International Journal of Research
Available at https://edupediapublications.org/journals
Special Issue on Conference Papers
e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 05 Issue 06
March 2018

The One-Sender-Multiple-Receiver Technique and Downlink Packet Schedulingby Simultaneous Multiple Packet Transmission

[1]Boddupelli Sravanthi(15861A0401) [2]Putta Laxmi(15861A0408) [3]Puttala Mounika (15861A0409)
[4]Sandupatla Anila (15861A0412) [5]Erra Nagamani(15861A0403)
Students of Dept. Electronics & Communication Engineering
Mother Theressa College Of Engineering and Technology, Peddapalli,India
E-Mail:putta.laxmi00@gmail.com[1]

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 OðjEjÞ time, where jEj 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

1 INTRODUCTION

WIRELESS access networks have been more and more widely used in recent years, since compared 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 Local Area Network (LAN) has increased from 1 Mbps in the early version of 802.11b to 54 Mbps 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 a novel Multiple-Input, Multiple-Output (MIMO) technique called Multiple Packet Transmission (MPT) [1], with which the sender can send more than one packet to distinct users simultaneously (Fig. 1). 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- tage of MPT that 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 AP 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

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 antenna. 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 MPT and will explain the details of it in Section 2. For now, we want to point out the profound impact of MPT technique on wireless LANs. A wireless LAN is usually composed of orthogonal.

To perform MPT, the sender needs four 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 need 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 two receivers at the same time if the sender has more than two antennas. In this paper, we focus on the more practical two- antenna case. Also note that MPT requires wireless channels to be slowly 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 the 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 record for the channel coefficient vectors of all nodes that have been reported to it previously. If, based on the past channel coefficient vectors, U_1 and U_2 are likely to be compatible and there are two packets that should be sent to them, the AP sends out a Require To Send (RTS) packet, which contains, in addition to the traditional RTS contents, a bit field indicating that the packet about to send is an MPT packet. If U_1 appears earlier than U_2 in the destination field, upon receiving the RTS packet, U_1 will

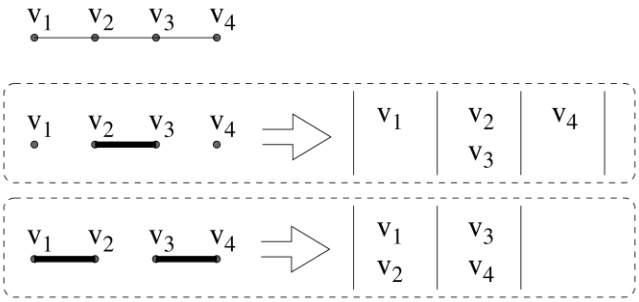

Fig. 2. Four packets and different schedules.

first reply a Clear To Send (CTS) packet containing the traditional CTS contents plus its latest channel measure- ments. After a short fixed amount of time, U_2 will also reply a CTS packet. After receiving the two CTS packets, the AP will update their channel coefficient vectors. It will then decide whether U_1 and U_2 are still compatible, and if so, the AP will send two packets to them. 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 i 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. As measurement study [13] shows, typical packets in a wireless LAN are of two sizes, the data packets of size around 1,500 bytes and the acknowledgment packets of size around 40 bytes. Because MPT involves the overhead of RTS/CTS packet exchange, it is most efficient for the data packets. In this paper, for simplicity, we consider the case when the data rates of the users are the same. When the data rates are the same, all data packets takes roughly the same amount of time to transmit, which will be referred to as a time slot. We do not make any assumption about the compat- ibilities 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 four packets destined for four 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 i 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 four packets have to be sent in three 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 four packets in only two time slots. When the number of packets grows, the problem of finding the best matching strategy will become more difficult. 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, the maximum number of packets is sent out in pairs; therefore, the problem of finding an optimal schedule is equivalent to finding the 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 the 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 "M-saturated" or simply "saturated;" otherwise, it is "M-unsaturated" (unsaturated) or "M-free" (free) or "sin- gle." 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 regard 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 that represent packets for the same user have exactly the same set of neighbors in the graph. More specifically, in the graph, we say vertices 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 i 3. We have the following lemma:

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

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 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 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 Oðn2 Þ 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 Oðn2 Þ 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 a packet 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 that 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 OðjEjÞ time complexity, where jEj is the number of edges in the graph and has a performance ratio of 1/2, which means that the matching it finds has a size at least half of M where M denotes the maximum matching. In this section, we give a new OðjEjÞ approximation algorithm for maximum match- ing with an improved performance 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 Þ 1 edges not in M for some integer i 0. Therefore, if the shortest M - augmenting

path has length at least 7, $\frac{jMj}{j} > 3=4$, since to increment the j j 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 three edges in M and add in four 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 a 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.

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 ith element in the array is the vertex matched to the ith 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 ¼ S. We will check edges in S from the first to the last to augment M . When checking edge ðu; vÞ, 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 ðu; vÞ, which is x u v y. We can eliminate this augmenting path and augment M by removing ðu; vÞ from M and adding ðu; xÞ and ðv; yÞ to M . We call ðu; xÞ and ðv; yÞ the new matching edges. To find x and y, we can first search all neighbors of u and let x be the first M -free neighbor of u. We will temporarily assign x to u and mark

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.
    $M \leftarrow M \cup \{(x, u), (v, y)\} \setminus \{(u, v)\}$.
  end if
end for

as M -saturated. Clearly, if we cannot find an M -free neighbor of u, we can quit checking ðu; vÞ and go on to check the next edge in S. Otherwise, we search all neighbors of v and let y be the first M -free neighbor of v. If such y is found, the augmenting path is found. If otherwise, that is, if we cannot find an M -free neighbor of v, we cannot simply quit checking edge ðu; vÞ since v may be adjacent to x, which has been temporarily assigned to u, while u may be adjacent to some other M -free vertices. Therefore, if v is adjacent to x, we will "assign" x to v and search the neighbors of u that have not been previously searched. By doing this, we make sure that at the time when checking ðu; vÞ, if there is a length-3 augmenting path involving ðu; vÞ, it can be found. 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 augmenta- tion, 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 augment- ing path, as shown in Fig. 4. If there is a length-3 augmenting path involving one of the new matching edges, say, ðx; uÞ, 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. □

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

Proof. By contradiction, if there is still a length-3 augment- ing path, let it be x u v y, where u and v are saturated. By Lemma 2, ðu; vÞ 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 ðu; vÞ was checked and should have been found. □

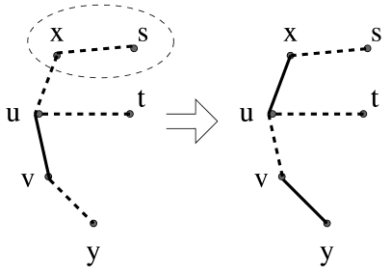

Fig. 4. Augmenting M according to x u v y. s and x are both unsaturated, which contradicts the fact that M is a maximal matching.

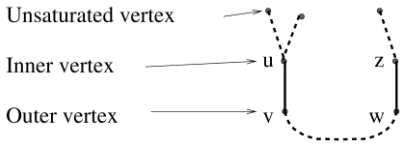

Fig. 5. Inner vertices and outer vertices.

Lemma 3. The algorithm runs in OðjEjÞ time, where jEj is the number of edges.

Proof. Note that when checking edge ðu; vÞ, 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 the above discussions, we have Theorem 1.

Theorem 1. The algorithm in Table 1 eliminates all length-3 augmenting paths in OðjEjÞ 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," as shown in Fig. 5. 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 OðjEjÞ 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 obtain or update lðuÞ, which is the list of unsaturated neighbors of u: If lðuÞ has not been established earlier, we search the neighbor list of u to get lðuÞ; otherwise, we check the vertex in lðuÞ (in this case, there can only be one vertex in lðuÞ, for reasons to be seen shortly) and remove it from lðuÞ if it has been matched.

After getting lðuÞ, if lðuÞ is empty, we quit checking v, remove v from T, and go on to the next vertex in T.

6 CONCLUSIONS

In this paper, we have considered using MPT to improve the downlink performance of the wireless LANs. With MPT, the AP 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 of at least 3/4 the size of a maximum matching. We studied the performance of wireless LAN after it was 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 probabil- ity. 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.

REFERENCES

[1] D. Tse and P. Viswanath, Fundamentals of Wireless Communication. Cambridge Univ. 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, Sept. 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, Sept. 2004.

[5] Q. Liu, S. Zhou, and G.B. Giannakis, "Cross-Layer Scheduling with Prescribed QoS Guarantees in Adaptive Wireless Networks," IEEE J. Selected Areas in Comm., vol. 23, no. 5, pp. 1056-1066, May 2005.

[6] V. Kawadia and P.R. Kumar, "Principles and Protocols for Power Control in Wireless Ad Hoc Networks," IEEE J. Selected Areas in Comm., vol. 23, no. 1, pp. 76-88, Jan. 2005.

[7] A. Czygrinow, M. Hanckowiak, and E. Szymanska, "A Fast Distributed Algorithm for Approximating the Maximum Match- ing," Proc. 12th Ann. European Symp. Algorithms (ESA '04), pp. 252-263, 2004.

[8] http://grouper.ieee.org/groups/802/11/, 2008.

[9] W. Xiang, T. Pratt, and X. Wang, "A Software Radio Testbed for Two-Transmitter Two-Receiver Space-Time Coding OFDM Wire- less LAN," IEEE Comm. Magazine, vol. 42, no. 6, pp. S20-S28, June 2004.

[10] J. Edmonds, "Paths, Trees, and lowers," Canadian J. Math., vol. 17, pp. 449-467, 1965.