Optimal Buffer Partitioning on a Multiuser Wireless Link

Omur Ozel Elif Uysal-Biyikoglu
Department of Electrical and Electronics Engineering
Middle East Technical University
Ankara Turkey
oozel,elif@eee.metu.edu.tr

Tolga Girici
Department of Electrical and Electronics Engineering
TOBB University of Economics and Technology
Ankara Turkey
tgirici@etu.edu.tr

***Abstract*— We consider a finite buffer shared by multiple packet queues. Throughput can be considerably improved by partitioning the buffer space among the queues judiciously, especially under a high load regime. We formulate optimal buffer partitioning as a resource allocation problem, the solution of which is found through a greedy incremental algorithm in polynomial time. The rest of the work is devoted to applying the optimal buffer allocation strategy in different scenarios modeling a wireless downlink. First, the strategy is applied in a general parallel $M/M/1/m_i$ system and a numerical study verifies that the strategy may boost the throughput considerably. Then, a multichannel extension of this system is considered when the users have different arrival rates and channels have different outage probabilities. Jointly optimal buffer space allocation and channel assignment problems in this scenario are shown to be separable. Lastly, buffer allocation is considered in a system where users need to be multiplexed and scheduled based on channel state. It is shown that this system can be modeled as a set of parallel $M/G/1/m_i$ queues to which the optimum buffer allocation strategy is again applicable. The improvement brought by optimal buffer allocation to scheduling based solely on channel-state is explored. It is observed that buffer optimization can result in remarkable throughput increase on top of channel-based user selection.**

I. 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 over-

Omur Ozel is now with the Department of Electrical and Computer Engineering, University of Maryland College Park MD 20742 USA loaded 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 Figure 1 where the last hop along the network 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.

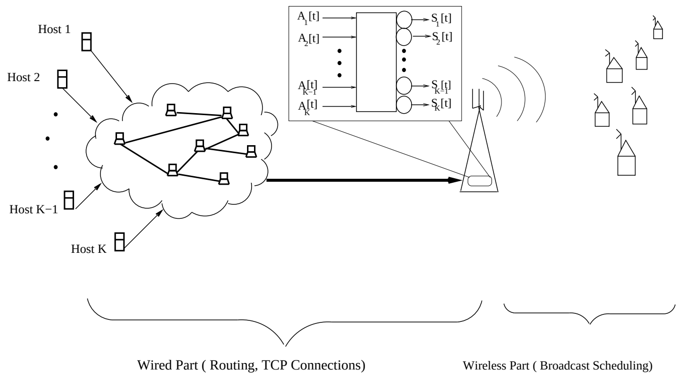
Host 1
Host 2
Host K−1
Host K
A1[t]
A2[t]
AK-1[t]
AK[t]
S1[t]
S2[t]
SK-1[t]
SK[t]
Wired Part (Routing, TCP Connections)
Wireless Part (Broadcast Scheduling)

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

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

A. 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 2×2 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].

B. 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) What is the throughput performance if we let scheduling be done without regard to queue state, and use optimally partitioned buffer for the resulting service rates? In answer to the first question, an optimal iterative algorithm for allocating buffer space to queues based on their arrival and service rates will be derived. The uniqueness of the resulting throughput maximizing buffer distribution will be shown. The second question is mainly addressed by extensive numerical studies using MATLAB. We first consider a "toy problem" on which we get an encouraging answer that separately handling buffer management and channel scheduling can get us near the performance of MaxWeight. We then consider a downlink multiuser system, where N independent packet arrival processes are separately queued to be sent by a single transmitter over a wireless channel which can be described as a stationary stochastic process. The service model depends on how the data streams are multiplexed to be transmitted. We consider two main channel allocation mechanisms:

- Case 1. Fixed channel allocation (e.g. an FDMA system with orthogonal channels, experiencing outage and possibly correlated fading.)

- Case 2. Channel-Aware Dynamic Scheduling (e.g. selecting user(s) with good channel states at each scheduling interval.)

In Case 1 (with parallel channels) we show that the jointly optimal buffer allocation-scheduling problems are separable. In Case 2, the buffer allocation problem is solved using an approximation to the $M/G/1/m$ blocking probability, and the resulting schemes are compared with queue-aware scheduling policies.

II. Buffer Partitioning

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

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

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[1].

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 [14], [11] 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.

A. 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 $\{i, 1 \le i \le N\}$ that work in parallel. Let $T(\lambda_i, m)$ be the throughput of the i^{th} queue, with arrival rate λ_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 $m+1$.

$$\Delta T_i(m) = T_i(m+1) - T_i(m) \qquad (1)$$

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:

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

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[2], 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 $k_{\max} = \max\{k = 1, 2, \ldots, B - B_r | \Delta T_j(m_j + k - 1) \ge \Delta T_i(m_i) \forall i \ne j\}$
```

This algorithm was previously reported in [17] as a computationally efficient version of Shih's algorithm [18], which solves an optimal resource allocation problem equivalent to ours. In our setting, 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 $k \ge 1$ 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

[1]In queues occurring in practical communication systems (such as routers and switches) keeping track of the number of packets belonging to each session is usually excessive. We realize that the partitionin model as given is not necessarily practical. We will go on with this idealized model regardless within the scope of this work, with the intention that the structural results obtained can provide guidelines to more sophistical models.

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

here: Let $\{m_i^*\}$ be an optimal allocation. By feasibility, $\sum m_i^* \leq B$. The total throughput with this allocation is:

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

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 $(B - N)N$: $\{\Delta T_1(1), \Delta T_1(2), \dots, \Delta T_1(B - N), \dots, \Delta T_N(1), \Delta T_N(2), \dots, \Delta T_N(B - N)\}$. 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 $\{\Delta T_i(m_i)\}$, $i = 1, 2, ..., N$, **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, $\sum_{i=1}^{N} T_i(m_i^*)$.

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 $O(N^2(logB)^2)$ is reported in [19]. This could be advantageous for $B \gg N$.

Next, we consider the application of optimal buffer allocation in several scenarios.

III. APPLICATIONS

We start by presenting the solution of the $M/M/1/m_k$ case. Next, we investigate an idealized a 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.

A. 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:

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

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

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 Figure 2. This observation motivates considering other application scenarios for optimal partitioning. Note that the

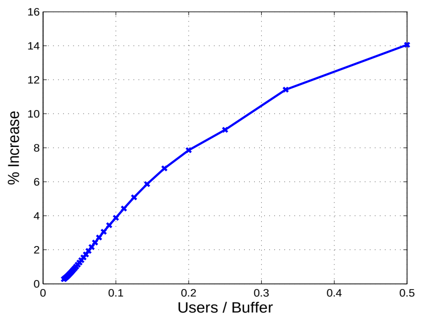

Fig. 2. The percentage increase in total throughput v.s. users per buffer. Optimal buffer allocation is compared to even buffer allocation in parallel M/M/1/m_i system with a total buffer of $B = 3500$. 25% of all users have $\rho_1 = 1.1$, and the remaining have $\rho_2 = 0.1$. As more users share the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

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

B. 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

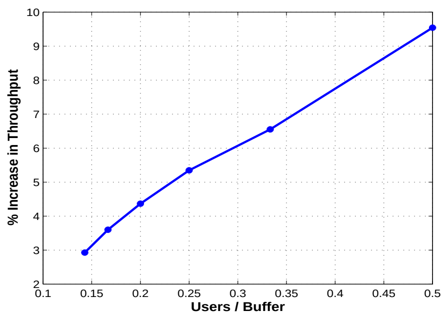

Fig. 3. Percentage increase in throughput when the buffer management is switched from even partitioning to optimal partitioning in M/D/1/K queues. Percentage increase is higher for higher number of users sharing the buffer.

probabilities are, $P_k = R_k P_0,\ k = 1, ..., K$, where

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

$R_1 = e^A - 1$ 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 Figure 3.

C. 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[3].

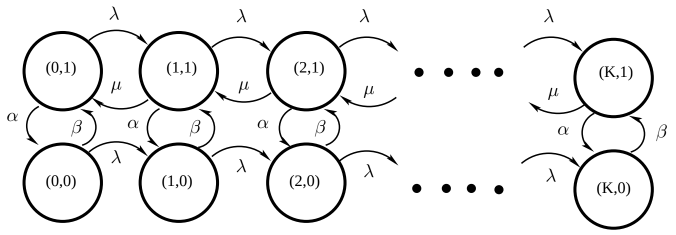

Fig. 4. State transition diagram for joint channel and queue states. Channel is either on or off and queue states are allowed up to allocated buffer K.

The channel outage process of each user is assumed to be a continuous time Markov chain with rate of transitions α and β for on to off and off to on transitions respectively. For channel i we have,

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

Various settings for α and β model various rates of channel variation with respect to arrival rate. The case where $\alpha, \beta << \lambda, \mu$, modeling a channel variation timescale much slower than arrivals, is particularly interesting, because in the limit $\alpha \rightarrow 0$, $\beta \rightarrow 0$ (α/β 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:

$$T(\lambda, p^{out}, m) = \lambda[1 - P_d^{avg}(\lambda, p^{out}, m)] \tag{8}$$

$$= (1 - p^{out})\lambda[1 - P_d(\lambda, m)] \tag{9}$$

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 λ_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 $m_i \geq 1$ and π 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. Note that monotone

[3]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.

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

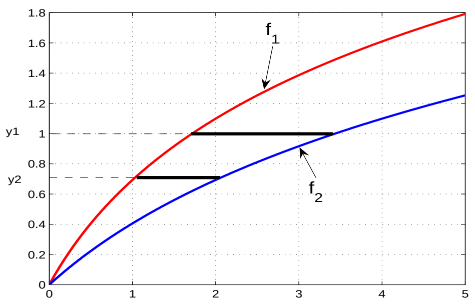

Fig. 5. Monotone Inverse Disuniting Functions. The difference increases as y is increased

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

Theorem 2: Let M be a positive constant and

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

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 π^* 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 π'' 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 Eqn 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.

D. 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.)

Our ultimate goal is to understand whether the scheduling and buffer management problems are separable. Toward that goal, we first explore the issue on the simplest possible problem. In the following, we describe and explore this "toy problem". Then, the more general problem will be considered.

1) Toy problem: two-users with on/off channels: Consider the model depicted in Figure 6. There is a single-user transmitter, shared by two users. Packet arrival streams of the two users are Poisson with rates λ_1 and λ_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 that only one of the channels is ”on”, the corresponding user is selected for transmission, and its data will be 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 subsection III-C, 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:

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

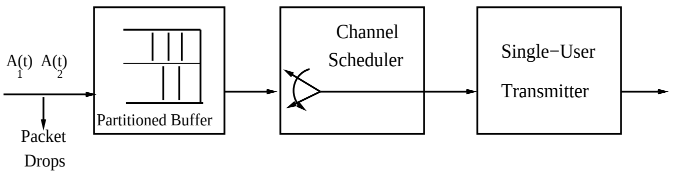

Fig. 6. The model for two user joint buffer management and user scheduling in a time-varying channel

Then long term throughput of user i is:

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

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 $\frac{\partial}{\partial a}[T_1(\lambda_1, m_1^*(a), a) + T_2(\lambda_2, m_2^*(a), a)] = p_0^2(f_1(m_1^*) - f_2(m_2^*))$ 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 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”. Figs. 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.

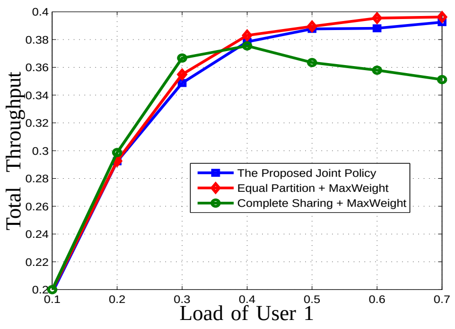

Fig. 7. Performance comparison of the proposed joint policy and policies with MW scheduling. B=5 buffers per user, $P_o = 0.3$ and $\lambda_2 = 0.1$.

2) The General Case: Now, a more general multiuser wireless downlink model similar to polling. Selection of user is based on channel state and the scheduling is performed at the end of service of a packet. Packet lengths are assumed constant. The achievable rate of any user is drawn from the same distribution, independently. Let this rate be R. The random variable $R \in \{1, 2, \ldots, r_{\max}\}$ is described by some probability mass function $p_R(r)$. We will assume that users have symmetric channels; more explicitly, their channel gains $h_i(t)$ are independent memoryless random processes with the

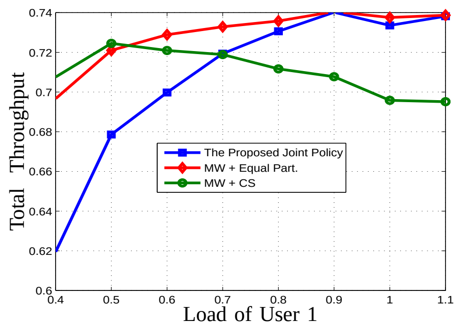

Fig. 8. Performance comparison of the proposed joint policy and policies with MW scheduling. B=5 buffers per user. $P_o = 0.5$. $\lambda_2 = 0.4$.

same statistics. Accordingly, if the packet of user i has been selected for service, the service time of a packet will be $1/R_i$ where R_i has the same distribution as R: $R_i \sim R$. Scheduling is performed at the same instant as the service of previous packet is finished and the user with maximum rate is selected. Since channels vary independently, scheduling times are IID. As arrivals are Poisson, the individual user packet queues can be viewed as M/G/1/m_i systems. We will use Gelenbe's approximate expression [21] for M/G/1/m_i packet drop probability P_d.

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

where $s = \frac{Var(T_s)}{E(T_s)^2}$. Throughput can be expressed in terms of P_d as follows:

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

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

E. Comparison of Queue-Aware and Queue-Blind Policies

In this section, we will compare throughput performances of several joint buffer management and scheduling policies by means of simulations. In particular, queue aware and queue blind scheduling with and without buffer partitioning will be compared in a multiple state wireless downlink channel.

Simulated user scheduling mechanisms are MaxWeight (MW), Max. Channel (MC) and Time Division Multiplexing (TDM). MW scheduling calculates the product of backlog and rate at each slot and selects the user that has the maximum of the products. Note that MW scheduler relies on cross layer operation of link layer and physical layer as it necessitates instantaneous backlog and channel state information. MC selects the user that has the best rate. Due to discrete nature of the rates, there may be ties, i.e. best rate can be achieved by more than one user. We assume that the scheduler has the information of arrival rate λ and the user with higher λ is selected in case of ties. This way, MC scheduler does not process the instantaneous backlog information but rather first order statistic of the arrival process, which makes operation of MC less complex than MW. TDM scheduling, the simplest of the three, is basically the round robin scheduling of users. As buffer management policies, complete sharing (CS), equal partitioning (EP) and optimal partitioning (OP) schemes are considered. CS policy allows each user to be accommodated if there is an available space. On the other hand, EP reserves equal buffer spaces for each user. OP policy is the one proposed in Section II with Gelenbe's expression used in the packet drop probability. The service rate and second order statistic of the service process is assumed to be known to the buffer manager.

One may argue that the proposed scheduling and buffer partitioning assumptions rely mainly on the idealistic assumption of Poisson arrivals. In the simulations, we will also examine the performance of the joint policies for bursty arrivals.

1) Simulation Setting and Results: Wireless channel of users is modeled as four-state independent discrete random variables. In each state, a rate is achievable. Assuming fixed length packets, we normalize the rate with packet length. For channel state i, i packets can be sent in each slot, $i \in \{0, 1, 2, 3\}$.

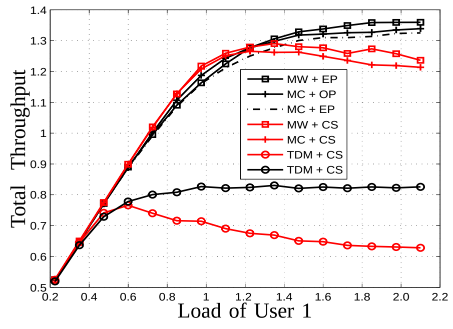

Fig. 9. N=2 and B=5 buffers per user. $\rho_2 = 0.3$. MC-OP performs quite close to MW.

We simulated several joint buffer management and user scheduling policies and compared their total throughput and average packet drop probability performances using MATLAB. In each experiment, 10^6 packet service time in the slowest rate are simulated. Operation is not slotted. Arriving packets are accepted if the management policy allows and the scheduling is done at the end of service of each packet. Since the channel states of users vary independently, the rate allocated to a packet is independent from those of other packets.

Firstly simulated is a 2 user system with λ_2 fixed and λ_1

is varied. Channel service capacity is $\mu = 0.35$ pkts/slot/user. Total throughput for different policies are depicted in Figs. 9 and 10. Loads and throughput are normalized according to μ. $\rho_2 = 0.3$ in Fig. 9 and $\rho_2 = 0.6$ in Fig. 10. The multiuser diversity gain is observed when TDM and MC scheduling are compared. Throughput of MW scheduling with CS or EP is observed to outperform the others but the MC scheduling with OP performs quite close to MW. CS policy in each scheduling has decreasing throughput after some load level though partitioning retains its performance.

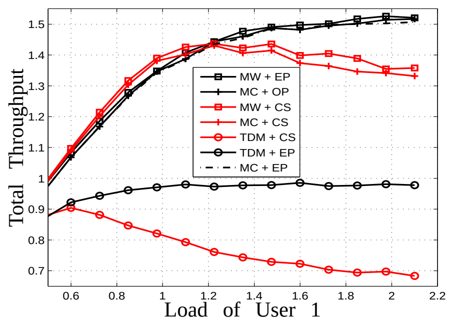

Fig. 10. N=2 and B=5 buffers per user. Load of user 1 is changing in the x-axis while $\rho_2 = 0.5$

Next, we experiment the joint policies in a 5 user downlink with unbalanced loads. The results are shown in Fig. 11. MW scheduling clearly outperforms the others. MC + OP policy comes after the MW. The advantage of optimal partitioning is observed.

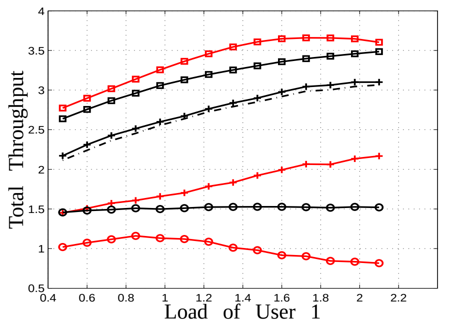

Fig. 11. N=5 and B=5 buffer per user. A realistic unbalanced load regime: x-axis represents load of user 1 and $\rho_2 = 0.2$ $\rho_3 = 0.8$, $\rho_4 = 0.3$, $\rho_5 = 0.9$

In the last experiment, we examine the performance of the policies under bursty arrivals though we use the formulas for Poisson arrival. In particular, Markovian Modulated Poisson Arrivals (MMPA) are assumed. Total throughput and packet drop probability of the joint policies are shown in Fig. 12. Similar trends are observed as the Poisson case. Finally, as expected, CS policies enter the hogging regime (where throughput starts its downward slope) sooner as arrivals get burstier.

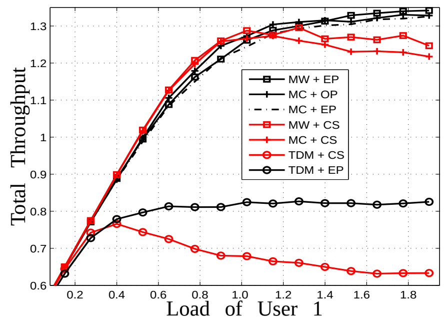

Fig. 12. The performance of buffer partitioning under MMPA with modulating Markov chain transition probabilities equal to 0.5. Arrival rate is 0.2λ and 1.8λ according to the state of the Markov chain. $\rho_2 = 0.3$

IV. CONCLUSION

In this paper, we examined buffer partitioning as a buffer management method, and its possible use in enhancing the use of a multiple-user wireless channel. Partitioning buffers for unbalanced load according to the arrival and service statistics came out as an effective method to boost throughput performance. We exhibited a polynomial-time iterative algorithm for finding optimal partitions, and adapted it for various scenarios. We showed the separability of the optimal buffer partitioning and user scheduling or channel assignment problems under several simple models. It was numerically observed that using first order statistic of the arrival process along with buffer partitioning can provide good performance improvement. These results encourage further study of optimal scheduling and buffer management in more realistic system models.

V. APPENDIX

A. 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.

B. 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. ρ 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 $F_{21}(y) = f_2^{-1}(y) - f_1^{-1}(y)$.

$$F_{21}(y) = \frac{\ln\left(\frac{\rho_2 - y}{\rho_2(1-y)}\right)}{\ln\rho_2} - \frac{\ln\left(\frac{\rho_1 - y}{\rho_1(1-y)}\right)}{\ln\rho_1} \tag{17}$$

$$\begin{aligned} F'_{21}(y) = {} & \left(\frac{1}{\rho_1 - y}\right)\frac{1}{\ln\rho_1} - \left(\frac{1}{1-y}\right)\frac{1}{\ln\rho_1} \\ & - \left(\frac{1}{\rho_2 - y}\right)\frac{1}{\ln\rho_2} + \left(\frac{1}{1-y}\right)\frac{1}{\ln\rho_2} \end{aligned} \tag{18}$$

Collecting common terms once more, we get,

$$\begin{aligned} F'_{21}(y) = {} & \frac{1}{\ln\rho_2}\left(\frac{\rho_2 - 1}{(\rho_2 - y)(1-y)}\right) \\ & + \frac{1}{\ln\rho_1}\left(\frac{\rho_1 - 1}{(\rho_1 - y)(1-y)}\right) \end{aligned} \tag{19}$$

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. ■

Acknowledgment: 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 Transactions on Communications*, vol. COM-26, 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, pp. 992–1003, July 1980.

[3] P. Viswanath, D. Tse, and R. Laroia, "Opportunistic beamforming using dumb antennas," *IEEE Transactions on Information Theory*, vol. 48, June 2002.

[4] L. Tassiulas and A. Ephremides, "Dynamic server allocation to parallel queues with randomly varying connectivity," *IEEE Transactions on Information Theory*, vol. 39, pp. 466–478, 1993.

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

[6] H. Kim, C. Han, and I. Kang, "Reducing tcp response time in face of wireless uplink losses," in *Vehicular Technology Conference, 2001. VTC 2001 Fall. IEEE VTS 54th*, vol. 1, pp. 262–266 vol.1, 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," in *[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 Transactions on Information Theory*, vol. 50, pp. 3197–3220, Dec. 2004.

[9] E. Yeh and A. Cohen, "Throughput and delay optimal resource allocation in multiaccess fading channels," *Information Theory, 2003. Proceedings. IEEE International Symposium on*, pp. 245–245, June-4 July 2003.

[10] G. Foschini and B. Gopinath, "Sharing memory optimally," *IEEE Transactions on Communications*, vol. 31, pp. 352–360, March 1983.

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

[12] L. L. H. Andrew, S. V. Hanly, and R. G. Mukhtar, "Active queue management for fair resource allocation in wireless networks," *IEEE Transactions on Mobile Computing*, vol. 7, pp. 231–246, February 2008.

[13] I. Cidon, L. Georgiadis, R. Guerin, and A. Kamisy, "Optimal buffer sharing," *IEEE Journal on Selected Areas in Communications*, vol. 13, pp. 1229–1240, September 1995.

[14] G.-L. Wu and J. Mark, "A buffer allocation scheme for atm networks: complete sharing based on virtual partition," *IEEE/ACM Transactions on Networking*, vol. 3, 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 Transactions on Automatic Control*, vol. 53, pp. 2696–2701, Dec. 2008.

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

[17] J. M. Einbu, "On shih's incremental method in resource allocations," *Operations 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," *Operations Research*, vol. 25, pp. 587–597, December 1974.

[19] T. I. N. Katoh and H. Mine, "A polynomial time algorithm for the resource allocation problem with a convex objective function," *Operations Research*, vol. 30, pp. 449–455, May 1979.

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

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