Simple and Regular Mini-Slot Scheduling for IEEE 802.16d Grid-based Mesh Networks

Jia-Ming Liang, Jen-Jee Chen, Ho-Cheng Wu, and Yu-Chee Tseng
Department of Computer Science,
National Chiao-Tung University, Hsin-Chu, 30010, Taiwan
Email: {jmliang, chencz, jero1103, yctseng}@cs.nctu.edu.tw

***Abstract*—This work addresses the mini-slot scheduling problem in IEEE 802.16d *wireless mesh networks (WMNs)*. A practical mini-slot scheduling needs to take into account following issues: the transmission overhead, the scheduling complexity, and the signaling overhead to notify the scheduling results to subscriber stations. We focus in a grid-based WMN, which is the most recommended topology due to its high capacity and connectivity. In this paper, we propose scheduling schemes featured by low complexity and low signaling overhead. The proposed schemes help find periodical and regular schedules, which can balance between transmission overhead and pipeline efficiency. They can achieve near-optimal transmission latencies. Simulation results show that our schemes outperform other schemes, especially when the network size is larger.**

***Index Terms*—IEEE 802.16, routing tree, mini-slot scheduling, WiMAX, wireless mesh network.**

I. Introduction

The IEEE 802.16 standard [1] has been proposed to achieve wide-range wireless broadband access. The standard is based on a common *MAC (medium access control)* protocol compliant with different physical-layer specifications. The protocol supports the *point-to-multipoint (PMP)* and the *mesh* modes. The IEEE 802.16 *wireless mesh networks (WMNs)* are widely selected as wireless backbones for broadband data access [2]. In a WMN, the base station (BS) is directly connected to the wired backhaul to provide its subscriber stations (SSs) Internet access. These SSs can be connected to the BS in a multi-hop manner. The wireless access of WMNs follows a *TDMA (time division multiple access)* based MAC protocol, built on the *OFDM (orthogonal frequency division multiplexing)* physical layer. In [2], it has been shown that TDMA has better throughput than *CSMA/CA (carrier sense multiple access with collision avoidance)* used in IEEE 802.11 networks [3], especially when the network is in heavy traffic load. In such WMNs, scheduling is a critical issue that may significantly impact the system performance. It involves constructing a *scheduling tree* from the network and planning wireless resources for SSs to send/receive their data to/from the BS.

The wireless resource on each link in an IEEE 802.16d WMN is a sequence of fixed-length time slots, called *mini-slots*. However, before actually transmitting on a mini-slot, a sender must wait for a fixed number of mini-slots, called *transmission overhead*, to avoid collisions [4], [5]. This is a guard time to synchronize and tolerate the air-propagation delay of the transmission occurring on the mini-slot right before the aforementioned overhead mini-slots. Once starting its actual transmission, a node may send on several consecutive mini-slots. References [6], [7] observe that if the (actual) transmission is too short, most of the time will be occupied by the transmission overhead. On the other hand, if the transmission is too long, it may hurt fairness and pipeline efficiency (i.e., there could be less concurrent transmissions in the pipelines). So, a good scheduling should balance between the ratio of transmission overhead and the pipeline efficiency by adjusting the sizes of (actual) transmissions. In this work, we propose to use three metrics to evaluate a scheduling scheme (i) the total latency (i.e., the time to deliver all data to BS), (ii) the scheduling complexity, and (iii) the signaling overhead (i.e., the cost to notify all SSs their schedules).

In the literature, several works [8], [9] have studied the scheduling problem in WMNs. Reference [8] considers that each transmission can transmit one piece of data and tries to maximize pipeline efficiency to minimize the total transmission time. However, the results in [8] are not optimal when the transmission overhead is non-zero. Considering transmission overheads, [9] proposes to always find the maximal number of concurrent transmission links in each round. This problem has been shown to be NP-hard [10]. Although it performs close to optimum, its computational complexity is too high to be used by the BS. Also, the signaling overhead incurred by [9] is quite high because the scheduling patterns for SSs are not regular.

In this paper, given the uplink loads of all SSs in a WMN, we consider the problem of scheduling their traffics such that the total latency to transmit all data to the BS is minimized and the scheduling complexity and signaling overhead are as low as possible. (The scheduling in the downlink direction is similar, so we focus in only one direction.) In our approach, we first try to find the optimal transmission size for the given loads to strike a balance between the ratio of transmission overhead and the pipeline efficiency. We observe that when the actual transmission size is small, the pipeline could be full for the most of the time, but the transmission overhead could occupy too much time. On the other hand, when the actual transmission size is too large, the above problem may be fixed, but the pipelines may not be filled with sufficient concurrent transmissions, thus hurting spatial reuse. We then assign the transmissions of each link in a periodic and regular manner with a proper transmission size. Since our scheduling is periodical, the signaling overhead to inform each SS becomes low. To the best of our knowledge, our work is the first one with these properties. Our schemes incur low complexity and the approach is applicable to most regular topologies, such as chain and grid networks, which have been proved to have many applications and outperform random topologies in terms of their achievable network capacity, connectivity maintenance capability, and coverage-to-node ratios (about two times that of random topologies) [11]. We remark that the chain topology is a special case of grid topologies, which is the most suitable for long-thin areas, such as railways and highways [8]. Simulation results are provided to verify our claims on these topologies.

The rest of this paper is organized as follows. Section II formally defines our mini-slot scheduling problem. Section III presents our schemes. Simulation results are given in Section IV. Section V concludes this paper.

II. Problem Definition

We are given one BS and n SSs, $SS_i, i = 1..n$. These BS and SSs are deployed in a chain and grid topologies, as shown in Fig. 1. The BS only can be placed at the end point of the topology. All nodes share the same communication channel. The amount of data that a node can transmit per mini-slot is d bytes. Since the topology is regular, two nodes are allowed to transmit concurrently if they are at least H hops away from each other. We consider the uplink scheduling. So we abstract the uplink mini-slots of the system by concatenating them together into an infinite sequence and ignore the downlink mini-slots. Each SS_i has a traffic demand of p_i bytes. Our goal is to construct a scheduling tree $\mathcal{T}$ such that each SS_i receives a collision-free schedule T_i and the total time to deliver all SSs' data to the BS is as less as possible. In our work, we impose that the schedule T_i for each SS_i should be periodical as defined below.

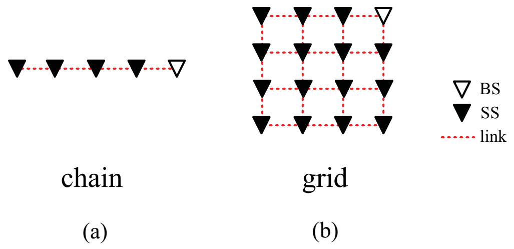

Fig. 1. (a) A 5-node chain topology and (b) a 4×4 grid topology.

The transmission schedule is formulated as follows. For each SS_i and each mini-slot, we use a character in $\{0, 1, h\}$ to represent its state. A '0' means that the mini-slot is idle for SS_i. A '1' means that SS_i can transmit at most d bytes in this mini-slot. An 'h' means that SS_i is preparing to transmit (i.e., this mini-slot is considered a transmission overhead). To start an actual transmission, a SS must wait for α mini-slots of state 'h' so as to synchronize and tolerate the air-propagation time of the transmission occurring right before the overhead mini-slots, where α is a system-dependent constant. For example, when $\alpha = 2$, we can use a string '$000hh1111$' to indicate that a SS is idle in the first three mini-slots, waits for two overhead mini-slots, and then transmits for four mini-slots.

In this work, we enforce that all SSs' transmission schedules are periodical and regular. Specifically, all SSs' schedules have the same of period of ρ. Each SS's transmission schedule has the format of $(0^a h^\alpha 1^b 0^c)^*$, where $a \geq 0$, $b > 0$, and $c \geq 0$ are integers and $a + \alpha + b + c = \rho$. Symbol '$*$' means a number of repetitions of the string in parentheses until all necessary data is delivered. Different SSs may have different patterns. For example, Fig. 2 shows a chain network with one BS and seven SSs. Only SS_7 has a traffic demand of $p_7 = 4$ bytes. Assuming $\alpha = 1$, $d = 1$, and $H = 3$, we show three schedules. In the first schedule, $b = 1$ mini-slot of data is transmitted in each cycle. The other parameters $a = 0/2/4$ and $c = 4/2/0$, respectively. So there are three types of schedule patterns: $(h10000)^*$, $(00h100)^*$, and $(0000h1)^*$. In the second schedule, $b = 2$ mini-slots in each cycle; $a = 0/3/6$ and $c = 6/3/0$, respectively. In the third schedule, $b = 4$; $a = 0/5/10/15/20/25/30$ and $c = 30/25/20/15/10/5/0$, respectively (however, only one cycle is needed). The second schedule is the most efficient. Our goal is to find the most efficient regular schedules.

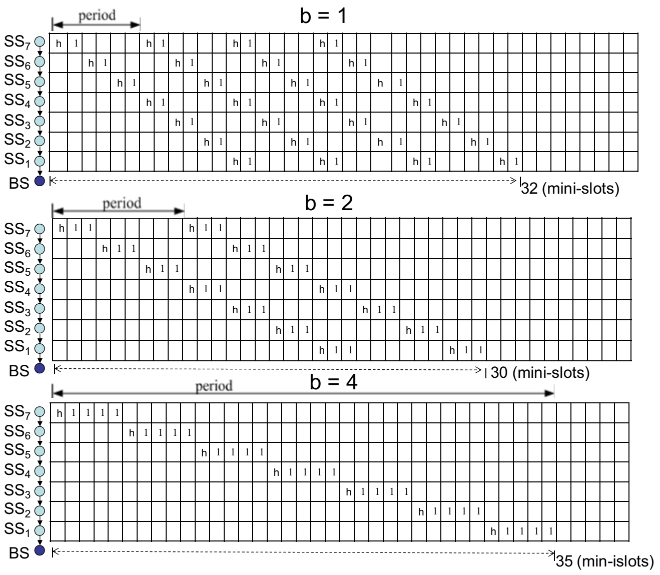

Fig. 2. Transmission schedules for nodes in a chain network (idle state '0' is omitted in the drawing).

III. Scheduling and Tree Construction Schemes

Next, we present our scheduling schemes for chain and grid topologies. We first consider the chain topology with different locations of source SSs on the chain. Then we use these results as basic components to solve the scheduling problem for grid topology. Given a grid network, we first construct a comb-like tree. The comb-like tree is decomposed into individual chains, each of which can be scheduled using the previous chain solutions. Below, we present two solutions for a chain, from simpler to more complicated cases. Then, we combine these solutions for the grid topology.

A. A Chain with A Single Request

Since there are only one source and one destination, we can model the chain, without loss of generality, as a path $SS_n \to SS_{n-1} \to ... \to SS_1 \to BS$ such that only SS_n has a non-zero demand p_n. To increase parallelism, we partition these SSs into k concurrent transmission-able groups, where $k = H$ if $n \geq H$ and $k = n$ otherwise (recall that H is the least spatial-reuse distance). Specifically, we define group G_j, $j = 0..k-1$, as follows:

$$G_j = \{SS_i| \ (n-i) \bmod k = j, i = 1..n\}. \tag{1}$$

Nodes in the same group have the same schedule. We simply denote by T_j the transmission schedule of G_j. Since we are interested in having regular schedules, we enforce each G_j to have a schedule of the format $(0^{a_j} h^{\alpha} 1^b 0^{c_j})^*$, where a_j and c_j are group-specific constants and b is a fixed constant for all groups, such that the following conditions hold: (i) $a_0 = 0$, (ii) $a_j + \alpha + b + c_j = \rho$ is a constant and ρ is the period for all groups, and (iii) $a_j + \alpha + b = a_{j+1}, j = 0..k-2$. Conditions (iii) means that each G_{j+1} is obtained from G_j by shifting the latter to the right by $(\alpha + b)$ positions. Given any b, we can compute the total latency $L_1(n, p_n, b)$ to deliver SS_n's data to the BS:

$$L_1(n, p_n, b) = \begin{cases} \lceil \frac{p_n}{b \cdot d} \rceil \cdot H \cdot (\alpha + b) & \\ \quad + (n - H) \cdot (\alpha + b), & \text{if } n \geq H \\ \lceil \frac{p_n}{b \cdot d} \rceil \cdot n \cdot (\alpha + b), & \text{otherwise.} \end{cases} \tag{2}$$

When $n \geq H$, each cycle has a length of $H \cdot (\alpha + b)$ mini-slots. It takes $\lceil \frac{p_n}{b \cdot d} \rceil$ cycles for SS_n to transmit its last piece of data. At the end of the $\lceil \frac{p_n}{b \cdot d} \rceil$th cycle, the last piece of SS_n's data will arrive at node SS_{n-H}. Then it takes another $(n-H)$ hops, each requiring $(\alpha + b)$ mini-slots, to travel to the BS. This gives the upper term in Eq. (2). For the lower term, the derivation is similar.

Given fixed n, p_n, and α, we are interested in knowing the optimal value of b, denoted by $\hat{b}$, that gives the minimum latency L_1. To do so, we need to confine that p_n is divisible by $b \cdot d$ in Eq. (2). To minimize Eq. (2), we can let $L_1 = 0$ and take the first-order derivative of b. This leads to

$$\hat{b} = \begin{cases} \sqrt{\frac{\alpha \cdot p_n \cdot H}{d(n-H)}}, & \text{if } n \geq H \\ \frac{p_n}{d}, & \text{otherwise.} \end{cases} \tag{3}$$

The value of $\hat{b}$ in Eq. (3) is a real. The best value may appear in $\lceil \hat{b} \rceil$ or $\lfloor \hat{b} \rfloor$. Plugging this into Eq. (2), we can get the minimum $\hat{L_1}$.

B. A Chain with Multiple Requests

Next, we consider a path $SS_n \to SS_{n-1} \to ... \to SS_1 \to BS$ with multiple non-zero-load nodes. Without loss of generality, we assume SS_n's load is non-zero. Similar to Sec. III. A, we divide SSs into k groups $G_j, j = 0..k-1$. Again, we enforce G_j's schedule with the format $(0^{a_j} h^{\alpha} 1^b 0^{c_j})^*$, where a_j and c_j are group-specific constants and b is a fixed constant for all groups, such that the following conditions hold: (i) $a_0 = 0$, (ii) $a_j + \alpha + b + c_j = \rho$ is a constant and ρ is the period for all groups, and (iii) $a_j + \alpha + b = a_{j+1}, j = 0..k-2$. Conditions (iii) means that each G_{j+1} is obtained from G_j by shifting the latter to the right by $(\alpha + b)$ positions. To find an appropriate value of b, we imagine that all data are originated from SS_n by assuming that all SSs have zero loads except that SS_n has a load $p'_n = \sum_{i=1}^{n} p_i$. Then we plug p'_n into p_n in Eq. (3) to find the best $\hat{b}$.

With this $\hat{b}$, we need to find the latency $L_2(n, p_1, p_2, ..., p_n, \hat{b})$ to deliver all SSs' data on the original path. The transmission is similar to a pipeline delivery, but with some bubbles sometimes. To model the pipeline behavior, we do not take a 'micro-view' on the system. Instead, we take a 'macro-view' to partition the path into $n' = \lceil \frac{n}{k} \rceil$ *trains*, by traversing from the end (i.e., SS_n) toward the head (i.e., SS_1) of the path by grouping, every consecutive k SSs are as one train (when n is not divisible by k, the last few SSs are grouped into one train). We make two observations on these trains.

Observation 1: In each cycle, a train can deliver up to $b \cdot d$ bytes of data to the next train, no matter where these data are located in which SSs of the train.

However, a bubble appears when a train does not have sufficient data to be delivered to the next train. Below, we show when bubbles will not appear.

Observation 2: Except the first $n' = \lceil \frac{n}{k} \rceil$ cycles, the BS will continuously receive $b \cdot d$ bytes of data in every cycle until no more data exists in the path.

Observation 2 implies that if we can derive the network state at the end of the n'th cycle, the latency can be easily derived. To derive the network state after each cycle, let $S_i = (w_1^{(i)}, ..., w_{n'}^{(i)})$ be the network state at the end of the ith cycle, $i = 0..n'$, where $w_j^{(i)}$ is the total load remaining in the jth train at the end of the ith cycle. Initially, $w_j^{(0)}$ is the total loads of those SSs in the jth train. Then we enter a recursive process to find S_{i+1} from S_i, $i = 0..n'-1$ as follows:

$$w_j^{(i+1)} = \begin{cases} \max\{w_j^{(i)} - bd, 0\} & \\ + \min\{w_{j-1}^{(i)}, bd\}, & j = 2..n' \\ \max\{w_j^{(i)} - bd, 0\}, & j = 1. \end{cases} \tag{4}$$

Eq. (4) is derived based on observation 1. In the upper equality, the first term is the remaining load of the jth train after subtracting delivered data and the second term is the amount of data received from the previous train. The lower equality is delivered similarly.

According to observation 2, after the n'th cycle, the BS will see no bubble until all data on the path is empty and it will take $\lceil \frac{\sum_{j=1}^{n'} w_j^{(n')}}{b \cdot d} \rceil$ more cycles to deliver all remaining data. This leads to

$$L_2(n, p_1, p_2, ..., p_n, \hat{b}) = (\lceil \frac{\sum_{j=1}^{n'} w_j^{(n')}}{b \cdot d} \rceil + n') \cdot \rho, \tag{5}$$

where $\rho = k \cdot (\alpha + b)$ is the period of cycles. As has been clear from the context, previous $\hat{b}$ in Eq. (5) is just an estimation.

The optimal b may appear at a point to the left of $\hat{b}$[1]. One may repeatedly decrease $\hat{b}$ to find a better value.

C. A Grid Topology

Here we show how to extend our scheduling schemes to a grid topology. The scheduling is built on top of the previous chain scheduling results. First, we will construct a *comb-like tree* from the grid network. The comb-like tree is further decomposed into horizontal and vertical chains. For example, Fig. 3(a) shows how such tree is formed. One of the chain passing the BS is called the *trunk chain*, and the others are called *branch chains*. Then, we schedule all branch chains to transmit their data to the trunk chain. Branch chains are divided into H groups and we schedule these groups to transmit sequentially. Finally, we schedule SSs in the trunk chain to transmit their data to the BS.

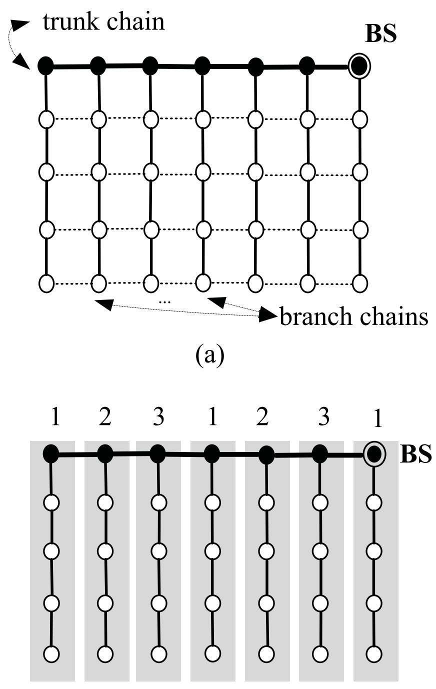

Fig. 3. (a) A comb-like tree on the 5×7 grid topology. (b) The grouping of branch chains when $H = 3$.

Details of the scheme are as follows. We consider a $X \times Y$ grid topology. Without loss of generality, we assume $X \leq Y$ and we decompose the tree into Y vertical chains (branch chains) and one horizontal chain (trunk chain). Intuitively, the trunk chain is larger than the branch chains. There are two phases. In the first phase, branch chains are scheduled to transmit. These chains are divided into H groups. Since two parallel branch chains with a distance of H hops can transmit concurrently without interference, we assign a number between 1 to H to each branch chain in rotation from left to right. Chains marked by the same number are in the same group. Then we schedule each group of chains to transmit sequentially. For example, when $H = 3$, in Fig. 3(b), the seven branch chains are numbered by 1..3 in rotation. Then we let group 1 to transmit until all data are forwarded to the trunk chain, followed by group 2, and then group 3 in a similar way. Since chains in the same group can transmit individually without interference, we can apply the optimal $\hat{b}$ for each chain as formulated above. The latency of phase one is the sum of all groups' latencies. In the second phase, data are already all aggregated at the trunk chain. So, we can apply the easier result again to schedule nodes' transmissions on the trunk chain.

[1]The current $\hat{b}$ is the upper bound of the optimal value because we previously imagine that all data are originated form SS_n.

IV. Performance Evaluation

In this section, we present our simulation results to verify the effectiveness of the proposed schemes. The simulator is written in JAVA language. Unless otherwise stated, the default parameters used in our simulation are $d = 1$ byte, $\alpha = 3$ mini-slots [4], and $H = 3$ hops.

We compare our scheme against two schemes, namely the basic IEEE 802.16d mesh operation [1] and the **BGreedy** scheme [8]. The basic IEEE 802.16d mesh operation assigns the cumulated data plus a transmission overhead as the transmission for each SS without any spatial reuse. **BGreedy** scheme makes each transmission as short as possible to maximize pipeline efficiency. Then, except **BGreedy** scheme, we construct our comb-like tree for all other schemes because they do not discuss the routing tree construction in their works.

In the following results, we use the total latency to compare different schemes. We simulate two scenarios: a chain with multiple requests (SN1) and a grid with multiple requests (SN2). Unless otherwise stated, we use a 15-node chain and a 7×7 grid for the last two scenarios.

A. Impact of Network Size

First, we investigate the effect of network size on the total latency (in mini-slots). Fig. 4(a) and (b) show our results for SN1 and SN2. Each SS has a randomly traffic demand from 0 to 20 bytes. Clearly, the total latencies of all schemes increase as the network size increases. Ours has the best performance. This indicates the necessity of balancing between transmission overhead and pipeline efficiency. This effect is more significant when the network size is larger. In addition, it is to be noted that the schedules generated by our schemes are regular and periodical, which is not so for other schemes.

B. Impact of Transmission Overhead

Next, we investigate the impact of transmission overhead (α) on total latency. Fig. 5 shows the results. The average traffic load of each station is 10 bytes. Naturally, the total latencies of all schemes increase as α increases. In both SN1 and SN2, our schemes significantly outperform the other schemes in all values of α. We see that a larger α will favor our schemes as compared to other schemes because our schemes can reduce the ratio of transmission overhead, especially when the transmission overhead is larger.

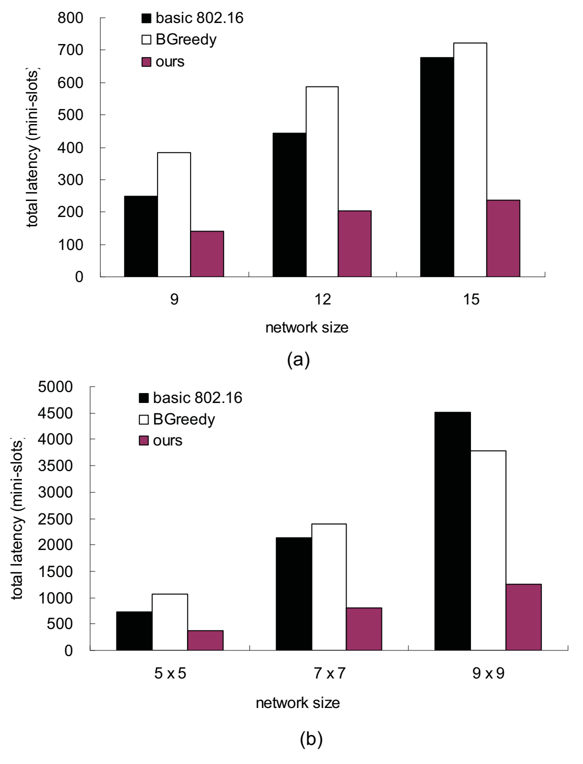

Fig. 4. The impact of network size on total latency in scenarios SN1 and SN2.

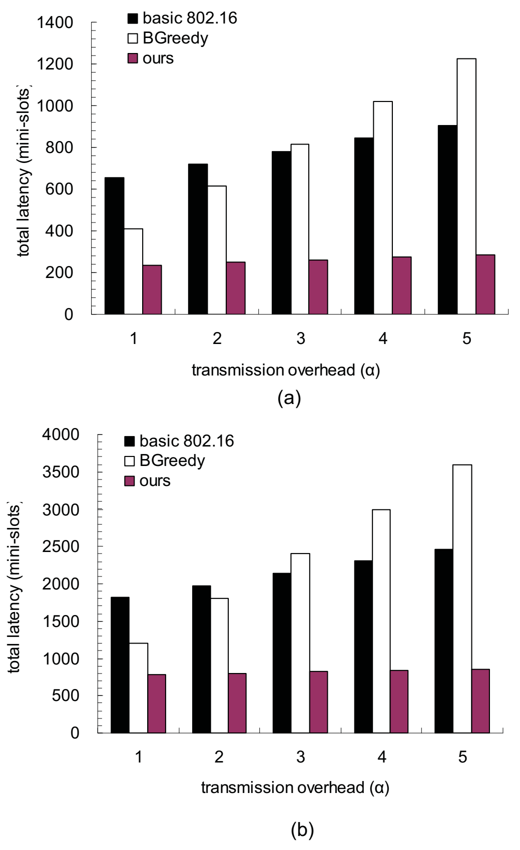

Fig. 5. The impact of transmission overhead (α) on total latency in scenarios SN1 and SN2.

V. Conclusions

This paper addresses the scheduling problem in a grid-based WMN. Most existing solutions try to maximize the pipeline efficiency factor; however, they disregard the cost of transmission overhead. Our approach arranges regular patterns for SSs to transmit repeatedly. Through finding the optimal transmission size for these patterns, the transmission overhead and pipeline efficiency can be balanced such that the total latency can be minimized. With these features, our schemes incur much low computational cost and allow an easy implementation of the scheduler. Simulation results also show that our schemes outperform other schemes, especially when the network size is larger.

Acknowledgements

Y.-C. Tseng's research is co-sponsored by MoE ATU Plan, by NSC grants 96-2218-E-009-004, 97-3114-E-009-001, 97-2221-E-009-142-MY3, and 98-2219-E-009-005, by MOEA 98-EC-17-A-02-S2-0048, and 98-EC-17-A-19-S2-0052, and by ITRI, Taiwan.

References

[1] IEEE Standard 802.16-2004, "IEEE Standard for Local and Metropolitan Area Networks Part 16: Air Interface for Fixed Broadband Wireless Access Systems," 2004.

[2] I. F. Akyildiz, X. Wang, and W. Wang., "Wireless mesh networks: a survey," *Computer Networks*, vol. 47, no. 4, pp. 445–487, 2005.

[3] IEEE Standard 802.11-1999, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," 1999.

[4] P. Djukic and S. Valaee, "Delay aware link scheduling for multi-hop TDMA wireless networks," *IEEE/ACM Transactions on Networking (TON)*, vol. 17, no. 3, pp. 870–883, 2009.

[5] S. Ahson and M. Ilyas, *WiMAX: standards and security*. CRC Press, 2007.

[6] Y. Xiao, *WiMAX/MobileFi: advanced research and technology*. Auerbach Publications, 2008.

[7] A. Taha and H. Hassanein, "IEEE 802.16 mesh schedulers: issues and design challenges," *IEEE network*, vol. 22, no. 1, pp. 58–65, 2008.

[8] F. Jin, A. Arora, J. Hwan, and H. Choi, "Routing and packet scheduling for throughput maximization in IEEE 802.16 mesh networks," in *Proceedings of IEEE Broadnets*, 2007.

[9] J. Zhang, H. Hu, L. Rong, and H. Chen, "Cross-layer Scheduling Algorithms for IEEE 802.16 Based Wireless Mesh Networks," *Wireless Personal Communications*, vol. 51, no. 3, pp. 375–378, 2009.

[10] K. Jain, J. Padhye, V. Padmanabhan, and L. Qiu, "Impact of interference on multi-hop wireless network performance," *Wireless networks*, vol. 11, no. 4, pp. 471–487, 2005.

[11] J. Robinson and E. Knightly, "A performance study of deployment factors in wireless mesh networks," in *the 26th IEEE International Conference on Computer Communications (INFOCOM)*, 2007, pp. 2054–2062.