Enhancing Downlink Performance in Wireless Networks by Simultaneous Multiple Packet Transmission *

Zhenghao Zhang and Yuanyuan Yang
Department of Electrical and Computer Engineering, State University of New York, Stony Brook, NY 11794, USA

Abstract

In this paper we consider using simultaneous Multiple Packet Transmission (MPT) to improve the downlink performance of wireless networks. With MPT, the sender can send two compatible packets simultaneously to two distinct receivers and can double the throughput in the ideal case. We formalize the problem of finding a schedule to send out buffered packets in minimum time as finding a maximum matching problem in a graph. Since maximum matching algorithms are relatively complex and may not meet the timing requirements of real time applications, we give a fast approximation algorithm that is capable of finding a matching at least $3/4$ of the size of a maximum matching in $O(|E|)$ time where $|E|$ is the number of edges in the graph. We also give analytical bounds for maximum allowable arrival rate which measures the speedup of the downlink after enhanced with MPT and our results show that the maximum arrival rate increases significantly even with a very small compatibility probability. We also use an approximate analytical model and simulations to study the average packet delay and our results show that packet delay can be greatly reduced even with a very small compatibility probability.

Index Terms: Multiple packet transmission, wireless LAN, matching, approximation algorithm, maximum allowable arrival rate, delay.

1 Introduction

Wireless access networks have been more and more widely used in recent years since comparing to the wired networks, wireless networks are easier to install and use. Due to the tremendous practical interests, much research effort has been devoted to wireless access networks and great improvements have been achieved in the physical layer by adopting newer and faster signal processing techniques, for example, the data rate in 802.11 wireless LAN (Local Area Network) has increased from $1Mbps$ in the early version of 802.11b to $54Mbps$ in 802.11a [8]. We have noted that in addition to increasing the point to point capacity, new signal processing techniques have also made other novel transmission schemes possible which can greatly improve the performance of wireless networks. In this paper, we study Multiple Packet Transmission (MPT), with which the sender can send more than one packets to distinct users simultaneously.

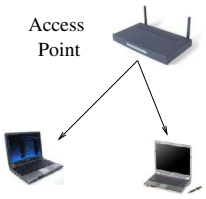


Figure 1. Multiple packet transmission – The access point can send two packets to two users simultaneously.

Traditionally, in wireless networks, it is assumed that one device can send to only one other device at a time. However, this restriction is no longer true if the sender has more than one antennas. By processing the data according to the channel state, the sender can make the data for one user appear as zero at other users such that it can send distinct packets to distinct users simultaneously. We call it Multiple Packet Transmission (MPT) and will explain the details of it in Section 2. For now, we want to point out the profound impact the MPT technique has on wireless LANs. A wireless LAN is usually composed of an Access Point (AP) which is connected to the wired network and several users which communicate with the AP through wireless channels. In wireless LANs, the most common type of traffic is the downlink traffic, i.e., from the AP to the users when the users are browsing the Internet and downloading data. In today's wireless LAN, the AP can send one packet to one user at a time. However, if the AP has two antennas and if MPT is used, the AP can send two packets to two users whenever possible, thus doubling the throughout of the downlink in the ideal case.

MPT is feasible for the downlink because it is not difficult to equip the AP with two antennas, in fact, many wireless routers today have two antennas. Another advan-

*The research work was supported in part by NSF grant numbers CCR-0207999 and ECS-0427345 and ARO grant number W911NF-04-1-0439.

1-4244-0054-6/06/$20.00

tage of MPT which makes it very commercially appealing is that although MPT needs new hardware at the sender, it does not need any new hardware at the receiver. This means that to use MPT in a wireless LAN, we can simply replace the access point and upgrade software protocols in the user devices without having to change their wireless cards, and thus incurring minimum cost.

In this paper we study problems related to MPT and provide our solutions. We formalize the problem of sending out buffered packets in minimum time as finding a *maximum matching* in a graph. Since maximum matching algorithms are relatively complex and may not meet the speed of real time applications, we consider using approximation algorithms and present an algorithm that finds a matching with size at least $3/4$ of the size of the maximum matching in $O(|E|)$ time where $|E|$ is the number of edges in the graph. We then study the performance of wireless LAN enhanced with MPT and give analytical bounds for maximum allowable arrival rate. We also use an analytical model and simulations to study the average packet delay.

Enhancing wireless LANs with MPT requires the Media Access Layer (MAC) to have more knowledge about the states of the physical layer and is therefore a form of cross-layer design. In recent years cross-layer design in wireless networks has attracted much attention because of the great benefits in breaking the layer boundary. For example, [5, 6] considered packet scheduling and transmission power control in cross-layer wireless networks. However, to the best of our knowledge, packet scheduling in wireless networks in the context of multiple packet transmission has not been studied before. [3, 4] have considered Multiple Packet Reception (MPR) which means the receiver can *receive* more than one packets from distinct users simultaneously. MPR is quite different from MPT since MPR is about receiving multiple packets at one node while MPT is about sending multiple packets from one node to multiple nodes.

2 Multiple Packet Transmission

In this section we briefly explain the MPT technique. As mentioned earlier, to use MPT, the sender makes the data sent for one user appear as zero at other users. This is possible if the sender has more than one antennas. With multiple antennas, the sender can adjust the amplitude and phase of the transmitted signals on different antennas such that the signals will add up constructively or destructively as desired. The following is a more detailed explanation which follows Chapter 10 in [1].

Wireless channels can be modeled as complex baseband channels, which means that with one antenna at the sender, the receiver will receive $y = h^*d$, where d is the complex data sent by the sender and h is the complex channel coefficient. The receiver can recover the data by dividing y by h. Note that here we consider flat fading which means that there is no inter-symbol-interference, and do not consider noise so that the core idea of MPT can be more easily seen.

If there are two antennas at the sender, the sender can send two different symbols denoted as x_1 and x_2 on antenna 1 and antenna 2, respectively. If there are two receivers, receiver 1 will receive $y_1 = h_{11}^*x_1 + h_{12}^*x_2$ and receiver 2 will receive $y_2 = h_{21}^*x_1 + h_{22}^*x_2$, where h_{ij} is the channel coefficient from antenna j to user i. For simplicity we will use $\mathbf{h}_i$ to denote $[h_{i1}, h_{i2}]^T$ and use $\mathbf{x}$ to denote $[x_1, x_2]^T$, and call them the channel coefficient vector and the transmitted vector, respectively. In vector forms, receiver i will receive $y_i = \mathbf{h}_i^*\mathbf{x}$.

Now let d_1 and d_2 denote the data that should be sent to receiver 1 and receiver 2, respectively. We cannot simply send d_1 via antenna 1 and d_2 via antenna 2 because the data will be mixed up at the receivers. However, suppose there are vectors $\mathbf{u}_1 = [u_{11}, u_{12}]^T$ and $\mathbf{u}_2 = [u_{21}, u_{22}]^T$ such that $\mathbf{h}_1^*\mathbf{u}_2 = 0$ and $\mathbf{h}_2^*\mathbf{u}_1 = 0$. We can let transmitted vector be $\mathbf{x} = d_1\mathbf{u}_1 + d_2\mathbf{u}_2$, that is, send $d_1u_{11} + d_2u_{21}$ on antenna 1 and $d_1u_{21} + d_2u_{22}$ on antenna 2. Thus receiver 1 will receive $\mathbf{h}_1^*(d_1\mathbf{u}_1 + d_2\mathbf{u}_2) = d_1\mathbf{h}_1^*\mathbf{u}_1$, and similarly receiver 2 will receive $d_2\mathbf{h}_2^*\mathbf{u}_2$, thus distinct data is sent to each receiver.

$\mathbf{u}_1$ can be any vector lies in V_1 which is the space orthogonal to $\mathbf{h}_2$, however, to maximize the received signal strength, $\mathbf{u}_1$ should lie in the same direction as the projection of $\mathbf{h}_1$ onto V_1. $\mathbf{u}_2$ should be similarly chosen. Since the total transmitted power is limited, not all pairs of receivers are *compatible*, i.e., can use MPT. Basically, the sender should choose two receivers if their channel coefficient vectors are already near orthogonal.

To perform MPT, the sender needs 4 more complex multipliers. It also needs to know the channel coefficient vectors of the receivers and run algorithms to smartly pair up the receivers. However, the receivers needs no additional hardware and can receive the signal as if the sender is only sending to it. It is also possible to send to more than 2 receivers at the same time if the sender has more than 2 antennas. In this paper we focus on the more practical 2-antenna case. Also note that MPT requires wireless channels to be slow changing as compared to the data rate, which is often true in a wireless LAN where the wireless devices are stationary for most of the time.

3 MAC Layer Modifications

In this section we describe the modifications to MAC layer protocol, in particular, 802.11, to support MPT. We say two users U_1 and U_2 are compatible if they can receive at the same time. If U_1 and U_2 are compatible, sometimes we also say that the packets destined for U_1 and U_2 are compatible.

The AP keeps the channel coefficient vectors of all nodes that have been reported to it previously. If, based on the past

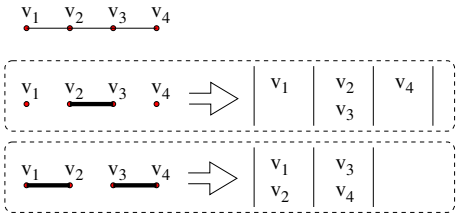


Figure 2. **4 packets and different schedules.**

channel coefficient vectors, U_1 and U_2 are compatible and there are two packets that should be sent to them, the AP sends out a RTS (Require To Send) packet, which contains, in addition to the traditional RTS contents, a bit field indicating that the packet about to send is a MPT packet. If U_1 appears earlier than U_2 in the destination field, upon receiving the RTS packet, U_1 will first reply a CTS (Clear To Send) packet containing the traditional CTS contents plus its latest channel measurements. After a short fixed amount of time U_2 will also reply a CTS packet. After received the two CTS packets, the AP will update the channel coefficient vectors. It will then decide whether U_1 and U_2 are still compatible and most likely they still are since the environment is slow changing. If in the rare case that the channels have changed significantly such that they are no longer compatible, the AP can choose to send to only one node. Therefore, before sending the data packets, the AP first sends 2 bits in which bit i is '1' means the packet for U_i will be sent for $1 \leq i \leq 2$. After the data packet is sent, U_1 and U_2 can reply an acknowledgment packet in turn.

In this paper we consider matching user packets of the same size. For simplicity, we consider the case when the bit rates are also the same, such that all packets need the same transmission time which we call a time slot. We do not make any assumption about the compatibilities of users and treat them as arbitrary.

4 Scheduling Algorithms for Access Points

While the idea of MPT is simple, the AP will encounter the problem of how to match the packets with each other to send them out as fast as possible. For example, suppose in the buffer of the AP there are 4 packets destined for 4 users denoted as v_1, v_2, v_3 and v_4, respectively. Assume packet v_i is compatible with v_{i+1} for $1 \leq i \leq 3$, as shown at the top of Fig. 2 where there is an edge between two packets if they are compatible. If we match v_2 with v_3, the 4 packets have to be sent in 3 time slots since v_1 and v_4 are not compatible. However, a better choice is to match v_2 with v_1 and match v_3 with v_4 and send the 4 packets in only 2 time slots. When the number of packets grows the problem of finding the best matching strategy will become harder. In this section we describe algorithms that solve this problem.

4.1 Algorithm for Optimal Schedule

We call a schedule by which packets can be sent out in minimum time an *optimal* schedule. Clearly, in an optimal schedule maximum number of packets are sent out in pairs, therefore the problem of finding an optimal schedule is equivalent to finding maximum number of compatible pairs among the packets. To solve this problem, as shown in Fig. 2, we draw a graph G where each vertex represents a packet and two vertices are adjacent if the two packets are compatible. In a graph, a *matching* M is defined as a set of vertex disjoint edges, that is, no edge in M has a common vertex with another edge in M. Therefore the problem reduces to finding a maximum matching in G. For example, the second matching in Fig. 2 is a maximum matching while the first one is not. Maximum matching in a graph can be found in polynomial time by algorithms such as the Edmonds' Blossom Algorithm which takes $O(N^4)$ time, where N is the number of vertices in the graph [10, 2].

Before continuing our discussion, we first give definitions of some terms. Let M be a matching in a graph G. We call edges in M the "matching edges." If a vertex is incident to an edge in M, we say it is "saturated;" otherwise, it is "unsaturated" or "free" or "single." An M-augmenting path is defined as a path with edges alternating between edges in M and edges not in M, and with both ends being unsaturated vertices. For example, with regarding to the first matching in Fig. 2, $v_1 - v_2 - v_3 - v_4$ is an augmenting path. It is well known in graph theory that the size of a matching can be incremented by one if and only if there can be found an augmenting path.

The buffer of the AP may store many packets, as a result, the graph can be quite large. However, the size of the graph can be reduced by taking advantage of the fact that vertices represent packets for the same user have exactly the same set of neighbors in the graph. More specifically, in the graph, we say vertex u and v belong to the same equivalent group, or simply the same group, if the packets they represent are for the same user. Vertices that belong to the same group have the same neighbors and are not adjacent to each other. Let $A = \{a_1, a_2, a_3\}$ and $B = \{b_1, b_2, b_3\}$ be two groups of vertices and suppose a_i is matched to b_i for $1 \leq i \leq 3$. We have

Lemma 1 *If there is an augmenting path traversing all 3 matching edges between A and B, there must exist an augmenting path traversing only 1 matching edge between A and B.*

Proof. This can be best explained with the help of Fig. 3, where edges in the matching are shown as heavy lines and edges not in the matching are shown as dashed lines. As in the figure, suppose an augmenting path traversing all three matching edges between A and B is $x - a_1 - b_1 - c - d - b_2 - a_2 - e - f - a_3 - b_3 - y$. However, if x is adjacent

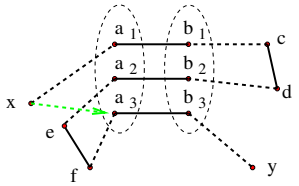


Figure 3. "shortcut" exists. The matching edges are shown as heavy lines and the non-matching edges are shown as dashed lines.

to a_1, it must also be adjacent to a_3 since a_1 and a_3 belong to the same group, thus there is a shorter augmenting path traversing only to the last matching edge between A and B which is $x - a_3 - b_3 - y$. (Note that the same proof also holds if in the augmenting path, the segment between, say, b_1 and b_2, is longer.) ■

As a result of this lemma, if there exists an augmenting path, there must also exist an augmenting path traversing no more than two matching edges between any two groups of vertices. This is because if the path traverses more than two matching edges between two groups of vertices, as we have shown in the lemma, there must be a shortcut by which we need only to traverse the last of the first three matching edges, and we can keep on finding such shortcuts and reducing the number of traversed matching edges until it is less than 3. Therefore, for any two groups of vertices, only two matching edges between them need to be kept and other redundant matching edges can be removed. After that there will be $O(n^2)$ saturated vertices left where n is the number of users. Also note that for the purpose of finding augmenting paths, only one of the unsaturated vertices belonging to each group needs to be considered. Therefore, the graph we work on contains $O(n^2)$ number of vertices which does not depend on the size of the buffer.

4.2 Practical Considerations

Although the optimal schedule can be found for a given set of packets by the maximum matching algorithm, in practice, the packets do not arrive all at once but arrive one by one. It is not feasible to run the maximum matching algorithm every time a new packet arrives due to the relatively high complexity of the algorithm. Therefore, after a new packet arrives, we can match it according to the following simple strategy: A new vertex is matched if and only if it can find an unsaturated neighbor. In this way we always maintain a *maximal* matching, where a matching M is maximal in G if no edge not belonging to M is vertex disjoint with all edges in M. For example, the two matchings in Fig. 2 are all maximal matchings. The maximum matching algorithm can be called only once a while to augment the existing maximal matching.

Another problem is that the packets do not stay in the buffer forever and must be sent out. We will have to make the decisions of which packet(s) should be sent out once the AP has gained access to the media and there is a delicate tradeoff between throughput and delay. To improve the throughput, we should always send out packets in pairs; however, this policy favors the packets that can be matched over the packets that cannot be matched, and will increase the delay of the latter. To prevent excessive delay of the single packets, in practice, we can keep a time stamp for each packet and if the packet has stayed in the buffer for a time longer than a threshold, it will be sent out the next time the AP has gained access to the media. If there are multiple such packets, the AP can choose one randomly. The threshold can be determined adaptively based on the measured delays of the packets that were sent out in pairs.

Finally, although maximum matching can be found in polynomial time, maximum matching algorithms are in general complex [11] and may not meet the timing requirements of real time applications, considering that the processors in the AP are usually cheap and not powerful. Therefore in some cases, a fast approximation algorithm which is capable of finding a "fairly good" matching may be useful, which will be discussed next.

4.3 A Linear Time $3/4$ Approximation Algorithm for Finding Maximum Matching

The simplest and most well known approximation algorithm for maximum matching simply returns a maximal matching. It is known that this simple algorithm has $O(|E|)$ time complexity where $|E|$ is the number of edges in the graph and has a performance ratio of $1/2$, which means that the matching it finds has size at least half of M^* where M^* denotes the maximum matching. In this section we give a simple $O(|E|)$ approximation algorithm for maximum matching with an improved ratio of $3/4$. To the best of our knowledge it is the first linear time approximation algorithm for maximum matching with $3/4$ ratio.

The idea of our algorithm is to eliminate all augmenting paths of length no more than 5. Note that any M-augmenting path must have i edges in M and $i+1$ edges not in M for some integer $i \geq 0$. Therefore if the shortest M-augmenting path has length at least 7, $\frac{|M|}{|M^*|} > 3/4$, since to increment the size of the matching by one, the "trade ratio" is at least $3/4$, i.e., the best we can do is to take out 3 edges in M and add in 4 edges not in M.

A maximal matching does not have augmenting paths of length 1, which is why the size of a maximal matching is at least half of the size of a maximum matching. In our algorithm, we will start with a maximal matching and then eliminate augmenting paths of length 3 and then of length 5. As can be seen, the algorithms themselves are simple and straightforward. However, it is interesting and somewhat surprising that they can be implemented to run in linear time.

Table 1. Finding Augmenting Paths of Length 3

```
Initially, M = S where S is a maximal matching.
for i = 1 to |S| do
    Let (u, v) be the i^th edge in S.
    Check if u and v have distinct unsaturated neighbors.
    if yes
        Let the neighbors of u and v be x and y, respectively.
        M ← M ∪ {(x, u), (v, y)} \ {(u, v)}.
    end if
end for
```

4.3.1 Eliminating Augmenting Paths of Length 3

We start with a maximal matching denoted by S and the output of our algorithm is denoted by M. For each vertex, a list is used to store its neighbors. An array is used to store the matching, that is, the i^{th} element in the array is the vertex matched to the i^{th} vertex. Note that with this array, it takes constant time to augment the matching with fixed length augmenting paths or to check whether a particular vertex is saturated or not.

The algorithm is summarized in Table 1. Initially, let $M = S$. We will check edges in S from the first to the last to augment M. When checking edge (u, v), we check whether both u and v are adjacent to some distinct unsaturated vertices. If there are such vertices, say, u is adjacent to x and v is adjacent to y, there is an M-augmenting path of length 3 involving (u, v) which is $x - u - v - y$. We can eliminate this augmenting path and augment M by removing (u, v) from M and adding (u, x) and (v, y) to M. We call (u, x) and (v, y) the new matching edges. The algorithm terminates when all edges in S have been checked this way.

Next we prove the correctness and derive the complexity of this algorithm. It is important to note that if the matching is always augmented according to augmenting paths, the following two facts always hold. First, all saturated vertices will remain saturated after each augmentation. Second, as a result of the first fact, if a vertex is unsaturated after an augmentation, it must be unsaturated before the augmentation and therefore throughout the process M remains a maximal matching.

Lemma 2 *The new matching edges need not to be checked because there cannot be augmenting paths of length 3 involving them.*

Proof. To see this, suppose $x - u - v - y$ is a length-3 augmenting path, as shown in Fig. 4. If there is a length-3 augmenting path involving one of the new matching edges, say, (x, u), let it be $s - x - u - t$, as shown in the right part of Fig. 4. Note that s and x are not saturated before the matching is augmented according to $x - u - v - y$, which contradicts the fact that the matching is always maximal. The left part of Fig. 4 shows this situation. ■

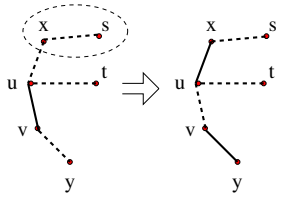


Figure 4. Augmenting M according to $x - u - v - y$. s and x are both unsaturated which contradicts the fact that M is a maximal matching.

Corollary 1 *When the algorithm terminates, there is no length-3 augmenting path.*

Proof. By contradiction. If there is still a length-3 augmenting path, let it be $x - u - v - y$ where u and v are saturated. By Lemma 2, (u, v) cannot be a new matching edge, therefore it is in S. But this cannot happen since if such an augmenting path exists after the algorithm terminates, it must also exist when (u, v) was checked and should have been found. ■

Lemma 3 *The algorithm runs in $O(|E|)$ time where $|E|$ is the number of edges.*

Proof. Note that when checking edge (u, v), the edges incident to u and v were checked at most once. Since the edges in S are vertex disjoint, the algorithm checks an edge in G no more than twice. ■

Combining these discussions, we conclude that

Theorem 1 *The algorithm in Table 1 eliminates all length-3 augmenting paths in $O(|E|)$ time.*

4.3.2 Eliminating Augmenting Paths of Length 5

After eliminating augmenting paths of length 3, we search for augmenting paths of length 5. We first check all edges in the current matching to construct a set T. A vertex v is added to set T if v is matched to some vertex u and u is adjacent to at least one unsaturated vertex. We call v an "outer vertex" and u an "inner vertex." Note that v can be both an outer vertex and an inner vertex when v and u are both adjacent to the same unsaturated vertex and are not adjacent to any other unsaturated vertices. Clearly, to find augmenting paths of length 5 is to find adjacent outer vertices. Also note that T can be constructed in $O(|E|)$ time.

The algorithm is summarized in Table 2 and works as follows. We check the vertices in T from the first to the last. When checking vertex v, let u be the inner vertex matched to v. We first get or update $l(u)$ which is the list of unsaturated neighbors of u: If $l(u)$ has not been established earlier, we search the neighbor list of u to get $l(u)$; otherwise, we check the vertex in $l(u)$ (in this case, there can only be one vertex in $l(u)$, for reasons to be seen shortly) and remove it

from $l(u)$ if it has been matched. After getting $l(u)$, if $l(u)$ is empty, we quit checking v, remove v from T and go on to the next vertex in T. Otherwise, we check the neighbors of v to find an outer vertex. If an outer vertex w is found to be adjacent to v, let z be the inner vertex matched to w. We get $l(z)$ which is the unsaturated neighbor list of z in the same way as for u. If $l(z)$ is empty, we remove w from T and go on to the next neighbor of v. Otherwise, we check if there is an augmenting path of length 5 involving (u, v) and (w, z), and note that this can be in constant time. This is because (1) if $l(z)$ contains at least 2 vertices, there must be such a path; (2) if $l(z)$ contains exactly 1 vertex, there is such a path if and only if $l(u)$ is different from $l(z)$. If an augmenting path is found, we augment M according to this path and remove both v and w from T; otherwise, we continue to check the next outer vertex neighbor of v. If all neighbors of v have been checked and no augmenting path is found, we remove v from T and continue to the next vertex in T. Now we can see why if an outer vertex is still in T after it has been checked, the unsaturated neighbor list of the inner vertex matched to it must contain exactly one vertex. This is because if it contains more than 1 vertices, an augmenting path must have been found when checking this outer vertex and it would have been removed from T. The algorithm terminates when T is empty. Note that this algorithm makes sure that it will find an augmenting path of length 5 involving (u, v) if such a path exists when checking outer vertex v. Also note that removing an element in a set is equivalent to marking this element which takes constant time.

Recall that if M is augmented by augmenting paths, M remains to be a maximal matching which means that it does not have any augmenting path of length 1. The next lemma shows that by augmenting M by length-5 augmenting paths, there will never be "new" augmenting paths of length 3.

Lemma 4 *Throughout the execution of the algorithm no augmenting paths of length 3 will be created.*

Proof. By contradiction. Since M does not have length-3 augmenting paths at the beginning, suppose the first such a path was created after augmenting M with length-5 augmenting path $a-b-c-d-e-f$. The length-3 augmenting path must involve one of the new matching edges which are (a, b), (c, d) and (e, f). If (c, d) creates an augmenting path of length 3, say, $u-c-d-v$, as shown in the right part of Fig. 5(a), there must exist augmenting path $a-b-c-u$ which has length 3, as shown in the left part of Fig. 5(a), which contradicts the fact that there is no such a path before the matching was augmented. Thus (c, d) cannot create an augmenting path of length 3. If (a, b) creates an augmenting path of length 3, say, $x-a-b-y$, as shown in the right part of Fig. 5(b), x and a must be both unsaturated before the matching was augmented, as shown in the left part of Fig. 5(b), which contradicts the fact that the matching is always maximal. Hence (a, b) cannot create augmenting path of length 3 and for the same reason neither can (e, f). ■

Table 2. Finding Augmenting Paths of Length 5

```
Construct T, the set of outer vertices.
while T is not empty
    Let v be a vertex in T that has not been checked.
    Suppose v is matched to u. Get or update l(u),
    the unsaturated neighbor list of u.
    if l(u) is empty
        Remove v from T and continue to the next outer
        vertex in T.
    end if
    while not all neighbors of v have been checked
        Let w be an outer vertex neighbor of v and suppose
        w is matched to z.
        Get or update l(z), the unsaturated neighbor list of z.
        if l(z) is empty
            Remove w from T and continue to the next
            neighbor of v.
        end if
        Based on l(u) and l(z), determine if there is a
        length-5 augmenting path.
        if an augmenting path is found
            Augment M according to this path and
            remove both v and w from T;
            break from the inner while loop.
        end if
    end while
    if no augmenting path is found
        Remove v from T
    end if
end while
```

Lemma 5 *Throughout the execution of the algorithm the new matching edges cannot be involved in any augmenting path of length 5.*

Proof. Suppose during the execution of the algorithm, the first time that a new matching edge becomes involved in a length-5 augmenting path is after we augment the matching according to augmenting path $a-b-c-d-e-f$. We first show that (c, d) cannot be involved in augmenting path of length 5 by contradiction. If there is such an augmenting

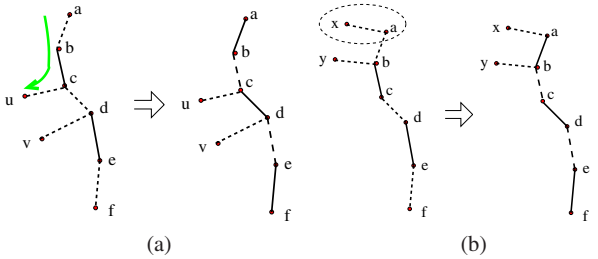


Figure 5. No new augmenting paths of length 3 will be created.

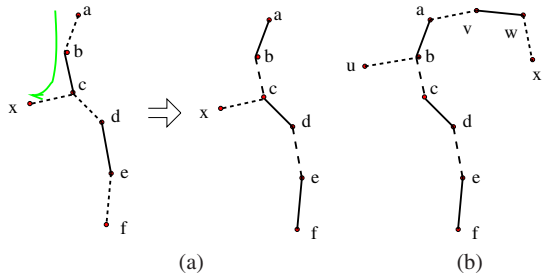


Figure 6. No new matching edge will be involved in augmenting paths of length 5.

path, there must be an unsaturated vertex adjacent to either c or d. Let the unsaturated vertex be x and without loss of generality due to symmetry, suppose x is adjacent to c, as shown in Fig. 6(a). Then there is an augmenting path of length 3 before the algorithm started: $a-b-c-x$, which contradicts the fact that there are no such paths.

We now show that (a, b) cannot be involved in augmenting path of length 5. The same proof can be used for (e, f) due to symmetry. First note that since the matching is maximal, a cannot be adjacent to an unsaturated vertex. Therefore the augmenting path must be in the form of $u-b-a-v-w-x$, as shown in Fig. 6(b). We claim that edge (v, w) cannot be an "old matching edge," i.e., cannot be in the matching before the algorithm started. Since if so, $a-v-w-x$ is an augmenting matching of length 3 before the algorithm started. Therefore (v, w) is also a new matching edge. Suppose edge (v, w) was first added to the matching by augmenting the matching according to a length-5 augmenting path P. Since no new matching edge was involved in any length-5 augmenting path before the matching was augmented according to $a-b-c-d-e-f$, edge (v, w) was not involved in any length-5 augmenting path except P. Note that due to the same reason, since (c, d) cannot be in any length-5 augmenting path, (v, w) cannot be the edge in the center of P, and thus (v, w) must be at the end of P. Since w is adjacent to an unsaturated vertex x, w cannot be the end vertex of P, thus v must be the end vertex of P. However, this means that both a and v were unsaturated before the algorithm started, which cannot happen since the matching is maximal. (Note that the proof also holds if (v, w) is (c, d) or (f, e).) ■

Corollary 2 *When the algorithm terminates, there is no augmenting path of length 5.*

Proof. Suppose it is not true, that is, there still exists an augmenting path of length 5, say, $a-b-c-d-e-f$. By Lemma 5, neither (b, c) nor (d, e) is a new matching edge. However, in this case the algorithm must have found this augmenting path. ■

Lemma 6 *The algorithm runs in $O(|E|)$ time.*

Proof. Consider checking an outer vertex v in set T. Note that all time needed for checking v is $O(d(v))$ where $d(v)$ is the degree of v except for getting the unsaturated neighbor lists for some inner vertices. Therefore overall the algorithm runs in $O(|E|)$ time plus the time needed for getting the unsaturated neighbor lists for inner vertices which also takes $O(|E|)$ time since it needs to be done for each inner vertex no more than once. Thus, the lemma follows. ■

Combining the above discussions, we conclude that

Theorem 2 *The algorithm in Table 2 eliminates all length-5 augmenting paths in $O(|E|)$ time.*

5 Performance Study

In this section we study the performance of the wireless LAN after enhanced by MPT. We first derive the maximum arrival rate of the downlink and then study the average packet delay by an analytical model and simulations.

The performance of a wireless network depends on many factors, for example, the physical environment, the locations of the wireless nodes, etc., such that the performance of one network could be different from that of another even when they are using the same devices. In many cases the performance of the same network may also be changing due to the occasional movements of the wireless nodes. This makes the performance evaluation in general a difficult job. However, we note that the performance gain of adopting MPT is mainly determined by the probability of two nodes being compatible, and this probability should be the same in networks under similar environments and with same devices. It is thus more insightful to use the compatibility probability p as the parameter for performance evaluation. For simplicity, we assume that the probability that two users are compatible is independent of other users.

5.1 Maximum Arrival Rate

The first and the most important question is: After using MPT, how much faster does the downlink become? This can be measured by the maximum allowable arrival rate, where an arrival rate is allowable if it does not cause the buffer of the AP to overflow. More specifically, suppose once the AP has got access to the media, on average it has to wait T seconds to be able to get access to the media again. In the following, for convenience, we refer to T as a time slot. The normalized arrival rate λ is defined as the average number of packets arrived in a time slot. Without MPT, clearly, $\lambda_{max} = 1$ where λ_{max} denotes the maximum allowable arrival rate. Next we derive the value of λ_{max} when MPT is used.

Suppose there are n users among which c users are compatible with some other users. These c users are called the "non-isolated" users and the rest are called the "isolated users." Consider W arrived packets. Assuming packets have random destinations, there will be $W(c/n)$ packets for

the non-isolated users and $W(1-c/n)$ packets for the isolated users. The fastest way to send out these W packets is to always send out the packets for the non-isolated users in pairs, thus the minimum time needed to send out all the packets is $W(1-c/2n)$ time slots. In other words, W packets should arrive in at least $W(1-c/2n)$ time slots. Thus, the maximum arrival rate for given n and c is $(1-c/2n)^{-1}$.

The number of non-isolated users is a random variable. Let $P_n(l)$ be the probability that out of n users, there are l isolated users. The average maximum arrival rate is

$$\lambda_{max} = \sum_{c=0}^{n} P_n(n-c)(1-c/2n)^{-1}$$

Therefore in the following we focus on finding $P_n(l)$.

Apparently, when $n = 1$, $P_1(0) = 0$ and $P_1(1) = 1$; when $n = 2$, $P_2(0) = p$, $P_2(1) = 0$ and $P_2(2) = 1 - p$ where p is the compatibility probability. To find $P_n(l)$ for larger n, we condition on the number of isolated users among the first $n-1$ users. Let $E_{x,y}$ be the event that in the x users, y are isolated and let L be random variable denoting the number of isolated users among n users.

$$P_n(l) = P(L = l) = \sum_{i=0}^{n-1} P_{n-1}(i)P(L = l|E_{n-1,i})$$

Clearly, for $i < l - 1$,

$$P_n(L = l|E_{n-1,i}) = 0,$$

since by adding a user, we can add at most one isolated user. For $i = l - 1$,

$$P_n(L = l|E_{n-1,l-1}) = (1-p)^{n-1},$$

since given there are $l-1$ isolated users among the $n-1$ users, there are l isolated users in the n users if and only if the n^{th} user is isolated, which occurs with probability $(1-p)^{n-1}$. For $i = l$, we have

$$P_n(L = l|E_{n-1,l}) = [1-(1-p)^{n-1-l}](1-p)^l,$$

since if there are already l isolated users in the $n-1$ users, the n^{th} user must not be isolated, i.e., must be compatible with some user in the first $n-1$ users. However, it cannot be compatible with any of the isolated among the $n-1$ users, since this will reduce the number of isolated users, thus it must be compatible with at least one of the users among the $n-1-l$ non-isolated users, which is an event that occurs with probability $[1-(1-p)^{n-1-l}](1-p)^l$. For $i > l$,

$$P_n(L = l|E_{n-1,i}) = \begin{pmatrix} i \\ i-l \end{pmatrix} p^{i-l}(1-p)^l,$$

since if $i > l$, the addition of the n^{th} user reduces the number of isolated users by $i-l$, thus it must be compatible with exactly $i-l$ previous isolated users.

Fig. 7 shows the maximum arrival rate for networks of different sizes under different compatibility probabilities.

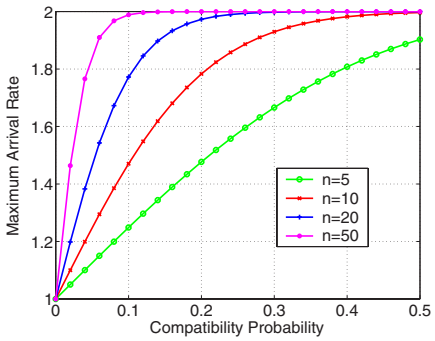


Figure 7. Maximum arrival rate for networks of different sizes under different compatibility probabilities.

It is remarkable to see that a significant improvement can be achieved even with very small compatibility probability. For example, for $n = 10$, when $p = 0.04$, the maximum arrival rate is 1.2, which is a 20% increase.

Finally, we want to argue that the maximum arrival rate is approximately achievable, although it is at the cost of excessive delay for the isolated users. Note that as mentioned earlier, the maximum arrival rate is achieved if packets destined for non-isolated users are always sent out in pairs and if no time slot is wasted, i.e., there is always at least one packet sent out in a time slot. Therefore, if there are compatible packet pairs in the buffer, we send the pair; otherwise, we send packets destined for the isolated users and keep on doing so until a new pair has formed after some new packets have arrived. Since at a high arrival rate the queues for the isolated users are most likely quite long, it is highly likely that we can wait until a pair appears before the queues for the isolated users are exhausted.

5.2 Average Packet Delay

As we have seen, adopting MPT can greatly increase the maximum allowable arrival rate. Note that MPT can also reduce the queuing delay of the packets comparing to Single Packet Transmission (SPT). In this section we use an analytical model along with simulations to see how packet delay can be reduced.

5.2.1 An Approximation Analytical Model

We first describe our analytical model. The model is developed for the purpose of comparing MPT with SPT and therefore only considers arrival rates less than 1. We assume that the AP maintains n queues in its buffer, one for each user. Note that to exactly model the behavior of the queues in the AP, many MAC layer related issues have to be considered, for example, how often can the AP gain access to the media and how many packets will arrive at the AP in a given time period, etc. All such issues are interacting with each other which makes exact analytical modeling very difficult. We therefore use some approximations

to simplify the model. As the simulations show, our model is very accurate when $\lambda < 1$.

The model is based on Markov chains. We take the total number of packets stored in the buffer before the AP has gained access to the media, which we will later refer to as the *AP sending* for convenience, as the state of the Markov chain. The advantage of doing so is that the Markov chain becomes discrete-time since we are only looking at the buffer at some discrete time instants. We will assume that between two AP sendings, the number of packets arrived at the AP follows the widely used Poisson distribution, that is, $P_a(K = k) = e^{-\lambda}\lambda^k/k!$, where K is the random variable denoting the number of arrived packets. It should be noted that our model is not limited to Poisson distribution and can also be used if the arrival follows other distributions. We also assume that the arrived packets have random destinations. Note that we have avoided explicitly dealing with the complex issue of how often can the AP gain access to the media, because it has been encapsulated in the assumption of the arrival distribution. That is, if the AP has to wait longer to access the media, we can choose a large λ since more packets can be expected to arrive and otherwise we can choose a small λ since less packets can be expected to arrive.

To more accurately model the queues, we also consider whether there exists a pair of compatible packets in the buffer. Therefore in our model, we use (b, r) as the state of the Markov chain, where b is the total number of packets, and $r = 0$ means that there is no compatible pair and $r = 1$ otherwise. We assume that if there exists a compatible pair the AP will always send it, since when the arrival rate is small, most likely the packets will not be delayed longer than the threshold. Thus, the transition probability for $(b, 0)$ is

$$(b, 0) \rightarrow (b - 1 + k, 0) : P_1 P_a(k)$$

where P_1 is the probability that given there is no compatible pair in the $b - 1$ packets left in the buffer, there is no compatible pair after k new packets have arrived, and, clearly,

$$(b, 0) \rightarrow (b - 1 + k, 1) : (1 - P_1)P_a(k).$$

Similarly, the transition probability for $(b, 1)$ is

$$(b, 1) \rightarrow (b - 2 + k, 0) : P_2 P_a(k)$$

where P_2 is the probability that given there was a compatible pair, after sending out the pair and after receiving k new packets, there is no compatible pair. Also,

$$(b, 1) \rightarrow (b - 2 + k, 1) : (1 - P_2)P_a(k).$$

Therefore in the following we need only to focus on finding P_1 and P_2. Note that since we have used only two random variables to model n queues, some information is lost, and the model is only an approximation model in the sense that the Markovian property only holds approximately. However, this is necessary since if the queues are considered separately, the complexity of the model will be exponential.

To find P_1 and P_2, we will make the assumption that the packets stored in the buffer have random destinations. Note that this is not true since the AP favors packets destined non-isolated users, and as a result, in the buffer, there will be more packets destined for isolated users than for the non-isolated users. However, this assumption makes the analytical modeling tractable and yields remarkably accurate results when $\lambda \leq 1$.

Let F_x be the event that among x packets there is no compatible pair. We have

$$P_1 = P(F_{b-1+k}|F_{b-1}) = \frac{P(F_{b-1+k}, F_{b-1})}{P(F_{b-1})} = \frac{P(F_{b-1+k})}{P(F_{b-1})}$$

since if there is no compatible pair in the $b - 1 + k$ packets, there cannot be a compatible pair in any subset of it, in particular, the $b - 1$ packets. P_2 is harder to find than P_1, since we do not know after sending out a compatible pair, whether there is still a compatible pair in the $b - 2$ packets. We make the assumption that there is no compatible pair in the $b - 2$ packets, since when λ is not large, the packets can be sent out rather swiftly, and we can safely assume that once a compatible pair is formed, it will be immediately sent out. With this assumption, similar to P_1, we have $P_2 = P(F_{b-2+k})/P(F_{b-2})$.

To find $P(F_x)$, we first find $P_U(x, s)$ which is the probability that knowing that x packets are for s users, there is at least one packet for each of the s users, i.e., none of the queues for the s users is empty. Clearly, $P_U(x, 1) = 1$ for all x and $P_U(2, 2) = 1/2$. For larger x and s, observe that if given t packets are for the first user, the event that none of the s queues is empty occurs if and only if the rest $x - t$ packets make the rest $s - 1$ queues all non-empty, which occurs with probability $P_U(x - t, s - 1)$. Thus, we have the recursive relation

$$P_U(x, s) = \sum_{t=1}^{x-s+1} P_U(x - t, s - 1) \left(\begin{array}{c} x \\ t \end{array} \right) (1/s)^t (1 - 1/s)^{x-t}$$

Let $P_W(x, s)$ be the probability that if there are totally x packets in the buffer, there are exactly s non-empty queues among the total n queues. We have

$$P_W(x, s) = \left(\begin{array}{c} n \\ s \end{array} \right) (s/n)^x P_U(x, s),$$

since there are $\left(\begin{array}{c} n \\ s \end{array} \right)$ ways to choose s queues from n queues and for any given s queues, this event occurs if and only if the x packets are all for the s users which occurs with probability $(s/n)^x$ and if the x packets make the s queues all non-empty which occurs with probability $P_U(x, s)$.

After obtaining $P_W(x, s)$, $P_F(x)$ is simply

$$P(F_x) = \sum_{s=1}^{n} P_W(x,s)(1-p)^{s(s-1)/2},$$

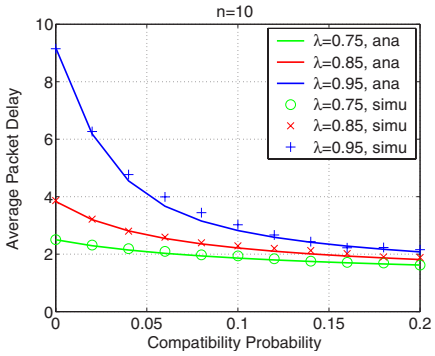
since the probability that there is no compatible pair among s users is $(1-p)^{s(s-1)/2}$.

5.2.2 Analytical and Simulation Results

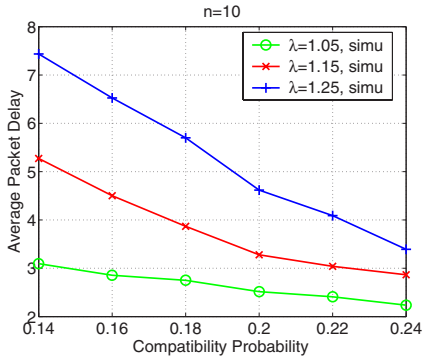
We also conducted simulations to verify our analytical model. In our simulations, each point is obtained by running on 100 random topologies and each topology is run for $100,000$ rounds. Fig. 8(a) shows the average packet delay as a function of compatibility probability when $\lambda < 1$ for $n = 10$ obtained by simulations and the analytical model. First we observe that the analytical results are very close to the simulation results. Second we observe that MPT greatly reduces the average delay even when p is very small. We also observe that the average delay decreases faster when p is smaller and will tend to converge to a value when p further increases. Fig. 8(b) shows the average packet delay obtained by simulations when $\lambda > 1$. Similarly, we can observe that increasing p will always reduce the delay. We have used large p in Fig. 8(b) than in Fig. 8(a) because when $\lambda > 1$, small p occasionally results in too separated topologies which causes the buffer that has limited size in our simulations to be unstable.

6 Conclusions

In this paper we considered using Multiple Packet Transmission (MPT) to improve the downlink performance of the wireless LANs. With MPT, the access point can send two compatible packets simultaneously to two distinct users. We have formalized the problem of finding a minimum time schedule as a matching problem, and have given a practical linear time algorithm that finds a matching at least $3/4$ the size of a maximum matching. We studied the performance of wireless LAN after enhanced with MPT. We gave analytical bounds for maximum allowable arrival rate which measures the speedup of the downlink and our results show that the maximum arrival rate increases significantly even with a very small compatibility probability. We also used an approximate analytical model and simulations to study the average packet delay and our results show that packet delay can be greatly reduced even with a very small compatibility probability.



(a) $\lambda < 1$



(b) $\lambda > 1$

Figure 8. Average packet delay as a function of compatibility probability under different arrival rates when there are 10 users. "ana" and "simu" stand for "analytical" and "simulation," respectively.

References

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

[2] D.B. West, "Introduction to graph theory," *Prentice-Hall*, 1996.

[3] T. Lang, V. Naware and P. Venkitasubramaniam, "Signal processing in random access," *IEEE Signal Processing Magazine*, vol.21, no.5, pp.29-39, 2004.

[4] G. Dimic, N. D. Sidiropoulos and R. Zhang; "Medium access control - physical cross-layer design," *IEEE Signal Processing Magazine*, vol.21, no.5, pp.40-50, 2004.

[5] Q. Liu, S. Zhou and G.B. Giannakis, "Cross-layer scheduling with prescribed QoS guarantees in adaptive wireless networks," *JSAC*, vol.23, no.5, pp.1056-1066, 2005.

[6] V. Kawadia and P.R. Kumar, "Principles and protocols for power control in wireless ad hoc networks," *JSAC*, vol.23, no.1, pp.76-88, 2005.

[7] A Czygrinow, M. Hanckowiak and E. Szymanska, "A fast distributed algorithm for approximating the maximum matching," *Algorithms - ESA 2004, Lecture Notes in Computer Science 3221*, pp.252-263, 2004.

[8] http://grouper.ieee.org/groups/802/11/.

[9] W. Xiang; T. Pratt and X. Wang; "A software radio testbed for two-transmitter two-receiver space-time coding OFDM wireless LAN," *IEEE Communications Magazine*, vol.42, no.6, pp.S20-S28, 2004.

[10] J. Edmonds, "Paths, trees, and flowers," *Canad. J. Math*, 17:449-467, 1965.

[11] H. N. Gabow, "An efficient implementation of Edmonds' algorithm for maximum matching on graphs," *Journal of the ACM (JACM)*, vol.23, no.2, pp.221-234, 1976.

[12] J. Magun, "Greedy matching algorithms: An experimental study," *Proc. 1st Workshop on Algorithm Engineering*, pp.22-31, 1997.