A Packet Scheduling Algorithm for Optimizing Downlink Throughput in Wireless LANs with the One-Sender-Multiple-Receiver Technique

Zhenghao Zhang and Steven Bronson
Computer Science Department
Florida State University Tallahassee, FL 32306, USA

***Abstract*—In this paper, we study the packet scheduling problem in wireless LANs with the One-Sender-Multiple-Receiver (OSMR) transmission technique. OSMR allows the Access Point (AP) to send distinct packets to multiple nodes simultaneously, and has great potential in improving the network downlink throughput. We note that the AP needs a packet scheduling algorithm to make the decision of when a packet should be sent and whether it should sent together with other packets using OSMR. In this paper, we focus on the problem of maximizing downlink throughout when packet fragmentation is not allowed. Since the processor of the AP is not powerful and cannot execute complicated algorithms in real time, we propose a simple algorithm and prove that it has a performance ratio of $\frac{1}{1+\sqrt{2}}$. We evaluated our algorithm with packet traces collected from 802.11a networks, and the results show that our algorithm improves the network throughput significantly.**

I. Introduction

Wireless Local Area Networks (LAN) are widely deployed due to their convenience. In this paper, we study the One-Sender-Multiple-Receiver (OSMR) technique, which allows one sender to send to multiple receivers simultaneously [1]. In a wireless LAN, if the Access Point (AP) is the OSMR sender, it can send distinct packets to multiple nodes simultaneously, hence improving the downlink performance. OSMR uses multiple antennas, and is a form of Multiple-Input-Multiple-Output (MIMO). However, it is different from the MIMO scheme proposed for 802.11n networks [2], because the MIMO transmission in 802.11n is still one-to-one, while OSMR allows one-to-many transmissions. OSMR has been studied in the context of multi-user MIMO in the signal processing community [3], [4] for cellular phone networks. However, existing works on multi-user MIMO typically assume homogeneous and constant traffic load of the nodes, while in a wireless LAN, the traffic load of the nodes are heterogeneous and random.

To take full advantage of OSMR, a packet scheduling algorithm is needed at the AP. The challenge is that only some pairs of nodes are *compatible*, i.e., allow the AP to send to them simultaneously with OSMR, where the compatibility relations are determined by the channel states. In [12], a packet scheduling algorithm was proposed for OSMR. However, for simplicity, the algorithm in [12] assumes nodes are at the same data rate and packets are of the same size. In this paper, we consider the more general case when nodes can be at different data rates and the packets can be of different sizes. We formalize the problem of maximizing network throughput when packet fragmentation is not allowed as finding a c-matching in a graph, and propose an algorithm with a performance ratio of $\frac{1}{1+\sqrt{2}}$ compared to the optimal algorithm. We evaluated our algorithm based on traffic traces collected from 802.11a networks, and the results show that it significantly improves the downlink throughput.

The rest of the paper is organized as follows. Section II discusses the background of OSMR and the basic transmission procedure. Section III describes our packet scheduling algorithm. Section IV evaluates the packet scheduling algorithms. Section V concludes the paper.

II. Background of OSMR and the Basic OSMR Transmission Procedure

Before discussing the packet scheduling problems, for completeness, we first briefly explain the physical layer background of OSMR, as well as the basic OSMR transmission procedure.

A. Background of OSMR

The following explanation is based on [1]. For simplicity, we consider a flat-fading channel. If the sender has two antennas, the sender can send a different symbol on each antenna, denoted as x_1 and x_2, respectively. When there are two receivers, each with one antenna, let h_{ij} be the channel coefficient from sender antenna j to receiver i where $i, j \in \{1, 2\}$. Suppose the received signal at node i is y_i. Suppose the noise at receiver i is n_i. The received signals can be represented as the product of the *channel matrix* and the transmitted signal vector, plus the noise:

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} n_1 \\ n_2 \end{bmatrix}$$

We denote the channel matrix as $\mathbf{H}$. Suppose d_1 and d_2 are the data symbols for receiver 1 and receiver 2, respectively. The data vector can be multiplied with a *processing matrix* $\mathbf{U}$ to produce the signals sent by the antennas:

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} u_{11} & u_{12} \\ u_{21} & u_{22} \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \end{bmatrix}$$

978-1-4244-4148-8/09/$25.00 ©2009
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE "GLOBECOM" 2009 proceedings.

Let $\mathbf{h}_i$ denote a row vector of $\mathbf{H}$ and let $\mathbf{u}_j$ denote a column vector of $\mathbf{U}$. If $\mathbf{h}_1\mathbf{u}_2 = 0$ and $\mathbf{h}_2\mathbf{u}_1 = 0$, $y_1 = \mathbf{h}_1[d_1\mathbf{u}_1 + d_2\mathbf{u}_2] + n_1 = d_1\mathbf{h}_1\mathbf{u}_1 + n_1$. Similarly, $y_2 = d_2\mathbf{h}_2\mathbf{u}_2 + n_2$. Since y_i is only relevant to d_i, different information is sent to to different receivers with no interference. We refer to $\mathbf{h}_i\mathbf{u}_i$ as the channel coefficient of the *effective channel* for receiver i. We say two receivers are *compatible* if matrix $\mathbf{U}$ can be found such that the Signal to Interference and Noise Ratios of their effective channels are above a threshold.

B. The Basic OSMR Transmission Procedure

To support OSMR transmissions, some extensions to the 802.11 MAC protocol are needed. As the focus of this paper is the packet scheduling algorithm, we give a short, high-level description of a basic OSMR transmission procedure adopted in our simulation, and leave the detailed MAC layer design issues to our future works. Basically, when the AP gains access to the medium, it may carry out an OSMR transmission, which consists of three phases: (1) channel estimation (2) data transmission (3) acknowledgments. Channel estimation is needed because the OSMR sender needs to process the channel according to the channel states. The AP may send a short packet, called Channel Estimation Request (CRQ), which contains a list of IDs of the nodes it wishes to send to. A node in the list should estimate the channel state and inform the AP of its channel state by sending a short packet back to the AP, called the Channel Estimation Report (CRP). To avoid collisions, nodes should send the CRP packets in turn according to the order of IDs in the CRQ packet. After getting all channel state reports, the AP may start the data transmission with OSMR. Once the transmission finishes, the nodes that received data correctly should send acknowledgments (ACK) to the AP, also in turn according to the ID list in the CRQ packet. The consecutive CRP packets and the ACK packets should be separated by SIFS.

III. Downlink Optimization

In this section, we study the packet scheduling problem with OSMR. In particular, we focus on maximizing the throughput on the downlink when packets are not fragmented. The main constraint is that the processor in the AP is usually inexpensive and not very powerful. In addition, the time to make the scheduling decision is short, e.g., less than the transmission time of a packet. We will therefore focus on simple algorithms that are capable of giving reasonably good schedules in common cases, although they may not always give the optimal schedules.

Before getting access to the medium, the AP inspects the packets in its buffer, and schedule one or multiple packets to send. To maximize the throughput, the AP should send out packets in minimum time. We assume that the AP first attempts to find an *optimal schedule*, with which the packets in the buffer can be sent in minimum time. The AP then selects one packet or a group of packets in this schedule and sends the selected packet(s).

In a wireless LAN, nodes may have different data rates. For example, 802.11a and 802.11g support data rates of 6, 9, 12, 18, 24, 36, 48 and 54 Mbps. Also, packets may have different sizes. It is possible to use OSMR to send packets of different sizes to nodes at different data rates because the AP can make the signal to one node appear as zero at the other node, and vice versa. In an 802.11 LAN, the packet transmission time involves not only the transmission time of the data, but also overhead such as DIFS, the possible random back-off, etc. We first focus on minimizing the data transmission time, because the data transmission time dominates the packet transmission time in most cases. At the end of this section, we will discuss our algorithm when the overhead is considered.

A high-level description of our approach is given in the following. Basically, we first formalize the problem of finding the optimal schedule as finding a *maximum weight c-matching* in a graph, then propose a greedy algorithm to solve it approximately. To maximize throughput, we need only run the greedy algorithm until it finds one *star*, which will be used to determine the group of packets to be sent.

A. Problem Formulation

For simplicity, we assume that the AP knows exactly the compatibility relations of nodes in the LAN. In practice, the AP may obtain this information approximately based on the received channel estimation reports. We also do not consider packet fragmentation in this paper. If only to minimize the packet transmission time, the schedule may become sending packets in a continuous stream of packet pairs, as shown in Fig. 1(a). However, this is not practical for two reasons. First, the channel coefficients may be outdated during the transmission. Second, a wireless LAN must ensure a certain level of fairness and sending the packets in a stream forbids other nodes from transmitting. We therefore focus on the practical case when one OSMR transmission involves sending one *main packet* along with one or multiple *side packets*, as shown in Fig. 1(b). We refer to such transmission as a *group transmission*. Clearly, in a group transmission, if there are v side packets, the transmission time of the main packet should be more than the total transmission time of the first $v-1$ side packets, because otherwise packet v can be sent as a stand-alone packet. Note that the side packets may have different destinations.

Given any optimal schedule that minimizes the packet transmission time, for any group transmission, we may sort the side packets according to their transmission time, and let the side packet with the longest transmission time start first. The modified group transmission is called a *sorted* group transmission. After the modification, if the transmission of the main packet finishes before some of the side packets start to transmit, we may let these side packets be sent as stand-alone packets. Note that the total transmission time of the sorted group transmission plus the possible stand-alone packets is the same as the original group transmission. Therefore, there must exist an optimal schedule in which all group transmissions are sorted. Therefore, when attempting to minimize the packet

978-1-4244-4148-8/09/$25.00 ©2009
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE "GLOBECOM" 2009 proceedings.

Fig. 1. (a). Sending packets in a stream of pairs, which is *not* practical. (b). Examples of group transmissions, where packets shown at the top are the main packets.

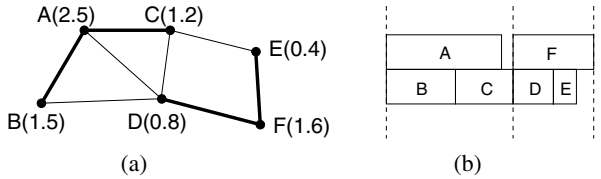


Fig. 2. (a). A graph with six vertices where the capacities of the vertices are shown in the parenthesis. The heavy edges belong to a c-matching. (b). The packet transmission schedule based on the c-matching.

transmission time, we need only consider schedules where all group transmissions are sorted.

We draw a graph G where each vertex represents a packet. Two vertices are connected by an edge if the packets are compatible, i.e., are destined to two compatible nodes. We define the *capacity* of a packet as the transmission time of the packet and denote it as $C()$. The capacity is basically the size of the packet divided by the data rate of the node. For example, Fig. 2(a) shows such a graph with six vertices representing six packets. We define the *weight* of an edge ab as $\min\{C(a), C(b)\}$ and denote it as $W(ab)$. Consider a star with root a denoted as $\phi(a) = \{ab_1, ab_2, \ldots, ab_v\}$. In this paper, when a star is written as $\{ab_1, ab_2, \ldots, ab_v\}$, it is always assumed that $W(ab_1) \geq W(ab_2) \ldots \geq W(ab_v)$. The star is called "legitimate" if $C(a) > \sum_{j=1}^{v-1} W(ab_j)$. Note that a legitimate star corresponds to a sorted group transmission where a is the main packet while b_1 to b_v are the side packets. For example, in Fig. 2, $\{AB, AC\}$ is a legitimate star. Define a c-matching of G as a set of vertex-disjoint legitimate stars. Note that any schedule for sending the packets where the group transmissions are sorted defines a c-matching, and vice versa. For example, $\{AB, AC\}, \{FD, FE\}$ is a c-matching in Fig. 2(a), which corresponds to the packet transmission schedule shown in Fig. 2(b). We use $W[]$ to denote the total weight of a set of edges. If $\phi(a)$ is a star in a c-matching M, we define the *actual weight* of $\phi(a)$ with respect to M as $U_M[\phi(a)] = \min\{C(a), W[\phi(a)]\}$. For example, in Fig 2, the actual weight of $\{AB, AC\}$ is 2.5 and the actual weight of $\{FD, FE\}$ is 1.2. Note that the actual weight of $\phi(a)$ is the air time saved for sending packets a, b_1, b_2, ..., b_v by using OSMR, comparing to sending the packets one-by-one without using OSMR. Define the weight of a c-matching as the total actual weight of the stars in the matching. Because the weight of the c-matching corresponds to the total air time that can be saved, a maximum weight c-matching in G corresponds to an optimal schedule. Therefore, in the following, we focus on finding a maximum weight c-matching in the graph.

B. The Scheduling Algorithm

Note that in the case when all vertices have the same capacity, the problem of finding the optimal c-matching reduces to finding a maximum matching which still takes $O(n^{2.5})$ time where n is the number of vertices in the graph [9]. Because the processors in the APs are not powerful, we focus on faster greedy algorithms. Before doing so we first define the *actual weight* of an edge with respect to a c-matching. Given a c-matching M, for a star $\phi(a) = \{ab_1, ab_2, \ldots, ab_v\} \in M$, if $C(a) \geq \sum_{j=1}^{v} W(ab_j)$, define the actual weight of ab_j as $U_M(ab_j) = W(ab_j)$ for all $1 \leq j \leq v$; otherwise, define the actual weight of ab_j as $U_M(ab_j) = W(ab_j)$ for $j < v$ and $U_M(ab_v) = C(a) - \sum_{j=1}^{v-1} W(ab_j)$. If an edge is not in M, its actual weight is not defined. For example, the actual weights of edge AB, AC, FD, FE are 1.5, 1.0, 0.8, and 0.4, respectively. Note that the total actual weight of edges in M is the weight of M. We also need the following lemma.

Lemma 1: $C(a) \geq U_M[\phi(a)]$ where $\phi(a)$ is the set of edges incident to a in a c-matching M.

Proof: If $\phi(a)$ is a star rooted at a in M, clearly, $C(a) \geq U_M[\phi(a)]$. Otherwise, $\phi(a)$ belongs to a star rooted at another vertex, and it must consist of only one edge, say, sa, while $C(a) \geq W(sa) \geq U_M(sa)$. ■

We propose Algorithm 1 which is a greedy algorithm for finding a c-matching M. Basically, the algorithm finds the vertex with maximum capacity denoted as a, and in each step, it adds the edge incident to a with maximum weight until $W[\phi(a)] > \frac{C(a)}{\sqrt{2}}$, where $\phi(a)$ denotes the set of edges in M incident to a. We prove that the weight of the matching returned by the greedy algorithm is at least a $\frac{1}{1+\sqrt{2}}$ fraction of the weight of the optimal c-matching, i.e., its *performance ratio* is $\frac{1}{1+\sqrt{2}}$. Condition $W[\phi(a)] > \frac{C(a)}{\sqrt{2}}$ is used in the algorithm because it maximizes the performance ratio, as explained in the proof.

Algorithm 1 A greedy algorithm for c-matching

1: $M \leftarrow \emptyset$.
2: **if** G is empty **then**
3: **return** M
4: **end if**
5: Let a be the vertex with maximum capacity.
6: **repeat**
7: Add to M the edge with maximum weight that is currently not in M and is incident to a.
8: **until** $W[\phi(a)] > \frac{C(a)}{\sqrt{2}}$ or no edge can be found
9: Remove a and all vertices matched to a as well as all edges incident to them from the graph. Goto 2.

Theorem 1: The greedy algorithm has a performance ratio of $\frac{1}{1+\sqrt{2}}$.

Proof: Let the optimal matching be M^*. When the greedy algorithm adds an edge, for example, ab, to M, we say a is matched by edge ab if a has not been matched by other edges before, and similarly for b. When the algorithm terminates, we check the vertices matched in M in the order when they were matched. In the case when two vertices were matched by the same edge at the same time, which only happens when the first edge is added to $\phi(a)$ for vertex a where a is the vertex

978-1-4244-4148-8/09/$25.00 ©2009
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE "GLOBECOM" 2009 proceedings.

found at line 5, a is checked first. When checking a vertex, say, a, we check edges in M^* and say an edge is "assigned" to a if this edge is incident to a and has not been assigned to other vertex before. Call the set of edges assigned to a vertex the "assigned set" of this vertex and denote it as $\Theta()$. Clearly, the assigned sets are disjoint with each other. Also, any edge in M^* must belong to one of the assigned sets, which we show by contradiction. Suppose this is not true, then there is an edge $st \in M^*$ not in any assigned set. It follows that both s and t are not incident to any vertex matched in M. But this cannot happen because the greedy algorithm will not leave two adjacent vertices unmatched. Therefore, the assigned sets for all matched vertices in M form a partition of M^*. We say the algorithm is *working on vertex* a when it is executing the repeat loop in line 6, 7, 8 for vertex a. Suppose the greedy algorithm added edge $\phi(a) = \{ab_1, ab_2, \ldots, ab_v\}$ to M when it finished working on a. We next prove that

$$U_M[\phi(a)] \geq \frac{1}{1+\sqrt{2}}\{U_{M^*}[\Theta(a)] + \sum_{j=1}^{v} U_{M^*}[\Theta(b_j)]\},$$

hence the performance ratio of the algorithm. We prove this by considering the two cases the algorithm exits the repeat loop.

Case 1. The algorithm exits the repeat loop because no edge can be added. We claim that in this case, $\Theta(a) \subseteq \phi(a)$. This is because if an edge in M^*, for example, sa, can be assigned to a, s must not have been removed from the graph when the algorithm started working on a. Since otherwise, suppose s has been removed from the graph when the algorithm added edge st to M before started working on a. In this case, sa should have been assigned to s, not to a. Therefore, all edges in $\Theta(a)$ were still in the graph when the greedy algorithm started on working a. Since the algorithm exits the loop because no edge can be added, all edges incident to a must have been added to $\phi(a)$, therefore $\Theta(a) \subseteq \phi(a)$. We partition the edges in $\phi(a)$ into two sets: those in $\Theta(a)$ and those not in $\Theta(a)$. Note that if the algorithm exits the loop because no edge can be added, $C(a) \geq \sum_{j=1}^{v} W(ab_j)$, and hence for any edge $ab_j \in \phi(a)$,

$$U_M(ab_j) = W(ab_j) = C(b_j).$$

Therefore, for an edge $ab_j \in \Theta(a)$,

$$U_M(ab_j) \geq U_{M^*}(ab_j),$$

since $C(b_j) \geq U_{M^*}(ab_j)$. For an edge not in M^*, say, ab_h, note that due to Lemma 1,

$$C(b_h) \geq U_{M^*}[\Theta(b_h)].$$

Therefore, if the algorithm exits the loop because no edge can be added, we actually have

$$U_M[\phi(a)] \geq U_{M^*}[\Theta(a)] + \sum_{j=1}^{v} U_{M^*}[\Theta(b_j)].$$

Case 2. The algorithm exists the repeat loop because $W[\phi(a)] > \frac{C(a)}{\sqrt{2}}$. Suppose when the algorithm exits the loop, $W[\phi(a)] = \beta C(a)$ where $\beta > \frac{1}{\sqrt{2}}$. Because the algorithm adds edges with largest weight to $\phi(a)$ first,

$$\frac{C(a)}{\sqrt{2}} > W(ab_v),$$

hence $\sqrt{2} > \beta$. Due to Lemma 1, $C(a) \geq U_{M^*}[\Theta(a)]$ and $C(b_j) \geq U_{M^*}[\Theta(b_j)]$ for all $v \geq j \geq 1$, hence

$$(1+\beta)C(a) \geq U_{M^*}[\Theta(a)] + \sum_{j=1}^{v} U_{M^*}[\Theta(b_j)].$$

If $1 \geq \beta > \frac{1}{\sqrt{2}}$, $U_M[\phi(a)] = \beta C(a)$, hence

$$U_M[\phi(a)] \geq \frac{\beta}{1+\beta}\{U_{M^*}[\Theta(a)] + \sum_{j=1}^{v} U_{M^*}[\Theta(b_j)]\}.$$

If $\sqrt{2} > \beta > 1$, $U_M[\phi(a)] = C(a)$, hence

$$U_M[\phi(a)] \geq \frac{1}{1+\beta}\{U_{M^*}[\Theta(a)] + \sum_{j=1}^{v} U_{M^*}[\Theta(b_j)]\}.$$

Note that in $[\frac{1}{\sqrt{2}}, 1]$, $\frac{\beta}{1+\beta}$ decreases as β decreases, with the minimum being $\frac{1}{1+\sqrt{2}}$ when $\beta = \frac{1}{\sqrt{2}}$. In $[1, \sqrt{2}]$, $\frac{1}{1+\beta}$ decreases as β increases, with the minimum being $\frac{1}{1+\sqrt{2}}$ when $\beta = \sqrt{2}$. Therefore overall we have

$$U_M[\phi(a)] > \frac{1}{1+\sqrt{2}}\{U_{M^*}[\Theta(a)] + \sum_{j=1}^{v} U_{M^*}[\Theta(b_j)]\}.$$

■

In practice, the AP may pick one star in the c-matching as the group of packets to be sent. Regarding to complexity of Algorithm 1, note that if the vertices are sorted according to the capacities and the edges incident to any vertex are sorted according to the weights, the greedy algorithm finishes in $O(n)$ time, where n is the number of vertices, because every execution of line 7 removes one vertex. Sorting the vertices takes $O(n \log n)$ time and sorting the edges takes $O(E \log E)$ time where E is the number of edges in the graph. Overall, the algorithm takes $O(E \log E)$ time. However, we note that the complexity is actually much smaller in practice. Note that the AP needs only choose one group of packets to send. As the algorithm never removes an edge from M once it is added to M, a star will remain in M once added to M. Therefore, the AP needs only run the algorithm until it added one star to M. Also, the sorting of the nodes and edges can be maintained incrementally upon packet arrivals and packet departures.

We next discuss the performance ratio when overhead is included. Because the overhead includes the random back-off time, a deterministic bound cannot be found, and we will focus on a bound in the average sense. Assume that the data transmission time of the optimal algorithm and the greedy algorithm are T_{o^*} and T_g, receptively. Assume the total air time of the packets is T_0. Based on Theorem 1, we have

$$\frac{T_0 - T_g}{T_0 - T_{o^*}} \geq \frac{1}{1+\sqrt{2}}. \quad (1)$$

978-1-4244-4148-8/09/$25.00 ©2009

We assume that the expected overhead incurred when sending the packets without using OSMR is αT_0, where α is a constant determined by the data rates of nodes in the network. When overhead is included, the optimal schedule needs at least T_{o^*}, which happens when the optimal algorithm has no overhead at all. We also argue that most likely, the overhead in the schedule given by the greedy algorithm is no more than αT_0. To see this, consider a star with $v+1$ vertices in the schedule given by the greedy algorithm. When using OSMR, the overhead includes one DIFS, one possible random back-off, one possible channel estimation process including the channel estimation packet sent by the AP and at most $v+1$ channel estimation reports, $v+1$ acknowledgment packets, and at most $2v+3$ SIFSs. When sending the packets one-by-one, the overhead includes $v+1$ DIFS, up to $v+1$ random back-off, $v+1$ acknowledgment packets and $v+1$ SIFSs. Note that DIFS is much longer than SIFS. Also, the channel estimation sequence and the reports are very short packets, while one random back-off can be substantially longer. Therefore, when overhead is included, with high probability, the schedule given by the greedy algorithm takes at most $T_g + \alpha T_0$ time. Due to Equ. 1, we have

$$T_g \leq T_0(1 - \frac{1}{1+\sqrt{2}}) + T_{o^*}\frac{1}{1+\sqrt{2}}. \quad (2)$$

We also note that $T_{o^*} \geq \frac{T_0}{2}$, which is because the optimal schedule can at most reduce the packet transmission time by half. Therefore,

$$\begin{aligned} \frac{T_g + \alpha T_0}{T_{o^*}} &\leq \frac{T_0}{T_{o^*}}(1 - \frac{1}{1+\sqrt{2}} + \alpha) + \frac{1}{1+\sqrt{2}} \\ &\leq 2 - \frac{1}{1+\sqrt{2}} + 2\alpha. \end{aligned} \quad (3)$$

Therefore,

Theorem 2: When overhead is considered, the transmission time according to the greedy algorithm is likely at most $2 - \frac{1}{1+\sqrt{2}} + 2\alpha$ times that of the optimal algorithm, where α denotes the ratio of the expected overhead over the data transmission time when sending the packets without using OSMR.

IV. Evaluations

We evaluate our algorithm with the trace data collected from 802.11a networks available at [10]. As we wish to evaluate the downlink packet scheduling, we used Trace 2 and Trace 3 in [10], in which the data were collected by TCPDump seen at the wired port at the AP, because it should preserve the arrival characteristics of the downlink traffic. More information about the trace data can be found in [10], [11].

In our simulation, we assume that on average, two nodes are compatible for α percent of the time, where α is randomly picked in $[0, 0.9]$. Two nodes alternate between the compatible state and the incompatible state, where the duration of the compatible period is set to be 0.4 second and the duration of the incompatible period is set according to α. As the network used for trace collection in [10] was in a confined 20m by 20m area, we assumed that all nodes are operating at 54 Mbps, the highest data rate of 802.11a networks, because all nodes are close to the AP. For a packet transmission not using OSMR, the transmission includes DIFS, random back-off, data transmission, SIFS and ACK. For a transmission of packets using OSMR, the transmission includes DIFS, random back-off, the channel estimation process, the OSMR data transmission, and the ACK packets from the nodes. If the group consists of n packets to v nodes, the packet transmission includes n ACKs but only v CRPs. The durations of DIFS, average backoff time and SIFS are set to be 34μs, 68μs, and 16μs, respectively. The transmission time of the data packet is assumed to be 20μs plus the time needed to send the data. The CRQ, CRP, and the ACK packets are assumed to take 25μs, 24μs, and 24μs, respectively.

Our simulation is event-driven. We keep track of T which is the time when the channel becomes free. When an uplink packet is encountered in the trace, T is incremented by the amount of time needed to transmit the packet. This, in effect, is to send the uplink packet immediately after the channel is free. We took this approach because the traffic in the trace is recorded at the wired port of the AP, therefore, when an uplink packet appears in the trace, the actual transmission already took place. When a downlink packet is encountered in the trace, it is added to the queue. The scheduling algorithm selects a packet or a group of packets in the queue to send when the channel is free and updates T.

Our algorithm is referred to as OSMR-c. For comparison, we implemented two other algorithms, referred to as FIFO and OSMR-s. FIFO does not use OSMR and sends packets in a first-in-first-out manner. The algorithm used in the commercial AP should be equivalent to FIFO in terms of throughput. OSMR-s uses OSMR, but follows a simple matching strategy: when looking for a star to send, it always regards the packet at the head of the queue as the main packet, then scans the packets in the buffer and adds a packet to the star if it is compatible with the main packet until the duration of the side packets exceeds the duration of the main packet. For further comparison, we also ran our simulation with our algorithm but assuming that all nodes pairs are always compatible and refer to it as OSMR-fl.

We first report the simulation results with Trace 2 in [10], which was collected in a LAN with 75 nodes for about 10 minutes. We ran our simulations for 500 seconds and show the throughputs of OSMR-c and FIFO in Fig. 3 for one random choice of the compatibilities of the nodes. We can see that both algorithms have almost exactly the same throughput. This is because the traffic trace was collected at an AP not supporting OSMR, hence the load is not high.

Because a higher traffic load is needed to evaluate OSMR, we processed Trace 2 and Trace 3 and created a single combined trace. We use the traffic trace from 400 seconds to 500 seconds, when load is more stable. As each trace contains 75 nodes, to reduce number of nodes, we merged the traffic of 7.5 nodes into one node on average and produced 20 merged nodes. We then randomly select the merged nodes and

978-1-4244-4148-8/09/$25.00 ©2009
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE "GLOBECOM" 2009 proceedings.

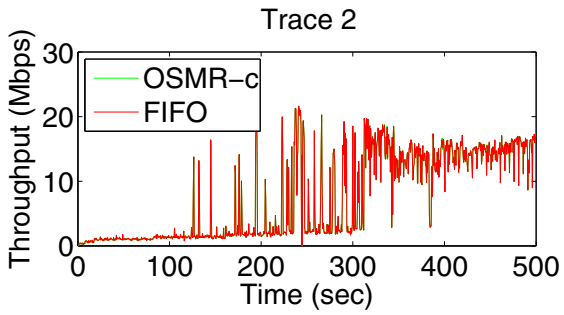
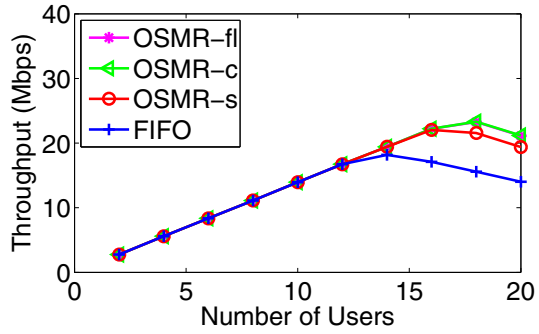


Fig. 3. Network throughput in 500 seconds.

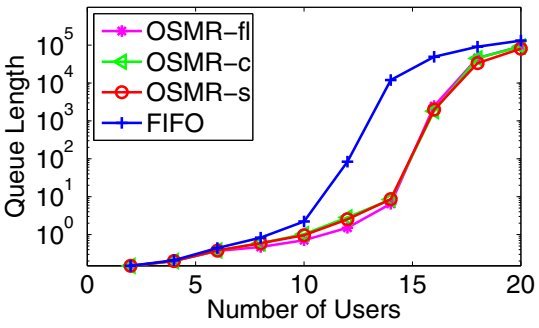
use their traffic as input to the simulation, where the number of selected nodes grows from 2 to 20 at a step of 2. The average network throughput during the 100 seconds and the average number of packets left in the queue after the 100 seconds are shown in Fig. 4(a) and Fig. 4(b), respectively, where each data point was obtained by averaging the results of 100 random seeds. We can see that when there no more than 12 merged users, the performance of all schemes are similar. However, when there are more than 12 merged users, the performance of FIFO is significantly worse than the OSMR schemes. For example, when there are 16 users, the throughput of FIFO is about 80% of the OSMR schemes, and the number of buffered packets is about one magnitude higher than the OSMR schemes. This confirms that OSMR is capable of significantly improving the network performance. Note that the performance of all schemes deteriorates when there are too many users, because when there are more users, the amount of uplink traffic also increases. Surprisingly, it can also be noticed that the performance of OSMR-c is almost identical to OSMR-fl, which suggests that an average compatibility percentage of 0.45 is sufficient to achieve the performance when nodes are all compatible. Also, although OSMR-s is better than FIFO, it is outperformed by OSMR-c when there are more than 16 users, which suggests that the greedy algorithm we propose is effective. Note that OSMR-c is a simple greedy algorithm and has a low complexity, so the improvement over OSMR-s is enjoyed at a low cost.

V. Conclusions

In this paper, we studied the packet scheduling problem when the wireless LAN adopts the One-Sender-Multiple-Receiver (OSMR) transmission technique which allows a sender to send to multiple receivers on the same frequency simultaneously. We focused on the problem of maximizing network throughout when packet fragmentation is not allowed. We proposed a simple algorithm and prove that it has performance ratio of $\frac{1}{1+\sqrt{2}}$ compared to the optimal algorithm. We evaluated our algorithm with packet traces collected from 802.11a LANs, and the results show that our algorithm significantly improves the throughput. Our future work includes physical layer implementation and experiments, MAC layer designs, as well as the scheduling algorithms when packet fragmentation is allowed.



(a)



(b)

Fig. 4. Comparison of different algorithms. (a) Throughput. (b) Queue length.

References

[1] D. Tse and P. Viswanath, "Fundamentals of wireless communication," *Cambridge University Press*, May 2005.

[2] "802.11n: Next-Generation Wireless LAN Technology," *http://80211n.com/white_paper/802_11n-WP100-R.pdf*.

[3] D. Gesbert, M. Kountouris, R. W. Heath, Jr., C. B. Chae, and T. Salzer, "From Single user to Multiuser Communications: Shifting the MIMO paradigm," *IEEE Signal Processing Magazine*, vol. 24, no. 5, pp. 36-46, Oct., 2007.

[4] Q. H. Spencer and A. L. Swindlehurst, "A hybrid approach to spatial multiplexing in multiuser MIMO downlinks," *EURASIP Journal on Wireless Communications and Networking*, vol. 2004, no. 2, pp: 236 - 247, December 2004.

[5] IEEE Computer Society LAN MAN Standards Committee, *IEEE Standard 802.11, Wireless LAN Media Access Control (MAC) and Physical Layer (PHY) Specifications, 1999*.

[6] D. Halperin, T. Anderson, and D. Wetherall "Taking the Sting out of Carrier Sense: Interference Cancellation for Wireless LANs," In *ACM MOBICOM 2008*.

[7] S. Gollakota and D. Katabi, "ZigZag Decoding: Combating Hidden Terminals in Wireless Networks," In *ACM SIGCOMM 2008*.

[8] M. Gast, "802.11 Wireless Networks: The Definitive Guide, 2nd Edition" *O'Reilly*, May 2005.

[9] H.N. Gabow and R.E. Tarjan, "Faster scaling algorithms for general graph matching problems," *Journal of the ACM*, 38(4):815853, 1991.

[10] http://www.winlab.rutgers.edu/ẽrgin/mobicom2007/

[11] M.A. Ergin, K. Ramachandran and M. Gruteser, "Extended Abstract: Understanding the Effect of Access Point Density on Wireless LAN Performance," In *ACM MOBICOM 2007*.

[12] Z. Zhang and Y. Yang, "Enhancing downlink performance in wireless networks by simultaneous multiple packet transmission," In *IPDPS '06*, Rhodes Island, Greece, April 2006.

978-1-4244-4148-8/09/$25.00 ©2009
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE "GLOBECOM" 2009 proceedings.