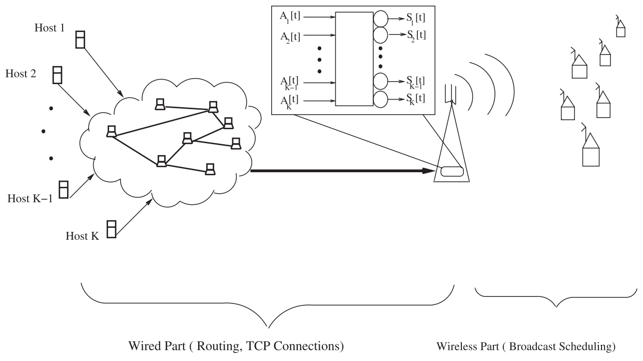

**Fig. 1** End-to-end network connection of multiple users with a shared wireless last hop.

work routing path is wireless. The buffers at the wireless transmitter will need to have a sufficient number of packets to be able to exploit multiuser diversity and operate at a timescale determined by the state of wireless channel. The queue lengths here could be capped at the optimal partitioning levels. The TCP's that work end to end could be responsible for satisfying a long-term rate requirement to ensure that the right number of packets is maintained. The buffer partitioning problem also reveals a trade-off between buffer utilization and multiuser diversity and the tradeoff between giving individual throughput guarantees to low rate users and maximizing overall throughput.

Buffer partitioning can also be performed jointly with user scheduling. This more general problem of optimal joint buffer management and scheduling under finite memory is still open. It has been shown [8] that maximum weight matching between queue lengths and channel rates at any time (in short, MaxWeight (see [9], for example.)), which is a well known throughput optimal algorithm under infinite buffers, is *not optimal* under finite buffers. Though not necessarily optimal, MaxWeight provides a solid benchmark for the throughput of a finite buffer system. Note that MaxWeight requires making rate allocation decisions based on joint queue and channel state information, hence is inherently *cross-layer*. We believe that being able to separate the rate allocation problem from the buffer management problem carries practical value, as it can be cumbersome to keep physical layer algorithms informed about queue state. The search in this direction is clearly encouraged by suboptimality of MaxWeight.

### 1.1 Related Work

The work on buffer management in the literature mostly focused on shared memory switches in wired networks. The main problem is finding the buffer occupancy threshold, above which the new arrivals are dropped. For example in [1], Irland computationally finds the optimal buffer sharing policy, that finds a simple threshold rule, which performs close to optimal. Kamoun and Kleinrock [2] defined some hybrid schemes in addition to complete sharing and partitioning. These schemes provide the minimum number of dedicated buffers and/or determine a maximum instantaneous occupancy limit for each session. Simulation results indicate that as the load increases the optimal allocation converges to a complete partitioning. Foschini and Gopinath [10] analytically determine the structure of the optimal sharing policies. The optimal policy involves limiting the buffer occupancy and dedicating some buffer space for each session. Krishnan et al. [11] propose a dynamic buffer partitioning mechanism, which can be difficult to implement in practice. Optimum scheduling and memory management with finite buffer space was studied in [8]. A closed form optimal scheduling policy was found for $2 \times 2$ switches with equal arrival rates [8]. The policy involves *push-out*, where an existing packet is discarded in favor of a new arrival, which may be difficult from an implementation perspective.

To the best of our knowledge, the buffer partitioning problem has not been previously addressed in the context of wireless networks. A related idea of modifying the Transport Control Protocol for exploiting multiuser diversity was presented by Andrew et al. [12].

### 1.2 Contributions

This paper mainly asks two questions: (1) Given a finite buffer, how should we partition it among users with given arrival and service rates to maximize total throughput? (2) How good is throughput performance in a multiuser wireless link if the buffer is simply partitioned and then scheduling is done *without regard to queue state*? In answer to the first question, an optimal iterative algorithm for allocating buffer space among an array of $(G/G/1/m)$ queues which uses the average drop probability expression as a function of the number of buffers, $m$, is derived. The uniqueness of the resulting throughput maximizing buffer distribution is shown. The second question is explored under different wireless communication scenarios. We first consider the problem of allocating to a set of users a set of orthogonal channels with occasional outage. We obtain the encouraging result that the problems of channel and buffer allocation are separable in this case. Next, a probabilistic scheduling policy proposed for a two-user model with on-off channels is optimized and shown to be also separable. Simulation results indicate that the proposed policy achieves a throughput performance close to that of MaxWeight. We then consider a more realistic downlink multiuser system, where N independent packet arrival processes are separately queued to be sent by a single transmitter over a wireless channel, whose state evolves according to a stationary stochastic process. The service model depends on how the data streams are multiplexed to be transmitted. We compare dynamic scheduling based solely on channel state (specifically, selecting the user with the strongest channel on each scheduling decision) with MaxWeight, and see that the throughput performance gap is small. We start by presenting the basic buffer partitioning problem in the next section.

## 2. Buffer Partitioning

In a system of $N$ users sharing a total buffer pool of size $B$, the set of feasible buffer partitions, $\Psi$, is defined as the following:

$$\Psi = \left\{ \mathbf{m} = (m_1, m_2, ..., m_N),\ m_i \in \mathbb{Z}_+ : \sum_{i=1}^{N} m_i \leq B \right\}$$

Accordingly, an arriving packet of user $i$ is accepted if there are less than $m_i$ packets belonging to user $i$ in the queue, otherwise, it is blocked[†].

Partitioning is not necessarily throughput-optimal. In fact, a dynamic allocation of buffer space among queues according to a coordinate-convex policy where $\sum_{i=1}^{N} m_i > B$ (that is, users are allowed to spill over to each other's allocation) may result in higher throughput [2], [10]. There are also push-out type of policies [13] where an existing packet in the queue can be dropped in case of arrival of another packet. However, partitioning was observed to perform very well (and is perhaps optimal) for unbalanced and high loads [2]. It is shown in [10] that optimal policy for two user balanced high load case is equally partitioning the available buffer for users. Benefit of buffer partitioning for unbalanced load are also discussed in [11], [14] under different data and flow models. The rest of the paper will be about optimal partitioning and its joint application with scheduling in a number of scenarios.

### 2.1 Maximizing Total Throughput under Buffer Partitioning

Our optimization rests on the concavity and monotonicity of throughput with respect to both arrival rate and buffer space in an M/G/1/m system [15], under a fixed service time distribution. Consider a set of queues $\{i, 1 \leq i \leq N\}$ that work in parallel. Let $T(\lambda_i, m)$ be the throughput of the $i$th queue, with arrival rate $\lambda_i$ when a waiting room of $m$ packets is allocated to this queue. In the rest, we use the shorthand $T_i(m)$ to mean $T(\lambda_i, m)$. We denote by $\Delta T_i(m)$ the increase in throughput that would result from increasing the buffer space in queue $i$ to $m + 1$.

$$\Delta T_i(m) = T_i(m+1) - T_i(m) \tag{1}$$

Increasing the waiting room always increases the throughput [15], [16], so $\Delta T_i(m) > 0$. But, concavity implies diminishing returns, i.e $\Delta T_i(m+1) < \Delta T_i(m)\ \forall m$.

The buffer allocation that maximizes total throughput is a solution to the following optimization problem:

**Problem 1:**

$$\max \sum_{i=1}^{N} T_i(m_i)\ s.t.\ \mathbf{m} \in \Psi \tag{2}$$

We now present an iterative algorithm for calculating the optimal allocation that exploits the monotonicity and concavity of throughput function. As no user will be denied service in our model[††], we must allocate a buffer space of at least one unit to each user. The remaining buffer space of $B - N$ units then need to be distributed among the $N$ users. The following pseudo-code summarizes the algorithm.

**Optimal Partitioning Algorithm (OP):**

```
1. Initialize the allocation: $m_i = 1\ \forall i$
2. Compute $\Delta T_i(m_i)$ for all $i$
3. While $B_r \triangleq \sum_i m_i < B$, do step 4
4. For $j = \arg\max_i\ \Delta T_i(m_i)$, $m_j := m_j + k_{\max}$
where $k_{\max} = \max\{k = 1, 2, \ldots, B - B_r | \Delta T_j(m_j + k - 1) \geq \Delta T_i(m_i) \forall i \neq j\}$
```

This algorithm is equivalent to the method reported in [17] as a computationally efficient version of Shih's algorithm [18], which solves a quite general optimal resource allocation problem. In our context, the basic idea of the algorithm is to greedily allocate one more buffer space in each step to the user (or one of the users) that would incur the maximum increase in throughput from that additional buffer space. Because of the monotonicity and concavity of the $\Delta T_i(m_i)$'s, the increase in throughput in each iteration is non-increasing with buffer size for each user. Taking advantage of this fact, the number of computations needed is reduced by allocating not one, but $k \geq 1$ buffer spaces at a time to the winner of each iteration, if after an increase of $k - 1$ it will still be the winner among all queues in terms of throughput increase per added buffer. We next prove the optimality of algorithm OP, and then discuss its complexity.

**Theorem 1:** Algorithm OP results in an optimal solution to Problem 1.

*Proof.* We refer the interested reader to the proof in [17], yet, for completeness, we include a concise proof of optimality here: Let $\{m_i^*\}$ be an optimal allocation. By feasibility, $\sum m_i^* \leq B$. The total throughput with this allocation is:

$$\begin{aligned}\sum_{i=1}^{N} T_i(m_i^*) &= \sum_{i=1}^{N} [(T_i(m_i^*) - T_i(m_i^* - 1)) \\ &\quad + (T_i(m_i^* - 1) - T_i(m_i^* - 2)) + \ldots \\ &\quad \ldots + (T_i(2) - T_i(1)) + T_i(1)] \\ &= \sum_{i=1}^{N} T_i(1) + \sum_{i=1}^{N} \sum_{k=1}^{m_i^*} \Delta T_i(k)\end{aligned}$$

[†]In queues occurring in practical communication systems (such as routers and switches) keeping track of the number of packets belonging to each session can be computationally intensive. Approaches for keeping approximate partitions, such as randomized methods, can be developed. We leave these outside the scope of this paper.

[††]Combining buffer allocation with admission or flow control is very interesting, yet outside the scope of this work.

Note that after initialization of each user with one unit of buffer, every possible allocation of the total $B$ buffer spaces to the $N$ users corresponds to choosing $B - N$ numbers out of the following set of size $(B - N)N$: $\{\Delta T_1(1), \Delta T_1(2), \ldots, \Delta T_1(B-N), \ldots, \Delta T_N(1), \ldots, \Delta T_N(B-N)\}$. OP performs an iteration for each next buffer unit, deciding which user to allocate this buffer unit. There are a total of $B - N$ units of buffer left after initialization, hence $B - N$ iterations in total. In iteration $k$, OP choses the highest of number among the yet unchosen elements of the set. Since for each $i$, $\Delta T_i(m_i^k)$ are non-increasing from one iteration to the next, the algorithm is equivalent to choosing the largest $B - N$ largest numbers in the set $\{\Delta T_i(m_i)\}$, $i = 1, 2, ..., N$, $\mathbf{m} \in \Psi$. The resulting sum cannot be smaller than $\sum_{i=1}^{N} T_i(m_i^*)$. As OP also respects feasibility, we conclude that the sum throughput of OP cannot exceed the optimal, and is therefore equal to the optimal, $\sum_{i=1}^{N} T_i(m_i^*)$.

### 2.1.1 Complexity

OP makes a total of $B - N$ selections in step 4, and per selection (except the final one) it makes $N - 1$ comparisons. Overall, no more than $B$ elements of the set $\{\Delta T_i(m_i)\}$ are computed. So overall, OP makes $O(B)$ computations and $O(N(B - N))$ comparisons. Therefore, this is a polynomial-time algorithm. Incidentally, note that the problem amounts to selecting the $B - N$ largest entries out of a set of size $N(B - N)$. Hence, depending on the relative sizes of $B$ and $N$ it may be possible to reduce the computations further using a binary search in this set, akin to “bubble-sort”. In fact, a -considerably more difficult to state- algorithm using Lagrange multipliers and the binary search idea, with complexity $O(N^2(\log B)^2)$ is reported in [19]. This could be advantageous for $B \gg N$. Next, we consider the application of optimal buffer allocation in several scenarios.

## 3. Applications

We start by presenting the solution of the $M/M/1/m_k$ case. Next, we investigate an idealized model of a system with parallel channels that undergo independent outage corresponding to Case 1 in the Introduction. We state and solve joint buffer allocation and channel assignment problem. We then turn to a setting where users or groups of users share the channel in time, which belongs to Case 2 defined in the Introduction. This time, the buffer allocation problem is solved for parallel $M/G/1/m_k$ queues.

### 3.1 Parallel M/M/1/$m_k$ Queues

Consider parallel $M/M/1/m_k$ systems such that $\sum_k m_k = B$. We want to know the throughput-maximizing $\{m_i\}$. Average throughput $T$ and packet drop probability $P_d$ of the $M/M/1/m$ queue [20] are:

$$T(\lambda, \rho, m) = \lambda \left(1 - \frac{(1-\rho)\rho^m}{1-\rho^{m+1}}\right) \tag{3}$$

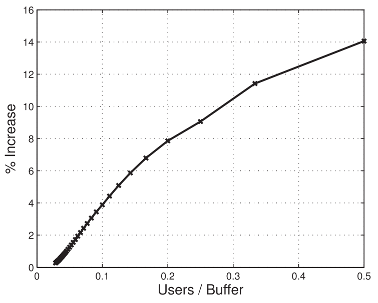

**Fig. 2** The percentage increase in total throughput vs. users per buffer. Optimal buffer allocation is compared to even buffer allocation in parallel M/M/1/$m_i$ system with a total buffer of $B = 3500$. 25% of all users have $\rho_1 = 1.1$, and the remaining have $\rho_2 = 0.1$. As more users share the buffers, buffer allocation yields higher increase in throughput.

$$P_d(\rho, m) = \frac{(1-\rho)\rho^m}{1-\rho^{m+1}} \tag{4}$$

Application of OP with the above throughput expression yields the optimal buffer partitions. We observe that even for parallel queues with Poisson arrivals and memoryless service distribution, optimal partitions can yield a significant increase in throughput compared to an even buffer allocation, as exhibited by numerical results some of which are presented in Fig. 2. This observation motivates considering other application scenarios for optimal partitioning.

Note that the percentage increase in the throughput becomes higher as more users share the available buffer space. This is due to monotone decreasing property of $\Delta T_i(m)$.

### 3.2 Parallel $M/D/1/m_i$ Queues

Towards a somewhat more realistic service model, consider a finite memory constraint, and packets of fixed length. There are parallel channels with constant rate, hence the service times are deterministic. For M/D/1/K, the buffer occupancy probabilities are, $P_k = R_k P_0,\ k = 1, ..., K$, where

$$R_k = \sum_{i=1}^{k} (-1)^{k-i} e^{Ai} \left[\frac{(Ai)^{k-i}}{(k-i)!} + \frac{(Ai)^{k-i-1}}{(k-i-1)!}\right] \ for\ k \geq 2 \tag{5}$$

$R_1 = e^A - 1$ and $A$ is the load factor. From $\sum_{j=0}^{K} P_j = 1$, we have, the blocking probability $P_0 = \frac{1}{1+\sum_{j=1}^{K} R_j}$ and normalized throughput $T = 1 - P_0$. The effect of optimally partitioning buffers is observed in Fig. 3.

### 3.3 FDMA with Channel Outage

Consider a frequency division multiple access (FDMA) multi-user downlink. There are N users, and a frequency band will be allocated to each user. Each frequency band exhibits *outage* at random times, that is, the SNR dips below a level that can support the (fixed) code rate being used. On every channel, the outage periods are i.i.d. across time, and the starting times of outage events form a renewal process. The probability of outage depends on the frequency band used, and not on which user is using this channel[†].

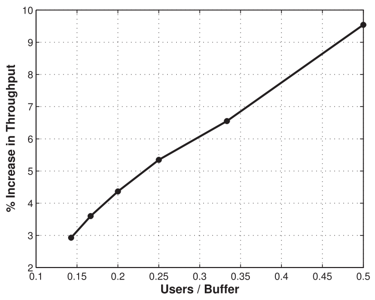

**Fig. 3** Percentage increase in throughput when the buffer management is switched from even partitioning to optimal partitioning in M/D/1/K queues. Percentage increase is higher for higher number of users sharing the buffer.

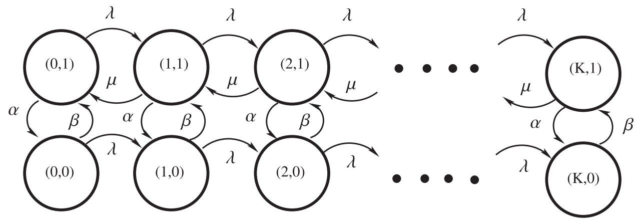

**Fig. 4** State transition diagram for joint channel and queue states. Channel is either on or off and queue states are allowed up to allocated buffer K.

The channel outage process of each user is assumed to be a continuous time Markov chain with rate of transitions $\alpha$ and $\beta$ for on to off and off to on transitions respectively, as shown in Fig. 4. For channel $i$ we have,

$$p_i^{out} = \frac{\beta_i}{\alpha_i + \beta_i} \tag{6}$$

Various settings for $\alpha$ and $\beta$ model various rates of channel variation with respect to arrival rate. The case where $\alpha, \beta \ll \lambda, \mu$, modeling a channel variation timescale much slower than arrivals, is particularly interesting, because in the limit $\alpha \to 0, \beta \to 0$ ($\alpha/\beta$ being constant), a closed-form expression can be written for long term-average drop rate. In this extreme case, both outage durations and the periods between two outages are long enough for sufficiently many packet arrivals and services such that the queue reaches steady-state. Since the queue reaches steady-state in both outage and non-outage, each user's queue behaves like an $M/M/1/m_i$ queue during non-outage, and is full (contains exactly $m_i$ packets) during outage. Specifically, let queue $i$ be served in frequency band $i$, whose outage probability is $p_i^{out}$. In this regime, the queue is full at steady state in outage, so the stationary probability of drop in outage is 1. In the non-outage case the packet drop probability is $P_d(\rho, m)$. The overall long term average drop rate for user $i$ is then:

$$P_{d_i}^{avg}(\lambda, p_i^{out}, m_i) = (1 - p_i^{out}) P_{d_i}(\lambda, m_i) + p_i^{out} \tag{7}$$

Correspondingly, the long-term average throughput is:

$$\begin{aligned} T(\lambda, p^{out}, m) &= \lambda[1 - P_d^{avg}(\lambda, p^{out}, m)] & (8)\\ &= (1 - p^{out})\lambda[1 - P_d(\lambda, m)] & (9) \end{aligned}$$

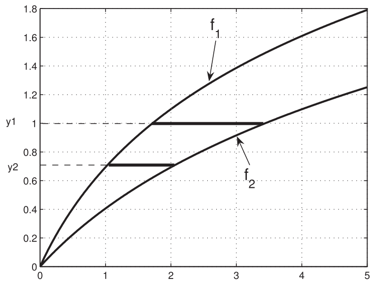

**Fig. 5** Monotone Inverse Disuniting Functions. The difference increases as y is increased.

The algorithm to find optimal buffer allocation can be applied with a slight modification in this case.

$$\Delta T_i^{out}(m_i, p_i^{out}) = (1 - p_i^{out})\lambda_i[P_d(\lambda_i, m_i) - P_d(\lambda_i, m_i + 1)] \tag{10}$$

Under these assumptions, the introduction of outage channel to the problem brings forth a new dimension in terms of optimization: assigning the channels to users for optimal total throughput. Channels with outage probabilities $p_1, p_2, \ldots, p_N$ are matched to the users in a one-to-one fashion.

**Problem 2:** Given $\lambda_i$ and available channels' outage probabilities $p_i$, maximize $\sum_i (1 - p_{\pi(i)}) T_i(\lambda_i, m_i)$ subject to $\sum_i m_i = M$ and $m_i \geq 1$, where $\pi$ is any permutation of $i = 1, 2, ..., N$.

We shall reach the solution of Problem 2 in Theorem 3, which will show that the problems of buffer allocation and channel assignment are separable in our outage formulation: The optimal solution is a best-channel highest-arrival rate allocation, i.e., channel assignment is based on arrival rate but not on queue (buffer) state. We start by noticing that the throughput functions are "monotone inverse disuniting".

Two monotone positive real functions are *monotone disuniting* if their difference diverges to infinity, as shown in Fig. 5. Note that monotone functions have well-defined inverse functions. In our analysis, we will use the same idea for inverses and we introduce *monotone inverse disuniting functions*.

**Definition 1: Monotone Inverse Disuniting Functions** The pair of functions $f_1$ and $f_2$ are said to be monotone inverse disuniting if

1. $f_1 : \mathfrak{R}^+ \to I_1$ and $f_2 : \mathfrak{R}^+ \to I_2$, $I_1, I_2 \subset \mathfrak{R}^+$ are monotone increasing with $f_1(x) > f_2(x)\ \forall x \in \mathfrak{R}^+$.
2. $\forall y_1, y_2 \in I_1 \cap I_2, y_1 > y_2 \Rightarrow$ $(f_2^{-1}(y_1) - f_1^{-1}(y_1)) > (f_2^{-1}(y_2) - f_1^{-1}(y_2))$

The following theorem is useful in finding the jointly optimal resource allocation.

**Theorem 2:** Let $M$ be a positive constant and

[†]The channel statistics not depending on user (and hence receiver location) may correspond, for example, to the case when the receivers are geographically clustered far away from the base station.

$$S \triangleq \{(x_1, x_2) : x_1 + x_2 \leq M, x_1 \geq 1, x_2 \geq 1\}$$

If $f_1, f_2$ are monotone inverse disuniting and $\alpha_1 > \alpha_2 > 0$,

$$\max_{\mathbf{X} \in S}\{\alpha_1 f_1(x_1) + \alpha_2 f_2(x_2)\} > \max_{\mathbf{X} \in S}\{\alpha_1 f_2(x_2) + \alpha_2 f_1(x_1)\}$$

Proof of Theorem 2 can be found in the Appendix. Note that this theorem is valid if the arguments of functions $f_1$ and $f_2$ are assumed real numbers, though they are integers in the problem. However, the argument in the proof is almost always true for the integer case also (see Appendix).

**Corollary 1:** For $\alpha_1 > \alpha_2 > ... > \alpha_K > 0$, and $(f_i, f_j)$ $\forall i < j$ are monotone inverse disuniting, permutation $\pi^*$ that solves the joint optimization problem

$$\max_{\pi, \mathbf{X} \in S} \alpha_{\pi(i)} f_i(x_i)$$

is the identity permutation $\pi^*(i) = i$

*Proof.* Assume another permutation $\pi'(i) \neq i$ solves the joint optimization problem. There exists at least two indices $i_1, i_2$ such that $i_1 < i_2$ and $\pi'(i_1) > \pi'(i_2)$ so that $\alpha_{\pi'(i_1)} < \alpha_{\pi'(i_2)}$. If above theorem is applied to these two indices, it is deduced that another permutation $\pi''$ with $\pi''(i_1) = \pi'(i_2)$ and $\pi''(i_2) = \pi'(i_1)$ yields better, which is a contradiction. Hence, the identity permutation $\pi^*(i) = i$ yields the joint optimal. ■

**Lemma 1:** For $\lambda_1 > \lambda_2$, let $f_i(m) = T(\lambda_i, m)$ $i = 1, 2$ as in Eq. (8). $f_1$ and $f_2$ are monotone inverse disuniting with $f_1(m) > f_2(m)$ $\forall m \in \Re^+$.

Proof of Lemma 1 can be found in the Appendix.

**Theorem 3:** Suppose $\lambda_1 > \lambda_2 > ... > \lambda_K$ and $p_1^{out} \leq p_2^{out} \leq ... \leq p_K^{out}$. Optimal channel allocation that solves Problem 2 is $\pi^*(i) = i$.

*Proof*: The result immediately follows from Theorem 2 and Lemma 1. ■

It is of interest whether the separation of the channel-aware scheduling and buffer partitioning can be carried on to more general multiplexers.

## 3.4 User Selection and Multiplexing in a Time-Varying Channel

Now, we generalize our service model to cover the allocation mechanism of Case 2 in the Introduction. Here, rather than having parallel channels, the transmitter allows the transmission of packets of a proper subset of users at each time. Hence, there is a scheduling decision that needs to be made: which user/users to select at each time to transmit the data of. In greatest generality, this scheduling decision could be a function of all that is known: instantaneous channel states and time-average channel coding rates available to each user, as well as the instantaneous queue states and long term packet arrival rates of each user. We will restrict attention to schedulers that are informed of arrival rates and instantaneous channel states. Specifically, we shall consider the following type of policy: the scheduling decision is made based only on channel state (without respect to queue state). The queues are handled by a buffer partitioning policy. The buffer partitions are calculated as a function of average arrival rates, and the long-term average transmission rates (note that the average transmission rates are a function of the scheduling policy.)

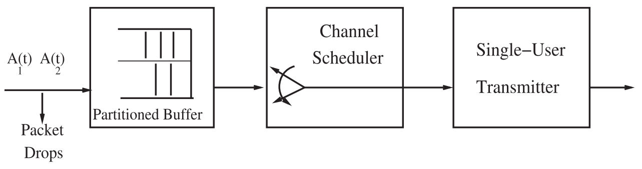

**Fig. 6** The model for two user joint buffer management and user scheduling in a time-varying channel.

Our ultimate goal is to understand whether the scheduling and buffer management problems are separable. Toward that goal, we first explore the issue on a two-user problem with on-off channels, the exact analysis of which will provide insight for the more general problem considered in the remainder of the paper.

### 3.4.1 Scheduling for Channels with Outage

Consider the model depicted in Fig. 6. There is a single-user transmitter, shared by two users. Packet arrival streams of the two users are Poisson with rates $\lambda_1$ and $\lambda_2$. W.l.o.g, let $\lambda_1 > \lambda_2$. Packet sizes are i.i.d., exponential with mean 1 unit. At any time, the channel states of the two users are independently "on" with probability $p_o$ and "off" with probability $1 - p_o$ (symmetric channels).

There is a scheduler that controls which user will access the transmitter. The scheduler works as follows: during epochs where only one of the channels is "on", the corresponding user is selected for transmission, and its data is transmitted (at unit rate.) When both channels are "on", user 1 will be selected with probability $a$, and user 2 will be selected for transmission with probability $1 - a$. As in the outage model of Sect. 3.3, we assume that channel change is slow so that scheduling epochs will be long enough (with respect to packet transmission) for the queues to reach steady-state in each epoch. Hence, whenever a user $i$ is selected, its buffer size evolves as an $M/M/1/m_i$ queue, where $m_i$ is the buffer partition assigned to it. The question we want to answer is the joint optimization of $m_i$ and $a$, and whether the optimization of one depends on the other, in this very simple setup.

The long term average fraction of time each user is effectively in outage is given by:

$$p_1^{\text{out}} = (1 - p_o) + (1 - a)p_o^2 \qquad (11)$$
$$p_2^{\text{out}} = (1 - p_o) + ap_o^2 \qquad (12)$$

Then long term throughput of user $i$ is:

$$T(\lambda_i, m_i, a) = (1 - p_i^{\text{out}})\lambda_i \left(1 - \frac{\lambda_i^{m_i}(1 - \lambda_i)}{1 - \lambda_i^{m_i+1}}\right) \qquad (13)$$

We now proceed to apply the $M/M/1/m$ optimal partitioning results to find the buffer allocation and state the joint buffer allocation-scheduling problem as follows:

**Problem 3:**

$$\max \ T_1(\lambda_1, m_1, a) + T_2(\lambda_2, m_2, a) \tag{14}$$

subject to

$$m_1, m_2 \geq 1, \ m_1 + m_2 \leq B, \ 0 \leq a \leq 1$$

Interestingly, the partitioning and channel allocation problems turn out to be separable. We summarize the optimal policy in the following theorem:

**Theorem 4:** Let $(a^*, m_1^*, m_2^*)$ be a solution of Problem 3. The following are true: (1) If $\lambda_1 = \lambda_2$, then $a^* = 0.5$, and if $\lambda_1 > \lambda_2$, then $a^* = 1$. (2)$(m_1^*, m_2^*)$ are found by running algorithm OP with the throughput functions stated above.

*Proof.* If $\lambda_1 = \lambda_2$, then by symmetry, $a = 1/2$. Let $\lambda_1 > \lambda_2$. First, by the previous separation theorem, it is clear that $a > 1/2$. The proof is based on the fact that $\frac{\partial}{\partial a}[T_1(\lambda_1, m_1^*(a), a) + T_2(\lambda_2, m_2^*(a), a)] > 0$ where $m_1^*(a)$ and $m_2^*(a)$ are the optimizing buffer allocations for fixed $a > 1/2$. More precisely, let $a$ be fixed and $m_i^*(a)$ be the corresponding buffer allocation. Since $\frac{\partial}{\partial a}\left[T_1(\lambda_1, m_1^*(a), a) + T_2(\lambda_2, m_2^*(a), a)\right] = p_0^2\left(f_1(m_1^*) - f_2(m_2^*)\right)$ where $f_1$ and $f_2$ are monotone inverse disuniting functions as discussed in Theorem 3. An implicit result of Theorem 2 is that $f_1(m_1^*) > f_2(m_2^*)$ because otherwise it would be possible to obtain better total throughput by assigning worst channel to the higher rate user. In conclusion, for fixed buffer allocation, it is possible to increase total throughput by incrementally increasing $a$. Since this result is true for all $a$ and corresponding optimal buffer allocations, then the optimizing value of $a$ must be 1. ■

It will be interesting to compare this policy with the *benchmark* queue-aware scheduling algorithm MaxWeight (MW). In this setting, MW reduces to selecting the user with longest queue when channels of both users are "on". Figures 7 and 8 depict the comparison of the proposed policy and MW: we see that the performance of the simple scheduler with optimal partitioning is very close to MaxWeight with equal partitioning. Here, the significance of partitioning to throughput is exhibited clearly: MW with CS has a throughput that falls with increasing load. This is due to "hogging" of the buffer by the first user.

### 3.4.2 The General Case

In this section we consider a more general case of a multiuser downlink, where the achievable rates to different receivers vary in time. The scheduler at the transmitter end will select one queue at a time. When the scheduler selects a queue, the packet (if any) at the head that queue will be transmitted to the corresponding user's receiver.

A set of assumptions will be made, to model the number of packets in queue $i$ as an $M/G/1/m_i$ queueing process, for each $i = 1, 2, \ldots, N$. In particular, suppose arrivals are Poisson processes at rates $\lambda_i$, $i = 1, 2, \ldots, N$, and packet lengths are constant. The effect of varying channel coding rates is captured by the assumption that the transmission time of a packet of user $i$ being random with some distribution function $F_{T_i}$. We assume that the transmission times of any two users are independent and identically distributed, such that $f_{T_n}(t) = f_T(t)$, $\forall\ n$, where $T$ is a nonnegative random variable. Furthermore, the transmission times for any user are assumed to be independent across *time*.

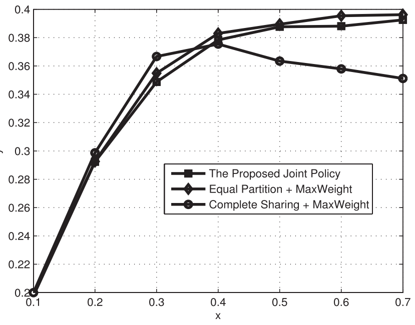

**Fig. 7** Performance comparison of the proposed joint policy and policies with MW scheduling. B = 5 buffers per user, $P_o = 0.3$ and $\lambda_2 = 0.1$.

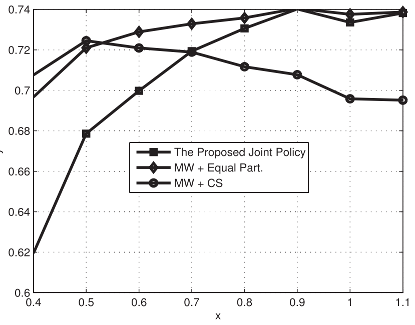

**Fig. 8** Performance comparison of the proposed joint policy and policies with MW scheduling. B = 5 buffers per user. $P_o = 0.5$. $\lambda_2 = 0.4$.

As soon as the service of a packet is finished, a decision will be made about the next queue to serve. This decision is in general a function of the current achievable rates of the users, equivalently, current transmission times, drawn independently from the distribution $F_T(t)$. For example, if the goal is to maximize sum rate, or instantaneous throughput, the corresponding decision rule is to select the user with the highest instantaneous rate, i.e., lowest of the instantaneous $T_i$'s. The resulting service time experienced by any given packet is the sum of the scheduling duration (the amount of time the packet spends at the head of line waiting to get selected by the scheduler), and the actual transmission time in the channel. By the independence across time of channel state processes, the service times of are IID. Thus, queue $i$, $i = 1, 2, \ldots, N$, is an $M/G/1/m_i$ system, where $m_i$ is the buffer allocated to it.

To get an approximate expression for average throughput, we will use Gelenbe's approximation [21] for M/G/1/$m$ packet drop probability for a queue with arrival rate $\lambda$, service rate $\mu$, and service times $X$, with $s^2 = \frac{Var(X)}{E(X)^2}$:

$$P_d^G(\lambda,\mu,m) = \frac{\lambda(\mu-\lambda)e^{-2\frac{(\mu-\lambda)(m-1)}{\lambda+\mu s^2}}}{\mu^2-\lambda^2 e^{-2\frac{(\mu-\lambda)(m-1)}{\lambda+\mu s^2}}} \tag{15}$$

The resulting approximate throughput is:

$$T(\lambda,\mu,m) = \lambda(1-P_d^G(\lambda,\mu,m)) \tag{16}$$

It can be verified that throughput in (16) is monotone increasing and concave with respect to $\lambda$ and $m$. Hence, the incremental buffer allocation algorithm also solves the throughput maximization problem here.

To proceed further, we will need to specify the service time distribution, which depends on the distribution of the transmission duration, $T$, as well as the scheduling rule.

Let $P(T > t) = e^{-t/\tau}$ for $t \geq 0$, that is, transmission times are IID Exponential with mean $\tau$. The expected service time of a packet of user $i$ the sum of transmission times of scheduled packets (of different users) from the time this packet advances to the head of its queue, to the end of its actual transmission.

(1) Scheduling the User with the Best Channel State

Let us first focus on a scheduling rule that selects the user with the best channel at any given time (with ties broken uniformly at random). We will call this policy MC (which stands for Max Channel). Under this selection rule, the probability that user $i$ is selected to receive service in a scheduling interval is simply $Pr(T_i = \min\{T_1,\ldots,T_N\}) = 1/N$.

The transmission time of the chosen user is the minimum of $N$ exponential r.v's. Let us call the transmission time of the user selected at the $n$th scheduling interval be $Q_n$. Note that $Q_n \sim Q$, which is again exponential with mean equal to $\tau/N$. The mean *service time* of a user consists of repeated trials until success. Let $K$ be the number of scheduling decisions up to and including the decision on which user $i$ is selected. $K$ is a geometric random variable. At each trial a user is selected and that user transmits. The total service time of a user therefore is the sum of a geometric number of IID exponential transmission durations, with the expectation and variance computed as below:

$$E[X] = E\left[\sum_{n=1}^{K} Q_n\right] = E[K]E[Q] = N\frac{\tau}{N} = \tau \tag{17}$$

Thus, $\mu = 1/E(X) = 1/\tau$. The variance is:

$$Var\left[\sum_{n=1}^{K} Q_n\right] = Var[K]E[Q]^2 + E[K]Var[Q] \tag{18}$$

$$= \frac{1-1/N}{1/N^2}(\tau/N)^2 + N(\tau/N)^2 = \tau^2 \tag{19}$$

As a result the parameter $s^2 = \frac{Var(T_s)}{E(T_s)^2} = 1$ for the scheduling policy MC under exponential packet durations.

(2) Round Robin Scheduling (TDM)

Under a simple round-robin scheduling policy, which corresponds loosely to a TDM (time division multiplexing) scheme, the parameters $s^2$ and $\mu$ can be computed as $E[X] = E[\sum_{n=1}^{N} T_n] = NE[T] = N\tau$, with $\mu = 1/E(X) = 1/(N\tau)$, and $\mathrm{var}[\sum_{n=1}^{N} T_n] = N\mathrm{var}[T] = N(\tau)^2$. So, for TDM scheduling under exponential transmission durations, $s^2 = 1/N$.

Of course, there are numerous other possibilities than MC and TDM for the scheduling rule. In fact, these are two of the simplest: the first, MC, seeks to maximize instantaneous rate alone, and is oblivious to long term throughput or fairness among users. On the other hand, the second, TDM, divides time evenly and fairly between users, but will consequently achieve a much smaller stability region than possible. They are both simple protocols corresponding to two different extremes. As a third scheduling policy, we shall consider MaxWeight (MW), which selects a queue to serve with the instantaneously highest ratio of queue size to transmission duration, also leads to a set of $M/G/1/m$ queues. This policy makes use of queue states in addition to channel states, and as mentioned in the introduction, *throughput optimal* under infinite buffers. Toward understanding whether partitioning mechanisms can make queue-blind policies perform close to queue-aware policies, in the next section a simulation experiment comparing these three scheduling mechanisms running side by side will be conducted.

## 4. A Comparison of Queue-Aware and Queue-Blind Policies on a Wireless Downlink

In previous sections, optimal partitioning methods have been given and shown to perform well under certain idealized scenarios. In two particular scenarios, it was shown that it is optimal to make the scheduling decision and the optimal buffer partition settings independently of each other, that is, the problems of scheduling and buffer partitioning are separable in those cases. A natural question to ask is whether such separation extends to less extreme and more practical wireless communication settings.

Approximate throughput expressions were derived in the previous section under certain assumptions for packet arrival processes and transmission durations, for two scheduling policies: MaxChannel and TDM. Using those throughput expressions, approximately optimal partitions can be set for a system using the respective scheduling policy, under the given channel statistics. Throughput expressions for MaxWeight under finite buffer is less tractable, so to obtain numerical values for throughput, we shall resort to simulation. Note that, as MaxWeight calculates the ratio of backlog to transmission duration for each user in any scheduling instant, and selects the queue with the highest ratio to serve, it relies on cross-layer information. In contrast, the other two scheduling rules, TDM, which is a simple round-robin policy, and MaxChannel, which selects the user with the best channel, do not require queue state information to schedule, hence they are simpler policies.

Users will be modeled to have IID channels as in the analysis of the previous section. Their queues will have different loads, however, because of the different arrival rates, $\lambda_i$. We will let the $\lambda_i$ of one of the users span a wide range while others' stay constant. This way, the throughput performance as the total load on the system increases (and gets more unbalanced) can be observed.

Different buffer management policies will lead to different throughput behaviors for a given scheduling rule. We will consider complete sharing (CS) and equal partitioning (EP) of the buffer for all three scheduling rules. The CS policy allows an incoming packet, regardless of which stream it is from, to be accommodated if there is available space in the buffer. On the other hand, EP reserves equal buffer spaces for each user, and does not packets use another stream's space. The OP policy is the one proposed in Sect. 2 with the throughput expression given in Eq. (16) substituted. The first and second order statistics of transmission duration on the channel is assumed to be known to the buffer manager.

Optimal partitioning (OP) is not meaningful for MaxWeight (as it inherently *controls* the queue sizes) and will only be computed for MC and TDM. However, for the sake of a fair exposition, the throughput of MW under the partitions set for MC-OP will also be plotted.

## 4.1 Simulation Setting and Results

The scheduling policies described above have been run on a long sequence of packet arrivals, for various arrival rates. Each data point is the result of running the given policy on $10^6$ packet arrivals generated randomly according to a Poisson process. Here is how the simulation is conducted: a packet that is generated by the arrival process is accepted and placed in queue if the buffer manager (whether it is CS, EP or OP) allows it (i.e., has room for a packet of this user). At the same time, the scheduling process continues in the following way: According to the scheduling policy, one of the queues is selected for service. This selection may be based on the current rates (or potential transmission durations) of the users. The transmission durations are generated as unit mean IID exponential random variables. The head-of-line packet (if any) of the selected user queue is served and leaves the queue, that is transmitted. At the end of this transmission, a new scheduling decision is made. Of course, if the selected queue is empty, a new scheduling instant starts immediately.

Figures 9–12 depict the results of this simulation experiment on a 2 user system where $\lambda_2 = 0.3$ is held constant, and $\lambda_1$ is varied. The service time distribution in the channel is exponential with mean 1, hence from equation $\mu = 1$ for each user under MC, and $\mu = 0.5$ for each user under TDM. Loads and throughput are normalized according to $\mu = 0.5$. Long term average throughput and the number of packet drops for all three scheduling rules combined with each of the buffer management policies are plotted respec-

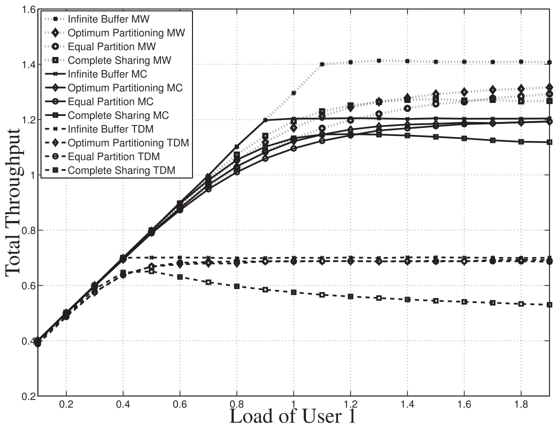

**Fig. 9** Throughput for N = 2 users, buffer allocation B = 5 per user, as the load on the first user is varied with second user's load set to $\rho_2 = 0.3$. The scheduling rules MaxWeight (MW), MaxChannel (MC), and Round Robin (TDM) are run on a sequence of $10^6$ packets, under each of three buffer constraints: infinite buffer, optimal partitioning, equal partitioning, complete sharing.

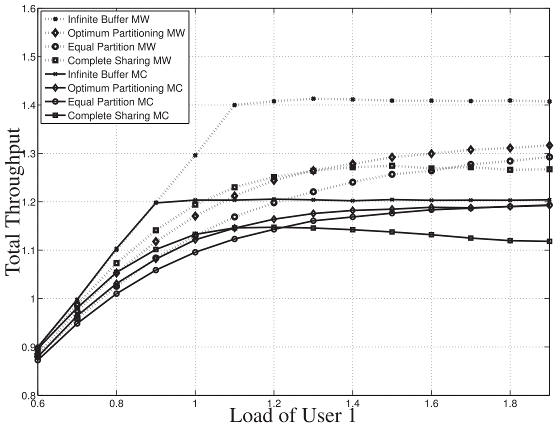

**Fig. 10** Throughput for N = 2 users (zoomed), buffer allocation B = 5 per user, as the load on the first user is varied with second user's load set to $\rho_2 = 0.3$. Performances of MaxWeight (MW) and MaxChannel (MC), run on a sequence of $10^6$ packets, shown under each of three buffer constraints: infinite buffer, optimal partitioning, equal partitioning, complete sharing.

tively in Figs. 9 and 11. Note that, on this figure results of running the policies under no restriction of buffer size, i.e., under infinite buffers, are also shown as a benchmark. Figures 10, 12 zoom in on the high load region for MW and MC, so that the differences between throughput and drop rate values of these schemes are better distinguished.

Figures 13 and 14 plot the total throughput and packet drops for the case $\rho_2 = 0.5$, for otherwise unchanged settings.

Next, we perform an experiment where the loads are unbalanced, in a 5-user system. The results are shown in Fig. 15 and Fig. 16.

## 4.2 Discussion of the Experimental Results

There are a number of observations to be made from the average throughput and drop rate curves plotted above. It

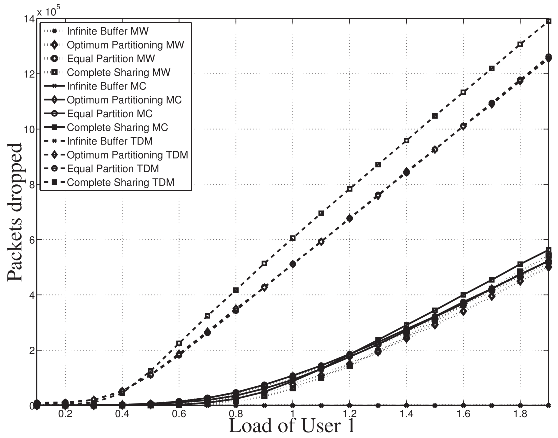

**Fig. 11** Number of packets dropped out of a total of $10^6$ incoming packets to N = 2 users, buffer allocation B = 5 per user, as the load on the first user is varied with second user's load set to $\rho_2 = 0.3$. The scheduling rules MaxWeight (MW), MaxChannel (MC), and Round Robin (TDM) are run on a sequence of $10^6$ packets, under each of three buffer constraints: infinite buffer, optimal partitioning, equal partitioning, complete sharing.

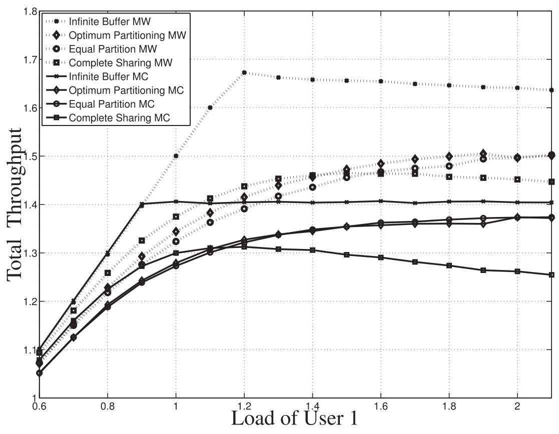

**Fig. 13** Throughput for N = 2 users, buffer allocation B = 5 per user, as the load on the first user is varied with second user's load set to $\rho_2 = 0.5$. The scheduling rules MaxWeight (MW), MaxChannel (MC), and Round Robin (TDM) are run on a sequence of $10^6$ packets, under each of three buffer constraints: infinite buffer, optimal partitioning, equal partitioning, complete sharing.

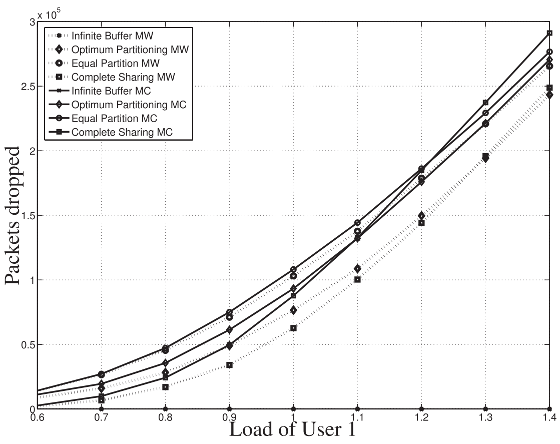

**Fig. 12** Number of packets dropped out of a total of $10^6$ incoming packets to N = 2 users, buffer allocation B = 5 per user, as the load on the first user is varied with second user's load set to $\rho_2 = 0.3$. The figure zooms in on the performance experienced by MaxWeight (MW), MaxChannel (MC), run on a sequence of $10^6$ packets, under each of three buffer constraints: infinite buffer, optimal partitioning, equal partitioning, complete sharing.

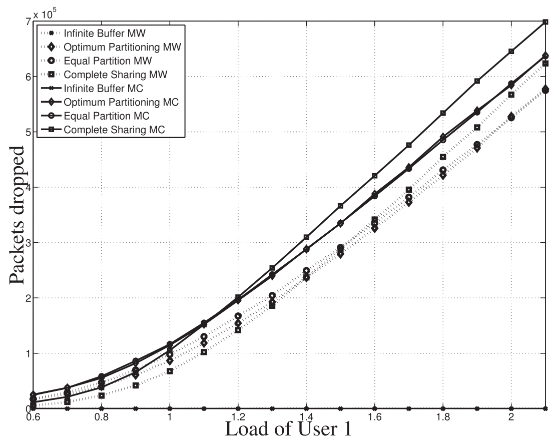

**Fig. 14** Total number of packet drops for N = 2 users, buffer allocation B = 5 per user, as the load on the first user is varied with second user's load set to $\rho_2 = 0.5$. The scheduling rules MaxWeight (MW), MaxChannel (MC), and Round Robin (TDM) are run on a sequence of $10^6$ packets, under each of three buffer constraints: infinite buffer, optimal partitioning, equal partitioning, complete sharing.

should be noted that these are only simulations, and the generality of the conclusions to be drawn from them are inevitably limited by the restrictions imposed by the simulation settings described above. Yet we shall make several observations that support the intuition gained from the preceding analysis in the paper, and several that seem to suggest new conclusions.

First and foremost, all the scheduling polices in consideration observe a steady drop in throughput after some load level under complete sharing. This was to be expected, due to high rate user(s) starting to occupy the buffer to the extent that the remaining queues are essentially starved. Thus, full capacity of the wireless channel cannot be utilized. Secondly, the "scheduling gain" of MW and MC over TDM is clearly observed: as expected, throughput is nearly doubled when going from TDM to MC. These were results that are expected from our analysis.

On the other hand, there are points that we have not analyzed in this paper, that are suggested by the plots. First, observing MaxWeight under complete sharing and partitioning is interesting. MaxWeight, while it has full control over the queue, also suffers a loss under some conditions with CS and can benefit from partitioning in these cases.

MaxChannel, even under Optimal Partitioning, cannot reach the throughput performance of MaxWeight. However, it is noteworthy that MC-OP and MC-EP perform quite close to MW, in the sense that the throughput difference is much less than that between MW-infinite buffer and MW-OP. This could be good news for system designers working with finite buffers, as schemes like MC, which do not

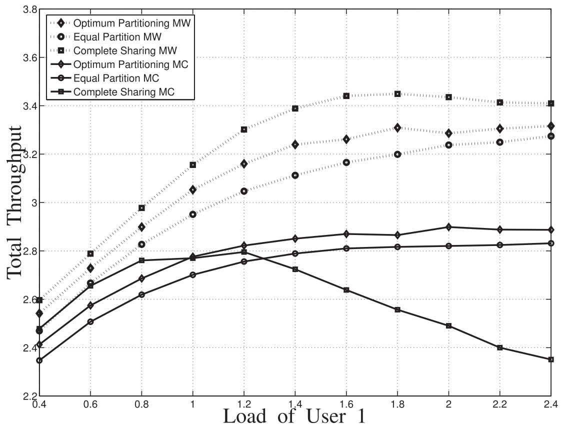

**Fig. 15** Throughput for N = 5 users with unbalanced load, with buffer space B = 5 per user. As the load of user 1 varies, the remaining users's loads are fixed at $\rho_2 = 0.2$ $\rho_3 = 0.8$, $\rho_4 = 0.3$, and $\rho_5 = 0.9$.

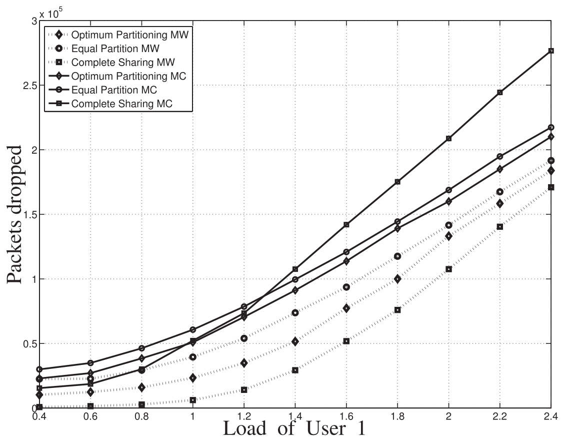

**Fig. 16** Number of packets dropped with increasing load for N = 5 users with unbalanced load, with buffer space B = 5 per user. As the load of user 1 varies, the remaining users's loads are fixed at $\rho_2 = 0.2$ $\rho_3 = 0.8$, $\rho_4 = 0.3$, and $\rho_5 = 0.9$.

need queue information (and just use static partitions that may depend on arrival and service statistics) are simpler to implement than a cross-layer mechanism such as MW that needs joint queue and channel state information to operate. What can also be good news from an implementation point of view is that using optimal partitions (OP) did not have an appreciable gain over equal partitions (EP) in these settings, even when loads of users are very unbalanced. This suggests that at least in some scenarios simply partitioning the buffer equally and using a channel-aware scheduler can get nearly full throughput.

Finally, it should be noted that the comparisons above have been made only for total throughput, as it is the focus of the paper. On the other hand, throughput per user can vary considerably between scheduling schemes, and among buffer allocation methods. It is to be expected, for example, for some users to be almost completely starved and see very major throughput loss under certain load conditions with complete sharing. While issues such as fairness between users is outside the scope of our treatment, these have not been addressed here. We refer the interested reader to [22] which explores the fairness issue to some extent.

## 5. Conclusion

This paper has examined buffer partitioning as a buffer management method that can allow multiuser diversity gain without a very complex scheduling algorithm in a finite buffer wireless downlink.

We started by showing the optimality of a polynomial-time iterative algorithm for finding partitions, and adapted it for various wireless communication models. We showed the separability of the optimal buffer partitioning and user scheduling or channel assignment problems under several simple models.

For optimum throughput on a time-varying multiuser wireless channel, opportunistic communication is necessary- that is, one must take advantage of times at which groups of users experience high channel gain, by sending at high rate to those users. In order for such channel scheduling to be effective, one needs to have a sufficient supply of packets belonging to the selected group of users, hence preventing starvation of queues crucial under realistic finite-buffer constraints. We have observed that under unbalanced or bursty load, a shared queue can lead to starvation, whereas partitioning buffers according to the arrival and service statistics is effective in keeping high throughput. A throughput-maximizing algorithm under finite buffers for a time-varying multiuser channel is not known. Combining the approximate throughput expressions and the results for optimal partitioning in a simulation experiment where three prominent scheduling policies are compared, we have gained the intuition partitioning coupled with simple channel-aware scheduling achieves close to maximum throughput. We conclude that these observations encourage further study of optimal scheduling for multiuser wireless channels under finite buffer constraints, as well as low complexity practical approaches such as buffer partitioning.

## Acknowledgments

We thank A. Sinan Akyurek for his help with the simulations, and N. Tekbiyik with her help with the plots. The material in this paper is based upon work supported by TUBITAK under Kariyer grant 106E119 and NSF under grant CCF-0635242.

### References

[1] M.I. Irland, "Buffer management in packet switch," IEEE Trans. Commun., vol.COM-26, no.3, pp.328–337, March 1978.

[2] F. Kamoun and L. Kleinrock, "Analysis of shared finite storage in a computer network node environment under general traffic conditions," IEEE Trans. Commun., vol.28, no.7, pp.992–1003, July 1980.

[3] P. Viswanath, D. Tse, and R. Laroia, "Opportunistic beamforming using dumb antennas," IEEE Trans. Inf. Theory, vol.48, no.6, June

2002.

[4] L. Tassiulas and A. Ephremides, "Dynamic server allocation to parallel queues with randomly varying connectivity," IEEE Trans. Inf. Theory, vol.39, no.2, pp.466–478, 1993.

[5] A. Paulraj, R. Nabar, and D. Gore, Introduction to Space Time Wireless Communications, first ed., Cambridge University Press, New York, USA, 2003.

[6] H. Kim, C. Han, and I. Kang, "Reducing tcp response time in face of wireless uplink losses," IEEE VTC 54th, Vehicular Technology Conference, 2001, vol.1, pp.262–266, 2001.

[7] D.J. Leith, L.L.H. Andrew, T. Quetchenbach, R.N. Shorten, and K. Lavi, "Experimental evaluation of delay/loss based TCP congestion control algorithms," [Online], http://www.hamilton.ie/net/delay_tests_final.pdf

[8] S. Sarkar, "Optimum scheduling and memory management in input queued switches with finite buffer space," IEEE Trans. Inf. Theory, vol.50, no.12, pp.3197–3220, Dec. 2004.

[9] E. Yeh and A. Cohen, "Throughput and delay optimal resource allocation in multiaccess fading channels," Proc. IEEE International Symposium on Information Theory, 2003, pp.245–245, June-July 2003.

[10] G. Foschini and B. Gopinath, "Sharing memory optimally," IEEE Trans. Commun., vol.31, no.3, pp.352–360, March 1983.

[11] S. Krishnan, A. Choudhury, and F. Chiussi, "Dynamic partitioning: A mechanism for shared memory management," Proc. IEEE INFOCOM'99, Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies, vol.1, pp.144–152, March 1999.

[12] L.L.H. Andrew, S.V. Hanly, and R.G. Mukhtar, "Active queue management for fair resource allocation in wireless networks," IEEE Trans. Mobile Comput., vol.7, no.2, pp.231–246, Feb. 2008.

[13] I. Cidon, L. Georgiadis, R. Guerin, and A. Kamisy, "Optimal buffer sharing," IEEE J. Sel. Areas Commun., vol.13, no.7, pp.1229–1240, Sept. 1995.

[14] G.-L. Wu and J. Mark, "A buffer allocation scheme for atm networks: Complete sharing based on virtual partition," IEEE/ACM Trans. Netw., vol.3, no.6, pp.660–670, Dec. 1995.

[15] S. Ziya, "On the relationships among traffic load, capacity, and throughput for the m/m/1/m, m/g/1/m-ps, and m/g/c/c queues," IEEE Trans. Autom. Control, vol.53, no.11, pp.2696–2701, Dec. 2008.

[16] E. Altman and A. Jean-Marrie, "The loss process of messages in an M/M/1/K queue," Proc. INFOCOM 94, Networking for Global Communications, pp.1191–1198, 1994.

[17] J.M. Einbu, "On shih's incremental method in resource allocations," Operational Research Quarterly (1970–1977), vol.28, no.2-Part 2, pp.459–462, 1977.

[18] W. Shih, "A new application of incremental analysis in resource allocations," Oper. Res., vol.25, no.4, pp.587–597, Dec. 1974.

[19] T.I.N. Katoh and H. Mine, "A polynomial time algorithm for the resource allocation problem with a convex objective function," Oper. Res., vol.30, no.5, pp.449–455, May 1979.

[20] R.G. Gallager, Discrete Stochastic Processes, Kluwer Academic Publishers, Boston/Dordrecht/London, 1996.

[21] G. Li and H. Liu, "Dynamic resource allocation with finite buffer constraint in broadband ofdma networks," Proc. IEEE Wireless Communications and Networking Conference (WCNC) '03, vol.2, pp.1037–1042, March 2003.

[22] T. Girici, O. Ozel, and E. Uysal-Biyikoglu, "Buffer sharing on an ofdma downlink," IEEE 21st International Symposium on Personal, Indoor and Mobile Radio Communications, pp.1162–1167, Sept. 2010.

## Appendix

### A.1 Proof of Theorem 2

*Proof.* Let $Z = \max_{\mathbf{x} \in S}\{\alpha_1 f_2(x_2) + \alpha_2 f_1(x_1)\}$, $\mathbf{x}^* = (x_1^*, x_2^*) = \arg\max_{x \in S}\{\alpha_1 f_2(x_2) + \alpha_2 f_1(x_1)\}$. It is enough to show that there exists some $(x_1^{**}, x_2^{**}) \in \mathcal{S}$ such that $\alpha_1 f_1(x_1^{**}) + \alpha_2 f_2(x_2^{**}) > Z$. To show this, we will consider two cases:
**1**: Assume $f_1(x_1^*) \geq f_2(x_2^*)$. Then setting $x_1^{**} = x_1^*$ and $x_2^{**} = x_2^*$ and exchanging the channels, $\alpha_1 f_1(x_1^{**}) + \alpha_2 f_2(x_2^{**}) > Z$.
**2**:Assume now $f_1(x_1^*) < f_2(x_2^*)$. Let's exchange the channels and define $x_1^{***} = f_1^{-1}(f_2(x_2^*))$ and $x_2^{***} = f_2^{-1}(f_1(x_1^*))$. Note that by definition we have $\alpha_1 f_1(x_1^{***}) + \alpha_2 f_2(x_2^{***}) = Z$. The same throughput is achieved with total buffer $X^{***} = f_1^{-1}(f_2(x_2^*)) + f_2^{-1}(f_1(x_1^*))$. In the previous allocation, total buffer was $X^* = x_1^* + x_2^* = f_1^{-1}(f_1(x_1^*)) + f_2^{-1}(f_2(x_2^*))$. Because of the monotone disuniting property (and for $f_1(x_1^*) < f_2(x_2^*)$), we have $f_2^{-1}(f_2(x_2^*)) - f_1^{-1}(f_2(x_2^*)) > f_2^{-1}(f_1(x_1^*)) - f_1^{-1}(f_1(x_1^*))$. After rearranging we get, $f_2^{-1}(f_2(x_2^*)) + f_1^{-1}(f_1(x_1^*)) > f_2^{-1}(f_1(x_1^*)) + f_1^{-1}(f_2(x_2^*))$. This means that $X^{***} < X^*$. The same throughput is achieved with smaller buffer memory. Hence, there exists some allocation $(x_1^{**}, x_2^{**}) \in S$ such that $\alpha_1 f_1(x_1^{**}) + \alpha_2 f_2(x_2^{**}) > Z$. ■

Now, assume arguments of $f_1$ and $f_2$ are restricted to integers. We can let the optimization be performed over integers. Then, the steps in the proof can be applied the same way in general. But there is an exceptional case in which monotone inverse disuniting property may not be sufficient. Let $f_1^{-1}(f_2(x_2^*)) = I_1 + d_1$ and $f_2^{-1}(f_1(x_1^*)) = I_2 + d_2$ such that $I_i$ and $d_i$ for $i = 1, 2$ are integer and fractional parts of the corresponding numbers. If $d_1 < 0.5$, $d_2 > 0.5$, $I_1 + I_2 = B - 1$ and $d_1 + d_2 < 1$, then a resource of amount $1 - (d_1 + d_2)$ is available but integer arguments can not be obtained by adding that amount. So, one has to decrease one of the arguments and increase the other. Adding the remaining *fractional* resource by decreasing one of the arguments and increasing the other may not yield better total throughput.

### A.2 Proof of Lemma 1

*Proof.* The derivative w.r.t. $m$ is $\frac{-\rho^{m+1}(1-\rho)\ln\rho}{(1-\rho^{m+1})^2}$, which is always positive. The derivative w.r.t. $\rho$ is $\frac{1+m\rho^{m+1}-(m+1)\rho^m}{(1-\rho^{m+1})^2}$, which is also greater than zero (The nominator of the derivative is a convex function with minimum of zero). Therefore the first condition is satisfied.

As for the second condition, after some rearrangement, we get $f_i^{-1}(y) = \frac{\ln\left(\frac{\rho_i - y}{\rho_i(1-y)}\right)}{\ln\rho_i}$. Let's define $F_{21}(y) = f_2^{-1}(y) - f_1^{-1}(y)$.

$$F_{21}(y) = \frac{\ln\left(\frac{\rho_2 - y}{\rho_2(1-y)}\right)}{\ln\rho_2} - \frac{\ln\left(\frac{\rho_1 - y}{\rho_1(1-y)}\right)}{\ln\rho_1} \tag{A·1}$$

$$F'_{21}(y) = \left(\frac{1}{\rho_1 - y}\right)\frac{1}{\ln\rho_1} - \left(\frac{1}{1-y}\right)\frac{1}{\ln\rho_1}$$

$$-\left(\frac{1}{\rho_2 - y}\right)\frac{1}{\ln\rho_2} + \left(\frac{1}{1-y}\right)\frac{1}{\ln\rho_2} \qquad (A\cdot 2)$$

Collecting common terms once more, we get,

$$F'_{21}(y) = \frac{1}{\ln\rho_2}\left(\frac{\rho_2 - 1}{(\rho_2 - y)(1-y)}\right) + \frac{1}{\ln\rho_1}\left(\frac{\rho_1 - 1}{(\rho_1 - y)(1-y)}\right)$$

We know that $y < 1$ and $y < \rho_1, \rho_2$, therefore we need to check for the positivity of the terms $\frac{\rho_i - 1}{\ln\rho_i}, i = 1, 2$. For both of the cases $\rho_i > 1$ and $\rho_i < 1$, it is positive therefore the inverse difference function $F_{21}(y)$ is increasing in $y$. Hence, the pair of functions are monotone inverse disuniting.

**Ömur Özel** received the B.Sc. and the M.S. degrees both in electrical and electronics engineering with honors from the Middle East Technical University (METU) Ankara Turkey in June 2007 and July 2009 respectively. Since August 2009, he has been a graduate research assistant in Communication and Signal Processing Laboratory of Institute for Systems Research in University of Maryland College Park working towards Ph.D. degree in electrical and computer engineering. His research focuses on wireless networks, multiuser information theory, game theory and distributed algorithms.

**Elif Uysal-Bıyıkoğlu** graduated with rank 1 from the Middle East Technical University (METU), Ankara, Turkey in 1997, received the S.M. degree in Electrical Engineering and Computer Science from Massachusetts Institute of Technology (MIT) in 1999, and the Ph.D. degree in Electrical Engineering from Stanford University in 2003. From 2003 to 2005 she was at MIT as a postdoctoral lecturer of EECS and a member of the Research Laboratory for Electronics. From 2005 to 2006 she was an assistant professor of Electrical and Computer Engineering at the Ohio State University. Since 2006, she has been on the faculty at METU Dept. of Electrical and Electronics Engineering. Dr. Uysal-Biyikoglu is a recipient of the Vinton Hayes Fellowship at MIT, the Stanford Graduate Fellowship, a 2006–2009 NSF Foundations of Communication research grant, a 2007–2010 Turkish National Science Foundation Young Faculty Career Development Award and an IBM Faculty Award in 2010. Her research interests are at the junction of communication and networking theories. She has served as consultant to the industry on the design of wireless networks, especially ad-hoc and sensor networks.

**Tolga Girici** received his B.S. degree from Middle East Technical University, Ankara, Turkey, in 2000 and his Ph.D. degree from the University of Maryland, College Park, MD U.S.A., in 2007, both in Electrical Engineering. During 2000–2005 he has been a research assistant in the Institute of Systems Research at the University of Maryland. In 2005 he was an intern at Intelligent Automation Inc., Rockville, MD. Between January 2006 and June 2007 he was at Fujitsu Labs, College Park, MD, as a research assistant and intern. Since November 2007 he is an assistant professor at the TOBB Economics and Technology University. His research interests include wireless communications, ad hoc and sensor networks, queueing analysis, energy-efficiency and broadband wireless access. He is a member of IEEE Communication and Information Theory Societies.