The One-Sender-Multiple-Receiver Technique and Downlink Packet Scheduling in Wireless LANs

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

Abstract

In this paper, we study the One-Sender-Multiple-Receiver (OSMR) transmission technique, which allows a sender to send to multiple receivers on the same frequency simultaneously by utilizing multiple antennas at the sender. OSMR has the potential to significantly improve the downlink performance of wireless LANs, because with OSMR, the Access Point (AP) can send distinct packets to multiple computers at the same time. To study the practicability of OSMR in the indoor environments typical to wireless LANs, we implemented a prototype OSMR transmitter/receiver with GNU Software Defined Radio and conducted experiments in a university building. To the best of our knowledge, this is the first implementation and experimentation of OSMR. Our results are positive and show that the wireless channels allow OSMR for a significant percentage of the time. We also note that with OSMR, packet scheduling is needed at the AP to determine when a packet should be sent and whether it should sent together with other packets using OSMR. We focus on the problem of maximizing network throughout, and propose a simple algorithm and prove that it has a performance ratio of $\frac{1}{1+\sqrt{2}}$ compared to the optimal algorithm. We evaluated OSMR and our algorithm with packet traces collected from 802.11a networks, and the results show that our algorithm significantly improves the network throughput. Our algorithm is simple and is suitable for the implementations in APs with inexpensive processors.

1 Introduction

Wireless Local Area Networks (LAN) offer convenient access to the Internet. However, wireless LANs are still much slower than wired LANs. For example, the maximum data rate of 802.11g and 802.11a networks is 54Mps, while the maximum data rate of a typical Ethernet LAN is 100Mps. In addition, measurement studies show that the typical throughput of an 802.11 network is only about half of the maximum data rate [19], while the typical throughput of an Ethernet can be much closer to its maximum data rate. As new applications such as Internet TV are demanding more and more bandwidth, improving the performance of wireless LANs has attracted much attention in both the academia and the industry.

In this paper, we study the One-Sender-Multiple-Receiver (OSMR) technique, which allows one sender to send to multiple receivers simultaneously by utilizing multiple antennas at the sender [12]. OSMR could significantly improve the downlink performance of wireless LANs, where the downlink refers to the link from the Access Point (AP) to the computers, because when the AP is the sender, it can send distinct packets to multiple com-

puters simultaneously. Most of the existing research related to OSMR focus on theoretical signal processing and often assume simplified network models, e.g., the availability of a feedback channel for channel state update, homogeneous and constant traffic load among users, etc [15, 18]. To the best of our knowledge, OSMR has not been implemented and tested for wireless LANs, where there is no feedback channel and traffic loads of users are heterogeneous and random. To find out the practicability of OSMR, we implemented a prototype OSMR transmitter/receiver with GNU Software Defined Radio (SDR) that allows one sender to send to two receivers simultaneously. An OSMR transmission depends on the channel states of the receivers because it requires the sender to process the signals according to the channel states. The critical questions related to the practicability of OSMR include (1) how likely are two receivers compatible, where two receivers being compatible means that their channel states allow the sender to use OSMR, and (2) whether the channel fluctuation speed is slow enough such that the measured channel state remains valid until the sender finishes sending, and whether the compatibility relations of receivers are stable enough to allow intelligent packet scheduling. Fortunately, our experiments reveal that two receivers are usually compatible for a significant percentage of the time. Also, although the compatibility relations vary as the channels fluctuate, in the indoor environment, the channel fluctuation is typically slow. Overall, our results are positive and show that the typical wireless LAN environments allow packet transmissions with OSMR. In addition, OSMR does not require much change to the receiver hardware and OSMR-capable nodes and OSMR-incapable nodes can co-exist in the same LAN.

To take full advantage of OSMR, a packet scheduling algorithm is needed at the AP. The AP runs this algorithm to decide which packet(s) to send to optimize the performance, e.g., maximizing the throughput. We formalize the problem of maximizing network throughput 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 throughout.

The rest of the paper is organized as follows. Section 2 discusses the related works. Section 3 describes our implementation of OSMR. Section 4 describes our OSMR experiments. Section 5 discusses application issues of OSMR and the backward compatibility with existing 802.11 networks. Section 6 describes our packet

scheduling algorithm. Section 7 evaluates the packet scheduling algorithms. Section 8 concludes the paper.

2 Related Works

In this section we discuss related works.

2.1 Wireless Transmission Techniques

Multiple-Input Multiple-Output (MIMO) and 802.11n. In an 802.11n LAN, to achieve a higher speed than existing wireless LANs, nodes use MIMO to communicate with the AP [16]. However, although multiple antennas are used, the transmission in 802.11n is still one-to-one. One the other hand, OSMR allows simultaneous transmissions between one sender and multiple receivers, which will help achieving an overall higher efficiency. For example, in a wireless LAN, often, some nodes have very strong channels while others have weak channels. Suppose node *A* has a weak channel, and the AP cannot further increase the data rate to *A* due to the constraint of transmission power. Suppose there is a node *B* that has a strong channel. Instead of sending only to *A*, the AP may allocate a very small amount of power to send to *B* simultaneously with *A* using OSMR. As *B* has a strong channel, even the AP is only allocating a very small amount of power for it, *B* may still receive at a good data rate. Also, because only a very small amount of power is diverged to *B*, *A* may still receive at the same data rate. Therefore, the downlink throughput is increased by an amount equal to *B*'s data rate.

Multi-User MIMO. OSMR is often referred to as multi-user MIMO in the signal processing community [15, 18]. Existing works on multi-user MIMO typically focus on the theoretical signal processing and often assume simplified network models, e.g., the availability of a feedback channel for channel state update, homogeneous and constant traffic load among users, etc. In a wireless LAN, there is no feedback channel and traffic loads of users are heterogeneous and random, hence the resource allocation problem is more challenging. In addition, in this paper, we provide implementations and experiments with OSMR in indoor environments typical to wireless LANs.

Multi-frequency approach. It has been shown that by simultaneously utilizing multiple frequency channels, the performance of a wireless LAN can be improved [22]. The implemented OSMR uses only one frequency. In fact, OSMR and multi-frequency techniques should complement each other because it is possible to schedule one OSMR transmission on each frequency channel. In this paper, we focus on OSMR transmissions on one frequency channel, as in a wireless LAN, there is almost always more than one node on the same frequency.

CDMA. OSMR is different from Code Division Multiplexing Access (CDMA), which also allows multiple nodes to communicate simultaneously on the same frequency. Basically, OSMR takes advantage of multiple antennas and is more efficient in utilizing the bandwidth than CDMA. A CDMA transmitter has to spread the signal bandwidth to a much larger bandwidth, which is not required in OSMR.

Cooperative MIMO. Cooperative MIMO [11] has been studied extensively, in which a set of network nodes jointly send information to the next hop. Cooperative MIMO usually focuses on using multiple nodes to send the same information to the next hop, while OSMR focuses on sending distinct information from one node to multiple nodes, which is of more practical interest in wireless LANs.

2.2 Other Recent Works

Recently, applying new signal processing techniques to packet-switched wireless networks has drawn much interest in the community, such as applying Successive Interference Canceling in [6], the ZigZag decoding in [7], the analog network coding in [10], and the joint packet demodulation in [9]. We note that OSMR addresses a unique issue on the downlink that has not been considered before.

In [4], it was demonstrated that it is possible to allow simultaneous transmissions between multiple sender/receiver pairs, as well as allowing one node to receive and forward simultaneously. However, to the best of our knowledge, techniques that allow one sender to send to multiple receivers has not been implemented before.

In [5], a packet scheduling algorithm for Multiple Packet Transmission, which is equivalent to OSMR, was proposed. However, the algorithm in [5] assumes nodes are at the same data rate and packets are of the same size, therefore, it only solves a special case of the problem considered in this paper. Also, OSMR was not implemented or tested in [5], while in this paper we provide implementation and measurements of OSMR transmissions.

3 The Implementation of OSMR

In this section, we describe our implementation of OSMR. We begin with the background of OSMR.

3.1 Background

We assume the channel is flat-fading. As wireless LANs typically operate in the high Signal to Noise Ratio (SNR) regime, in this explanation, for simplicity, we neglect noise. If the sender is sending data symbol d, the receiver will receive $y = hd$, where h is the complex channel coefficient. If there are two receivers and the sender has two antennas, the sender can send two different symbols denoted as x_1 and x_2 on antenna 1 and antenna 2, respectively. Suppose the channel coefficient from antenna j to user i is h_{ij} for $i, j \in \{1, 2\}$. Let the received signal at user i be y_i, which is a linear combination of the signals sent from each antenna multiplied by the channel coefficients:

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

We will use $\mathbf{H}$ to denote the channel matrix. We may process the data by picking a *processing matrix*

$$\mathbf{U} = \begin{pmatrix} u_{11} & u_{12} \\ u_{21} & u_{22} \end{pmatrix}$$

such that $\mathbf{h}_1 \mathbf{u}_2 = 0$ and $\mathbf{h}_2 \mathbf{u}_1 = 0$, where $\mathbf{h}_i$ denote a row vector of $\mathbf{H}$ and $\mathbf{u}_j$ denote a column vector of $\mathbf{U}$. If such matrix can be found, let d_1 and d_2 denote the data that should be sent to receiver 1 and receiver 2, respectively. We let

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

Thus receiver 1 will receive $\mathbf{h}_1(d_1 \mathbf{u}_1 + d_2 \mathbf{u}_2) = d_1 \mathbf{h}_1 \mathbf{u}_1$. Similarly, receiver 2 will receive $d_2 \mathbf{h}_2 \mathbf{u}_2$. Therefore, distinct data is sent to each receiver. In this paper, $\mathbf{h}_i \mathbf{u}_i$ is referred to as the *effective channel* for receiver i. Two receivers are *compatible* if a processing matrix can be found such that the strength of their effective channels are above a threshold.

3.2 GNU Software Defined Radio

We implemented OSMR in about 2,000 lines of C++ and Python code using GNU Software Defined Radio (SDR) [2]. GNU SDR is a very convenient platform for prototype implementations, as it allows developers to use software to generate the baseband waveforms. The generated digital waveform is sent to the Universal Software Radio Peripheral (USRP) [3], where it is converted to analog waveforms by the DA converter and then up-converted to the carrier frequency. On the receiving side, the USRP first down-converts the waveform and then converts the analog baseband waveform to the digital waveform and sends it to the computer. All signal processing is carried by the software, hence the SDR allows great flexibility and convenient debugging. More information about the GNU SDR and USRP can be found at [2, 3].

3.3 Two Key Components

We now discuss two key components in our implementation: the channel estimation and the choice of the processing matrix.

3.3.1 Channel Estimation

To use OSMR, the sender needs to know the channels to determine the processing matrix. Actually, to determine the processing matrix, for receiver i ($i \in \{1,2\}$), only the *channel ratio* defined as $g_i = h_{i2}/h_{i1}$ is needed. As the same channel estimation process is carried out at both receivers, in the following, we consider one receiver and refer to it as receiver i. To estimate g_i, a channel estimation sequence is transmitted at the sender. That is, we let the sender transmit $\{+1, -1, +1, -1, +1, ...\}$ at antenna 1 and transmit $\{-1, -1, -1, -1, ...\}$ at antenna 2. At the receiver, let the received powers from antenna 1 and antenna 2 of the sender be w and v, respectively. Note that $|g_i| = v/w$. Suppose the phase of g_i is ϕ. If the receiver's phase is locked to the phase of antenna 1 of the sender, when antenna 1 is transmitting +1, the received complex symbol should be $[w - v\cos(\phi)] + j[-v\sin(\phi)]$; when antenna 1 is transmitting -1, the received complex symbol should be $[-w - v\cos(\phi)] + j[-v\sin(\phi)]$. Therefore, if the receiver received two consecutive samples denoted as $S_1 = x_1 + jy_1$ and $S_2 = x_2 + jy_2$, where S_1 and S_2 correspond to the symbol when antenna 1 is transmitting +1 and -1, respectively, we have

$$u = \frac{x1 - x2}{2}, \tag{1}$$

and

$$\phi = \tan^{-1}\left(\frac{y1 + y2}{x1 + x2}\right), \tag{2}$$

and

$$v = \frac{-y_1}{\sin(\phi)}. \tag{3}$$

Note that there are two values for ϕ in $[-\pi, \pi]$ that satisfy Equ. 2. The ambiguity is resolved by choosing the one resulting in $v > 0$ in Equ. 3.

However, the receiver's phase will not be locked to the phase of antenna 1, because the receiver is receiving the addition of two signals with different phases. To cope with this, we let the sender transmit the same symbols $\{+1, +1, -1, -1, +1 + 1, ...\}$ at both antennas as training symbols for the phase tracking circuit of the receiver. After the training symbols there are a set of symbols to indicate the beginning of the channel estimation sequence.
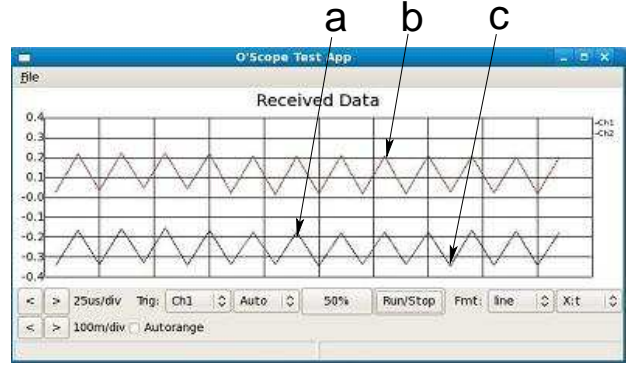

Figure 1. A screenshot of captured channel estimation symbols.

When the receiver receives these symbols, it stops the phase tracking circuit. At this time, the receiver's phase is locked to the symbol when both antenna 1 and antenna 2 are transmitting -1 in the channel estimation sequence. Suppose difference between the phase of the receiver and the phase of antenna 1 of the sender is θ. To estimate θ, suppose the receiver gets two consecutive samples $S_1' = a + jb$ and $S_2' = c$, where S_1' and S_2' correspond to the symbol when antenna 1 is sending +1 and -1, respectively. Note that S_2' does not have an imaginary component, because the receiver's phase is locked to the phase when both antennas at the sender is transmitting -1. Due to the definition of θ, $S_1 = S_1' e^{j\theta}$, $S_2 = S_2' e^{j\theta}$. As the imaginary components of S_1 and S_2 are the same,

$$a\sin(\theta) + b\cos(\theta) = c\sin(\theta), \tag{4}$$

hence,

$$\theta = \tan^{-1}\left(\frac{b}{c - a}\right). \tag{5}$$

After finding θ, S_1 and S_2 can be found, with which u, v, and ϕ can be found. The ambiguity of θ can be resolved by considering the sign of u.

For example, Fig. 1 shows a screenshot of captured channel estimation symbols, where $a = -0.18$, $b = 0.20$, and $c = -0.32$. It can be found that $\theta = -0.31\pi$, $u = 0.12$, $v = 0.27$, $v/u = 2.25$, and $\phi = -0.43\pi$.

3.3.2 Determining the Processing Matrix

The simplest choice of the processing matrix is the inversion of the channel matrix. In our current implementation, we took some extra measures in attempt to further optimize the performance as well as regulating transmitting power. First, to force the interference to be 0, we require

$$\mathbf{h}_1\mathbf{u}_2 = 0, \mathbf{h}_2\mathbf{u}_1 = 0. \tag{6}$$

Second, we require

$$|\mathbf{h}_1\mathbf{u}_1| \geq \eta|h_{11} + h_{12}|, |\mathbf{h}_2\mathbf{u}_2| \geq \eta|h_{21} + h_{22}| \tag{7}$$

where η is a constant. This is to make sure that the effective channels are not too weak compared to the original unprocessed channels. [1] Third, we require

$$|u_{11} + u_{12}| \leq 1, |u_{21} + u_{22}| \leq 1, \tag{8}$$

---

[1] We assume that if the sender has two antennas but does not use OSMR, it transmits the same signal at both antennas with equal power. This assumption was made because if the sender does not use OSMR, it does not know the channels and cannot process the channels using techniques such as *maximum ratio combining* [12]. In fact, in our implementation, the receiver must first get the channel estimation sequence which is trans-

to make sure that the transmitted signal power is within the limit of the transmitter. Note that if the data symbol to be sent to user i is d_i for $i \in \{1,2\}$, the signal sent by antenna i is $u_{i1}d_1 + u_{i2}d_2$. To make sure that each antenna is transmitting at no more than the regulated power, $|u_{i1}d_1 + u_{i2}d_2|$ should be no more than $|d_i|$ which is the transmitting magnitude of antenna i when OSMR is not used. The exact value of $u_{i1}d_1 + u_{i2}d_2$ depends on d_1 and d_2 which are random. However, if this constraint is satisfied, the peak transmitting power is never more than the transmitting power when OSMR is not used.

From Equ. 6, we have $u_{11} = -\frac{h_{22}}{h_{21}}u_{21}$ and $u_{12} = -\frac{h_{12}}{h_{11}}u_{22}$. Substituting $u_{11} = -\frac{h_{22}}{h_{21}}u_{21}$ into the first half of Equ. 7, we have

$$|u_{21}||h_{11}|| - \frac{h_{22}}{h_{21}} + \frac{h_{12}}{h_{11}}| = |u_{21}||h_{11}|| - g_2 + g_1| \ge \eta|h_{11}+h_{12}|,$$

therefore,

$$|u_{21}| \ge \gamma_1 = \eta \mid \frac{1+g_1}{-g_2+g_1} \mid \qquad (9)$$

Similarly,

$$|u_{22}| \ge \gamma_2 = \eta \mid \frac{1+g_2}{-g_1+g_2} \mid \qquad (10)$$

Equ. 9 and Equ. 10 give the minimum magnitude of u_{21} and u_{22}. To determine u_{21} and u_{22}, let the phase difference between u_{21} and u_{22} be δ. The problem then reduces to finding δ such that both of the following inequalities are satisfied:

$$| - g_2\gamma_1 - g_1\gamma_2 e^{j\delta}| \le 1, |\gamma_1 + \gamma_2 e^{j\delta}| \le 1. \qquad (11)$$

In our implementation, we start with $\eta = 0.1$ and first conduct a linear search over $[-\pi, \pi]$ at a step of $\frac{\pi}{360}$ for δ to check if a δ can be found such that both inequalities are satisfied. If no δ can be found, the two receivers are not compatible. Otherwise, we increase the magnitude of u_{21} and u_{22} by 10% of their minimum values and conduct another search. This is continued until no δ can be found and the δ found in the last round is used as the solution.

4 OSMR Experiments

Note that whether an OSMR transmission is successful or not depends on whether the two receivers are compatible at that moment. Because the channel is constantly fluctuating, two receivers may be compatible at some times while not compatible at other times. For OSMR to be applicable to wireless LANs, the percentage of the time when the receivers are compatible must be non-trivial. Therefore, the first question we seek to answer is: how often do the wireless channels allow OSMR transmissions? To answer this question, we conducted experiments with our prototype OSMR transmitter/receiver. In our experiments, the OSMR transmission is centered at 2.42GHz, which lies within the ISM band used by the 802.11b and 802.11g networks. The sender and receiver use the same carrier frequency. Differential Binary Phase Shift Keying (DBPSK) modulation is used and the symbol rate is 500,000 symbols per second, which results in a bit rate at 0.5Mbps. We refer the OSMR sender as S and the two OSMR receivers as $R1$ and $R2$. In the experiment, $R1$ and $R2$ are turned on first. The OSMR transmission is then carried out in three steps:

1. S transmits channel estimation frames for 0.5 second, then switches to listening mode to wait for the channel estimation reports from $R1$ and $R2$.

2. Both $R1$ and $R2$ wait until the S stops sending. Then, $R1$ sends the channel estimation report to S for 0.01 second, then switches to listening mode to wait for the data frames. After S stops sending, $R2$ waits for 0.01 second, then sends the channel estimation report to S for 0.01 second, then switches to listening mode to wait for the data frames.

3. After getting both channel estimation reports, S waits for 0.01 second, then switches to the transmitting mode and sends the data frames for 1 second. One data frame is 1524 bytes with 1500 bytes of randomly generated data and 24 bytes as the frame header.

Our experiments were conducted in a university building. We picked ten sender locations, and for each sender location, we conducted a set of four OSMR experiments at randomly selected receiver locations, where the distances between the sender and the receivers were between 6 to 30 feet. The sender location and the receiver locations in one set of experiments, for example, are shown in Fig. 2. In each experiment, OSMR transmissions were attempted with random intervals between 2 to 5 seconds. Therefore, we basically randomly sample the channels and find the percentage of time the channel allows OSMR transmission. An OSMR transmission is considered successful if the both receivers got the first 3 data frames with no bit error. The compatibility ratio is defined as the number of successful OSMR transmissions over the number of all OSMR transmissions carried out, where an OSMR transmission is carried out if the sender got both channel estimation reports and sent the data frames[2]. We report the results of 35 experiments in which at least 25 OSMR transmissions were carried out and show the cumulative distribution function (c.d.f.) of the compatible ratio in Fig. 3. We can see that roughly, the compatible ratio is uniformly distributed in $[0, 0.9]$. Therefore, this experiment suggests that OSMR transmission is possible in the indoor environments for a significant percentage of time. [3]

As mentioned earlier, another crucial question is the stability of the channel. As the wireless channel fluctuates randomly, before starting the OSMR transmission, the sender must get the channel estimations from the re-

---

[2]With the current GNU SDR, to switch between the transmitting and receiving mode, we have to disconnect a "flow graph" and connect another "flow graph," which could take non-trivial amount of time depending on the instantaneous state of the operating system. It could happen that two receivers send report at the same time, which results in a collision. If the sender did not get the channel estimation reports from both receivers, the sender will abort the transmission. Therefore, not all OSMR transmissions were carried out int full.

[3]Sometimes, wireless receivers can receive the signal from one sender when there are two simultaneous senders, provided that the signal from the sender is significantly larger than the other, known as the *capture effect*. Because OSMR is also transmitting two signal sources simultaneously, to make sure that our OSMR experiments are successful not because of the capture effect, we did a sanity check test in which we used the transpose of the processing matrix in the place of the processing matrix. In such tests, the transmissions almost never succeeded, which confirms that the OSMR transmissions were successful because the signals were processed correctly.

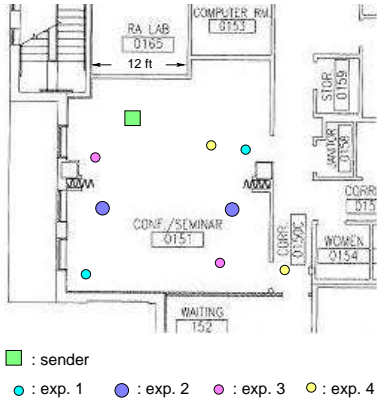mitted without processing the channels because the sender does not know the channels yet.

Figure 2. The sender location and the receiver locations in one set of experiments. The receiver locations are marked as circles, where the diameter of the circle is proportional to the compatibility ratio.
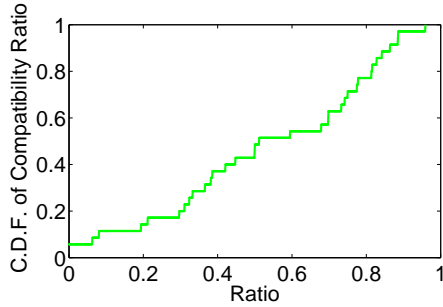

Figure 3. The c.d.f. of compatibility ratio found in the experiments.

ceivers. The sender then uses the estimations to calculate the processing matrix and transmit the frame. Because the sender does not have further feedbacks from the receiver, in order for the OSMR transmission to be successful, the shift of the channel during the frame transmission time must be limited. To find the characteristics of the channel shift in the indoor environments, we conducted experiments to measure the stability of the wireless channels. In our experiments, there were one sender and one receiver, where the sender has two antennas and the receiver has one antenna. Similar to the previous experiment, we picked ten sender locations, and for each sender location, four receiver locations were picked randomly. The sender transmits the OSMR channel estimation sequences every 1ms for a total of 50 seconds, and the receiver simply records the received symbols. With the received symbols, the fluctuation of channel ratio can be derived. If the channel ratio is $ae^{j\phi}$ at time t_0 and is $a'e^{j\phi'}$ at time t_1, the shift of the magnitude is defined as $\frac{|a'-a|}{a} \times 100\%$, and the shift of phase is defined as $|\phi' - \phi|$. The c.d.f. of the channel ratio shift after 1ms, 10ms, 100ms, and 1000ms are shown in Fig. 4. We can see that for more than 90% of the times, after 10ms, the magnitude shifts less than 10%, and phase shifts less than $\frac{\pi}{18}$. Considering that the packet transmission time in a wireless LAN is usually between 0.3ms and 10ms, this result suggests that OSMR is very likely to be applicable to wireless LANs. We can also see that even after 100ms, for more than 80% of the time, the magnitude shifts less than 20%, and the phase shifts less than $\frac{\pi}{12}$.

We must mention that the compatibility ratio depends on the implementation. The compatibility ratio reported in Fig. 3 was obtained by our prototype implementation.
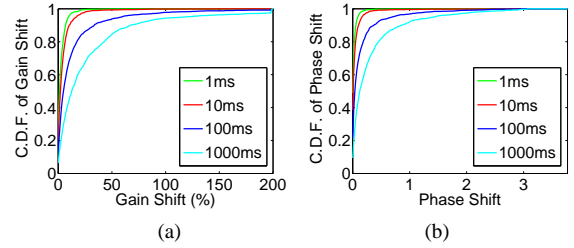

Figure 4. The c.d.f. of channel ratio shift. (a). Magnitude. (b). Phase.

If other implementation is used, the compatibility ratio might be different. Fortunately, it will most likely be higher. The main reason is that in our implementation, the channel estimation process may take more than 20ms by our estimate, where 20ms is needed for the two receivers to send channel estimation reports and the rest may be needed for the reconfiguration of the software. As can be inferred from Fig. 4, after the channel estimation process, the channels may have shifted significantly. Therefore, the successful OSMR transmissions reported in our experiments belong to those cases when the channels allow OSMR transmissions *and* did not shift too much after the channel estimation, which is a subset of the cases when the channels allow OSMR transmissions. A newer version of GNU SDR is under development which will allow the software to specify the exact time when a packet should be transmitted, with which we can reduce the estimation time significantly. In fact, the channel estimation time could be further reduced when implemented in hardware. The estimation should only take in the order of a hundred microseconds, because it only involves exchanging several small packets each of size around several tens of bytes. However, we note that our experiments still serve their purpose for this paper, which is to demonstrate that OSMR transmission is possible in the indoor environments and can be successful for a significant percentage of the time, while the percentage will be even higher if more efficient implementation is used.

5 Backward Compatibility and Application Issues

We believe OSMR can be a useful enhancement to existing 802.11 LANs. To use OSMR, the AP must be upgraded to be OSMR-capable, i.e., must have two antennas and be able to perform channel estimation and signal processing. On the other hand, using OSMR requires minimum change to the receivers. Because the AP makes sure that the signal sent to one receiver appears as zero at the other receiver, and vice versa, the receiver can use the same hardware for decoding the packet. The only change must be made for the receivers is that they must cooperate in channel estimation, which requires stopping the phase-tracking circuit and getting access to the received symbols. Depending on the vendors, the device drivers may or may not have this level of control. If yes, upgrading a receiver to be OSMR-capable requires only updating the device driver. Otherwise, the receiver must change its hardware. Fortunately, OSMR is completely backward-compatible. That is, it is possible for OSMR-capable nodes and OSMR-incapable nodes to coexist in the same LAN. The AP may use OSMR only on OSMR-capable nodes, while use the traditional one-to-one transmission on OSMR-incapable nodes.

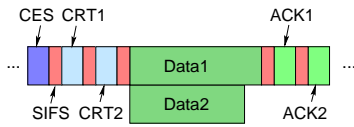In a wireless LAN, if the AP gains access to the

Figure 5. Packet transmission with OSMR. CES: channel estimation sequence. CRT: channel estimation report.

medium and wishes to initiate an OSMR transmission to two nodes, it should first carry out channel estimation to get the instantaneous channel states. It may first send the a packet to notify the two nodes, which also contains the channel estimation sequence. The two involved nodes should reply with the channel estimate report in a pre-determined order. If the AP finds that the two nodes are compatible, it can then start the transmission. After the transmission is completed, the two nodes should send acknowledgment packets back to the AP. The process is illustrated in Fig. 5, where CES denotes the channel estimation sequence and CRT denotes channel estimation report. The complete packet transmission may also include overhead such as DIFS and a possible back-off. Note that if the AP finds that the two nodes are not compatible, it may abort the OSMR transmission and send the packets one by one.

The channel estimation process in Fig. 5 is unique to OSMR and is not needed in one-to-one transmissions. Interestingly, with some slight modifications to the packet transmission scheme, it is likely that the channel estimation will not lead to much overhead, especially when the traffic load is high. The AP may piggyback the channel estimation sequence in every packet it sends, and ask the nodes to piggyback the channel estimation report in the acknowledgment packets. This will not introduce much overhead because the channel estimation sequence can be as few as 16 BPSK symbols, and the channel estimation report is simply the channel ratio which can be packed into less than 4 bytes. If the traffic load of some node is high, as the traffic usually exhibits bursty behavior, it can be expected that the AP may receive the channel estimation reports of this node in a timely manner, e.g., within several milliseconds, such that the channel has not shifted much with very high probability. Therefore, if the AP wishes to send to such nodes, no channel estimation is needed. On the other hand, when the traffic load is low, although the AP cannot get timely updates from the acknowledgment packets, spending time on channel estimation will be not as critical because the medium is not congested.

It is also desired for the AP to keep track of the compatibility relations of nodes in the network, which will prove to be useful for packet scheduling, as well as for avoiding initiating OSMR transmissions to nodes that are not compatible hence wasting the time spent in channel estimation. To achieve this, the AP needs to know the channel states of the nodes. As mentioned earlier, the AP may get piggybacked channel estimation reports from some nodes. For other nodes, the AP may broadcast the channel estimation sequence periodically, say, every 100 ms, and ask the nodes to send back updated channel estimation reports. As explained earlier, this will not introduce large overhead because the channel estimation sequence and channel estimation reports are small. Between two consecutive updates, although the instantaneous channel states of the nodes may have drifted from the most recent updates, the compatibility relations are

very likely the same because the channel fluctuates relatively slowly.

6 Downlink Optimization

In this section we focus on packet scheduling when OSMR is adopted. The packet scheduling is needed because the AP must make smart decisions to "pair up" packets to improve the overall downlink performance. In this section, we focus on maximizing the throughput on the downlink. 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, although may not always give the optimal schedule, but is capable of giving reasonably good schedules in practice.

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 attempt to find an *optimal schedule*, with which the packets in the buffer can be sent in minimum time. The AP then picks a packet or a group of packets according to the schedule it finds as the packet(s) to be sent next.

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. When deriving the algorithm, we focus on the data transmission time and temporarily neglect the overhead because the data transmission time dominates the packet transmission time in most cases. At the end of this section, we will discuss how our algorithm works when the overhead is considered.

Due to the reasons explained in Section 5, the AP is aware of the compatibilities of nodes in the LAN at any given time with high probability. In this section, for simplicity, we assume that the AP knows exactly the compatibilities of nodes. Theoretically speaking, 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. 6(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. 6(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,
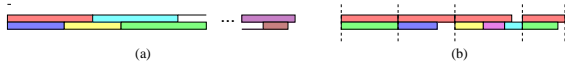
Figure 6. (a). Sending packets in a stream of pairs, which is *not* practical. (b). Examples of group transmissions, where packets shown at the top are the main packets.
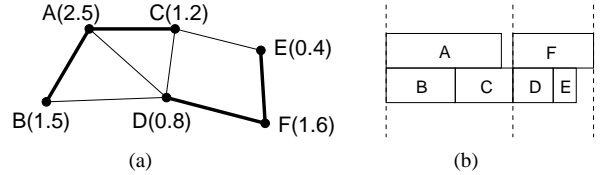

Figure 7. (a). A graph with six vertices where the capacities of the vertices are shown in the parenthesis. The heavy edges belong to a *c*-matching. (b). The packet transmission schedule based on the *c*-matching.

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

We first provide a high-level description of our approach. 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 next. More detailed descriptions are in the following.

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. 7(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. 7, $\{AB, AC\}$ 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. 7(a), which corresponds to the packet transmission schedule shown in Fig. 7(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 7, 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, we have:

LEMMA 1. *A maximum weight c-matching in G corre-*

sponds to an optimal schedule.

Therefore, in the following, we focus on finding a maximum weight *c*-matching in the graph. Note that in the case when all vertices have the same capacity, the problem reduces to finding a maximum matching which still takes $O(n^{2.5})$ time where n is the number of vertices in the graph [14]. 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 2. $C(a) \geq U_M[\phi(a)]$ *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 $\phi(a)$ denote set of edges in M incident to a. We show 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.

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 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 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 the $U_M[\phi(a)]$ is no less than $\frac{1}{1+\sqrt{2}}\{U_{M^*}[\Theta(a)] + \sum_{j=1}^{v} U_{M^*}[\Theta(b_j)]\}$, hence the performance ratio of the algorithm.

We prove this by considering two cases. First, consider when 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 2, $C(b_h) \geq U_{M^*}[\Theta(b_h)]$. Therefore, if the algorithm exits the loop because no edge can be added, we actually have $U_M[\phi(a)] \geq U_{M^*}[\Theta(a)] + \sum_{j=1}^{v} U_{M^*}[\Theta(b_j)]$.

Second, consider when 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 2, $C(a) \geq U_{M^*}[\Theta(a)]$ 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. If only to achieve higher throughput, the AP may simply pick an arbitrary star. The commercial APs may also have to consider issues such as fairness, quality of service, etc. As the packet scheduling algorithms in the commercial APs are not available to us, in this paper, we focus on maximizing throughput. However, Algorithm 1 can serve as a basis for the design of packet scheduling algorithms for the commercial APs when OSMR is supported. 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

$$\frac{T_0 - T_g}{T_0 - T_{o^*}} \geq \frac{1}{1 + \sqrt{2}}. \qquad (12)$$

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 es-

timation 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. 12, we have

$$T_g \leq T_0(1 - \frac{1}{1+\sqrt{2}}) + T_{o*}\frac{1}{1+\sqrt{2}}. \tag{13}$$

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,

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

$$\leq \quad 2 - \frac{1}{1+\sqrt{2}} + 2\alpha. \tag{14}$$

We therefore have the following remark.

REMARK 1. *When overhead is considered, with high probability, the greedy algorithm will give a schedule that takes at most $2 - \frac{1}{1+\sqrt{2}} + 2\alpha$ times the time of the optimal schedule, where α denotes the ratio of the expected overhead over the data transmission time when sending the packets without using OSMR.*

7 Evaluations

To evaluate the performance of our algorithm and OSMR, we conducted simulations using trace data collected from wireless LANs. The trace data used in our simulations is downloaded from [20] collected from 802.11a networks. As we wish to evaluate the packet scheduling algorithm, we used Trace 2 and Trace 3 in [20], 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 [20, 21].

In our simulation, we assumed that on average, two nodes are compatible for α percent of the time, where α is randomly picked in $[0, 0.9]$. Two nodes alternates 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 trace data does not reveal the instantaneous data rate of the nodes, we assumed that all nodes are operating at 54 Mbps, the highest data rate of 802.11a networks. [4] 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, plus what is shown in Fig. 5. In the simulations, the channel estimation process is always simulated, such that it may serve as a lower bound for the performance of OSMR. If the group consists of n packets to v nodes, the packet transmission includes n ACKs but only v channel estimation reports. The durations of DIFS, average backoff time and SIFS are set to be $34\mu s$, $68\mu s$, and $16\mu s$, respectively. The transmission time of the data is assumed to be $20\mu s$ plus the time needed to send the data. The channel estimation sequence is assumed to take

---

[4]This assumption was made first because the network in [20] is in a confined 20m by 20m area, therefore, all nodes are likely to be close to the AP and run at high data rates. Second, obtaining the data rate could be quite difficult because the data rate could change dynamically when running rate adaptation algorithms. Our simulation reveals similar network throughput as that measured in [21].
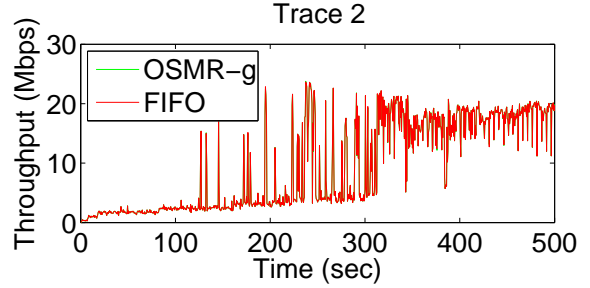

Figure 8. Network throughput in 500 seconds.

$25\mu s$. The channel estimation report and ACK packets are assumed to take $24\mu s$. The values are chosen according to the specifications of 802.11a networks [13].

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.

We refer to our algorithm as OSMR-g. 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 is unlikely to be as simple as FIFO, but should be equivalent 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 [20], 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-g and FIFO in Fig. 8 for one random choice of the compatibilities of the nodes. We can see that both algorithms have almost exactly the same throughput, which is because the traffic load is not high. Note that the upper layer protocols, e.g., TCP, typically probe the capacity of the network to avoid overloading the network, hence the traffic load in the trace data is unlikely to be high enough to reveal the benefit of OSMR because it was collected at an AP not supporting OSMR. However, this simulation does confirm that our simulation set up is correct, because the network throughput in Fig. 8 is very close to that in Fig.1(c) in [21] which is the network throughput measurement for the same trace.

To evaluate the performance of the network at higher traffic load, we processed the trace files and combined the Trace 2 and Trace 3 into one. As each trace contains 75 nodes, to reduce number of nodes, we merged the traffic of 7.5 nodes on average into one node and produced
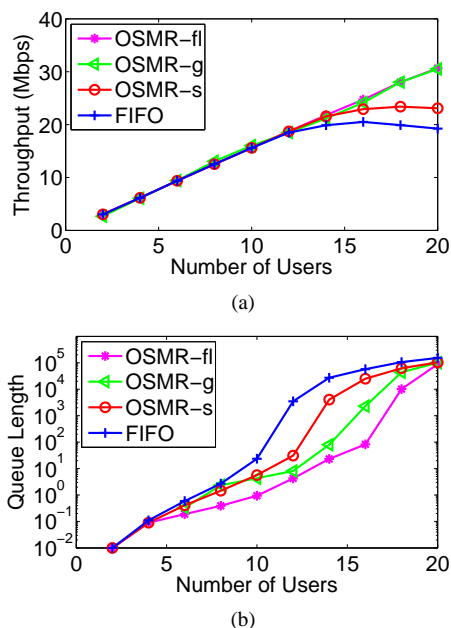
Figure 9. Comparison of different algorithms. (a) Throughput. (b) Queue length.

20 merged nodes. We then randomly select certain number of nodes and use their traffic as input to the simulation, where the number of nodes grows from 2 to 20 at a step of 2. We use the traffic trace from 400 seconds to 500 seconds, when load is more stable. 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. 9(a) and Fig. 9(b), respectively, where each data point was obtained by averaging the results of 100 random seeds. We measure the performance of an algorithm by the *maximum sustainable throughput*, defined as the maximum throughput of the network when the number of packets in the queue is no more than 1000. From Fig. 9 we can see that the maximum sustainable throughput of OSMR-fl, OSMR-g, OSMR-s, and FIFO are about 25Mbps, 22Mbps, 19Mbps, and 16Mbps, respectively. Therefore, OSMR-g is capable of improving the throughput by about 37.5% compared to FIFO. Also, although OSMR-s is better than FIFO, it is outperformed significantly by OSMR-g, which suggests that the greedy algorithm we propose is effective. We can also see that OSMR-fl achieves about 15% higher throughout than OSMR-g, which is the benefit that can be enjoyed with full compatibility compared to a 45% average compatibility.

8 Conclusions

In this paper, we studied the One-Sender-Multiple-Receiver (OSMR) transmission technique which allows a sender to send to multiple receivers on the same frequency simultaneously. We implemented OSMR with GNU Software Defined radio that allows a sender to send to two receivers simultaneously. To the best of our knowledge, this is the first implementation of OSMR. We conducted experiments and tested OSMR transmission in a university building, and our results show that OSMR succeeds for a significant percentage of the time. We also studied the problem of packet scheduling with OSMR. We focused on the problem of maximizing network throughout, and proposed a simple algorithm and prove that it has performance ratio of $\frac{1}{1+\sqrt{2}}$ compared to

the optimal algorithm. We evaluated OSMR and our algorithm with packet traces collected from 802.11a LANs, and the results show that our algorithm significantly improves the throughput.

9 References

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

[2] "Gnu radio - gnu fsf project," *http://www.gnu.org/software/gnuradio.*

[3] Ettus. Inc, "Universal Software Radio Peripheral," *http://ettus.com.*

[4] D. W. Bliss, P. A. Parker and A. R. Margetts, "Simultaneous Transmission and Reception for Improved Wireless Network Performance," *IEEE/SP 14th Workshop on Statistical Signal Processing, 2007. (SSP '07)*, pp. 478-482, Aug. 26-29, 2007.

[5] Z. Zhang and Y. Yang, "Enhancing downlink performance in wireless networks by simultaneous multiple packet transmission," In *Proc. of the 20th IEEE International Parallel and Distributed Processing Symposium (IPDPS '06)*, Rhodes Island, Greece, April 2006.

[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] K. Jamieson and H. Balakrishnan, "PPR: partial packet recovery for wireless networks," In *ACM SIGCOMM 2007.*

[9] G. Woo, P. Kheradpour, D. Shen and D. Katabi, "Beyond the Bits: Cooperative Packet Recovery Using PHY Information," In *ACM MOBICOM 2007.*

[10] S. Katti, S. Gollakota and D. Katabi, "Embracing Wireless Interference: Analog Network Coding," In *ACM SIGCOMM 2007.*

[11] S. Cui, A. J. Goldsmith, and A. Bahai, "Energy-efficiency of MIMO and Cooperative MIMO in Sensor Networks," *IEEE Journal on Selected Areas of Communications,* vol. 22, no. 6, pp. 1089- 1098, Aug. 2004.

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

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

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

[15] 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.

[16] "802.11n: Next-Generation Wireless LAN Technology," *http://80211n.com/white_paper/802_11n-WP100-R.pdf.*

[17] T. Henderson, "802.11g WLAN gear D-Link, Belkin top our early wireless tests," *Network World Global Test Alliance , Network World*, May 12, 2003.

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

[19] M. E. Hazen, "It's Channel Bit Rate As Advertised," *http://rfdesign.com/next_generation_wireless/news/channel_bit_rate* Apr 25, 2006.

[20] http://www.winlab.rutgers.edu/ērgin/mobicom2007/

[21] 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.*

[22] A. Adya, P. Bahl, J. Padhye, A.Wolman and L. Zhou, "A Multi-Radio Unification Protocol for IEEE 802.11 Wireless Networks," In *BROADNETS 2004*, San Jose, CA, October 2004.