Computer Communications 33 (2010) 2048–2056

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Mini-slot scheduling for IEEE 802.16d chain and grid mesh networks

Jia-Ming Liang [*], Ho-Cheng Wu, Jen-Jee Chen, Yu-Chee Tseng

Department of Computer Science, National Chiao-Tung University, Hsin-Chu 30010, Taiwan

A R T I C L E I N F O

Article history:
Available online 29 July 2010

Keywords:
IEEE 802.16
Routing tree
Mini-slot scheduling
WiMAX
Wireless mesh network

A B S T R A C T

This work considers the mini-slot scheduling problem in IEEE 802.16d *wireless mesh networks (WMNs)*. An efficient mini-slot scheduling needs to take into account the transmission overhead, the scheduling complexity, and the signaling overhead to broadcast the scheduling results. We are interested in chain and grid WMNs, which are the basic topologies of many applications. We propose scheduling schemes that are featured by low complexity and low signaling overhead. Compared to existing works, this work contributes in developing low-cost schemes to find periodical and regular schedules that achieve near-optimal transmission latencies by balancing between transmission overhead and pipeline efficiency and that are more practical and easier to implement. To minimize the transmission latency, we model the transmission latency as a function of the transmission size and the subscriber stations' traffic demands, and take the first-order derivative of the transmission size to find the minimum latency. Simulation results show that our schemes significantly improve over existing works in computational complexity while maintain similar or better transmission latencies.

1. Introduction

Recently, the IEEE 802.16 standard [1] has been proposed to support wide-range wireless broadband access. The standard is based on a common *medium access control (MAC)* protocol compliant with several physical layer specifications. The protocol supports *point-to-multipoint (PMP)* and *mesh* modes. This work considers the IEEE 802.16d *wireless mesh network (WMN)*, which can be used as a broadband wireless backbone [2,3]. A WMN has a *base station (BS)* connecting to multiple *subscriber stations (SSs)* via multi-hop transmissions. In an IEEE 802.16d WMN, transmission follows a *time division multiple access (TDMA)* mechanism over the underlying *orthogonal frequency division multiplexing (OFDM)* physical layer. 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 [4–6]. 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 [8–15] have studied the scheduling problem in WMNs. Reference [8] proposes an interference-aware, cross-layer scheduling scheme to increase the utilization of a WiMAX mesh network. Reference [9] suggests using concurrent transmissions to improve overall end-to-end throughput of a WMN. Reference [10] shows how to increase spatial reuse to improve system throughput and provide fair access for subscriber stations. Reference [11] presents a flow-control scheduling to provide quality of service guarantee for flows. Reference [12] shows how to maximize spatial reuse to achieve better overall network throughput and higher spectral efficiency. Reference [13] proposes four criteria to select conflict-free links to reduce the scheduling length. These criteria include random selection, min-interference selection, nearest-to-BS selection, and farthest-to-BS selection. Reference [14] considers that each transmission can transmit one piece of data and tries to maximize spatial reuse to minimize the total transmission time. However, all above schemes do not consider the cost of *transmission overhead* (to be defined later on).

[*] Corresponding author. Tel.: +886 3 5712121; fax: +886 3 5721490.
E-mail address: jmliang@cs.nctu.edu.tw (J.-M. Liang).

0140-3664/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2010.07.016

For example, the results in [13,14] are not optimal because they do not take this into account. Considering transmission overheads, [15] proposes to always find the maximal number of concurrent transmission links in each round. This problem has been shown to be NP-hard [16]. Although it performs close to optimum, its computational complexity is too high to be used by the BS. Also, the signaling overhead incurred by [15] 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 is also quite low. To the best of our knowledge, our work is the first one with these properties. Our scheme incurs low complexity and the result is applicable to most regular topologies, such as chain, grid, and triangle networks, which have been proved to have many applications, such as the mesh networks deployed in rural areas in South Africa to provide Internet access [17], the VoIP testbed [18], and the mobility testbed developed in [19]. In addition, these topologies 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) [20,21]. 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 [14]. Simulation results are provided to verify our claims on these topologies.

The rest of this paper is organized as follows. Section 2 formally defines our mini-slot scheduling problem. Section 3 presents our schemes. Simulation results are given in Section 4. Section 5 concludes this paper.

2. Problem definition

We are given one BS and n SSs, SS_i, $i = 1, \ldots, n$. These BS and SSs are deployed in a chain, grid or triangle topologies, as shown in Fig. 1. The BS can be placed at any location in 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.

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 '000hh111100' to indicate that a SS is idle in the first three mini-slots, waits for two overhead mini-slots, transmits for four mini-slots, and then stays idle in the last two 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 $a \geqslant 0$, $b > 0$, and $c \geqslant 0$ are integers and $a + \alpha + b + c = \rho$. The 0s at the end of a schedule are necessary when we consider periodical schedule. 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.

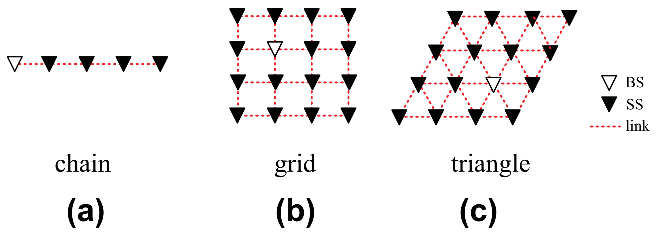

Fig. 1. (a) A 5-node chain topology, (b) a 4 × 4 grid topology, and (c) a 4 × 4 triangle topology.

3. Scheduling 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 and triangle topologies. Given a grid/triangle network, we first construct a fishbone-like tree. The fishbone-like tree is decomposed into individual chains, each of which can be scheduled using the previous chain solutions. Below, we present three solutions for a chain, from simpler to more complicated cases. Then, we combine these solutions for the grid/triangle topologies.

3.1. 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 $SS_n \to SS_{n-1} \to \cdots \to SS_1 \to BS$ 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 $n \geqslant H$ and $k = n$ otherwise (recall that H is the least spatial-reuse distance). Specifically, we define group G_j, $j = 0, \ldots, k-1$, as follows:

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

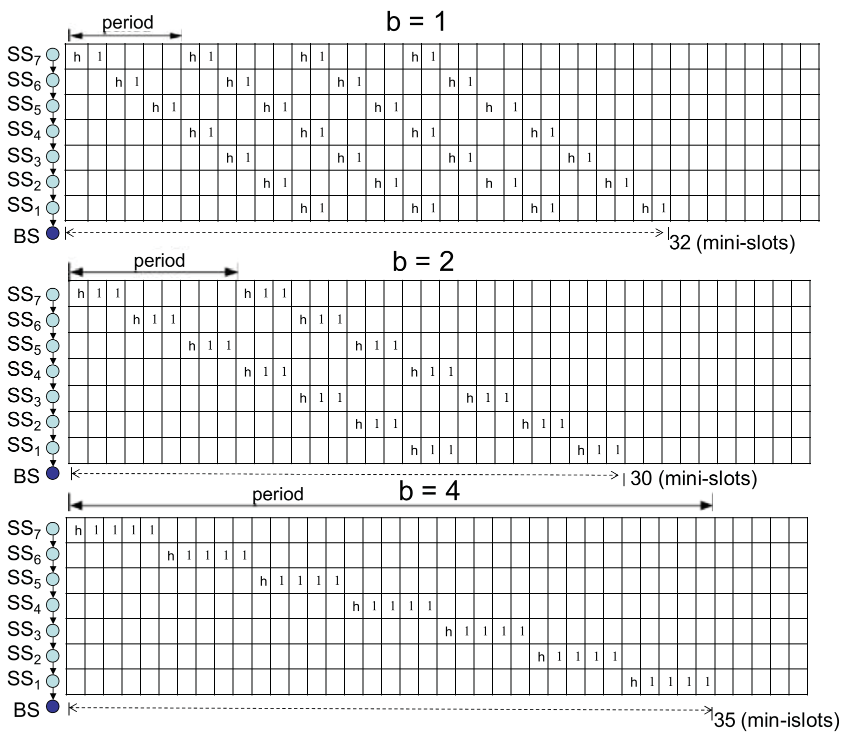

Fig. 2. Transmission schedules for nodes in a chain network (idle state '0' is omitted in the drawing).

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) $a_j + \alpha + b = a_{j+1}$, $j = 0, \ldots, k-2$. 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:

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

When $n \geqslant H$, each cycle has a length of $H \cdot (\alpha + b)$ mini-slots. It takes $\left\lceil \frac{p_n}{b\cdot d} \right\rceil$ cycles for SS_n to transmit its last piece of data. At the end of the $\left\lceil \frac{p_n}{b\cdot d} \right\rceil$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

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

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 $\hat{L_1}$.

3.2. A chain with multiple requests

Next, we consider a path $SS_n \to SS_{n-1} \to \cdots \to SS_1 \to BS$ with multiple non-zero-load nodes. Without loss of generality, we assume SS_n's load is non-zero. Similar to Section 3.1, we divide SSs into k groups G_j, $j = 1,\ldots,k-1$. 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) $a_j + \alpha + b = a_{j+1}$, $j = 0,\ldots,k-2$. 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 $L_2(n, p_1, p_2, \ldots, p_n, \hat{b})$ 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 $n' = \left\lceil \frac{n}{k} \right\rceil$ *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 $n' = \left\lceil \frac{n}{k} \right\rceil$ cycles, the BS will continuously receive $b \cdot d$ bytes of data in every cycle until no more data exists in the path.

Proof. Consider the first cycle, the data delivered by n'th train is only its data. However, in the second cycle, the data delivered by the n'th train will be the n'th chain's data or both the n'th and $(n'-1)$ th chains' data. So, in the n'th cycle, the data delivered by n'th train will be the n'th chain's data $\sim$1st chain's data. If the n'th chain's has no sufficient data (i.e., $<b \cdot d$), the delivery will combine the previous train's data, i.e., $(n'-1)$th chain's data or even $(n'-2)$th chain's data, etc. So, if the n'th train deliver less than $b \cdot d$ (so called bubble) to BS after n'th cycle, it means that the amount of data on all trains must be less than $b \cdot d$ after n'th cycle. Or it must deliver $b \cdot d$ data without bubbles. It's proved. □

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 $S_i = (w_1^{(i)}, \ldots, w_{n'}^{(i)})$ be the network state at the end of the ith cycle, $i = 0, \ldots, n'$, 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, $i = 0, \ldots, n'-1$ as follows:

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

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 $\left\lceil \frac{\sum_{j=1}^{n'} w_j^{(n')}}{b \cdot d} \right\rceil$ more cycles to deliver all remaining data. This leads to

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

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.

3.3. A chain with multiple requests and BS in the middle

Since the BS in the middle, we model the chain as a path $SS_\ell \rightarrow SS_{\ell-1} \rightarrow \ldots \rightarrow SS_1 \rightarrow BS \leftarrow SS'_1 \leftarrow SS'_2 \leftarrow \cdots \leftarrow SS'_r$. Without loss of generality, we assume $\ell \geqslant r$ and both SS_ℓ and SS'_r have non-zero loads. For simplicity, we call $SS_\ell \rightarrow SS_{\ell-1} \rightarrow \cdots \rightarrow SS_1 \rightarrow BS$ the left chain C_L, and $BS \leftarrow SS'_1 \leftarrow SS'_2 \leftarrow \cdots \leftarrow SS'_r$ the right chain C_R. The arrangement of concurrent transmission-able groups is more difficult because we intend to transmit sufficient data to the BS from both chains without congestion in each cycle. First, we need to identify a new value for k (the number of groups):

$$k = \begin{cases} H, & \text{if } 1 \leqslant H \leqslant 4, \\ 2H-4, & \text{if } H \geqslant 5. \end{cases} \tag{6}$$

Eq. (6) means that when $1 \leqslant H \leqslant 4$, we can still manage to have the most compact number of concurrent transmission-able groups. However, when $H \geqslant 5$, the number of groups will exceed H, which is not most compact. Fortunately, in practice, $H \leqslant 4$ in most cases.

Now, for $j = 0, \ldots, k-1$, we define group G_j as follows:

$$G_j = \{SS_i | (\ell - i) \bmod k = j, i = 1, \ldots, \ell\} \bigcup \{SS'_i | (\ell - i + \Delta) \bmod k = j, i = 1, \ldots, r\} \tag{7}$$

In Eq. (7), nodes in C_L and C_R are grouped sequentially similar to the earlier cases. However, for C_R, the grouping of nodes is shifted by a value of Δ to allow concurrent transmissions, where $\Delta = 1$ if $H = 2$ and $\Delta = H - 2$ if $H \geqslant 3$. Fig. 3 shows some examples with $\ell = 6$ and $r = 6$. In the example of $H = 2$, we shift the grouping of nodes in C_R by a value of $\Delta = 1$. In the examples of $H = 3$ and $H = 4$, we shift the grouping of nodes in C_R by a value of $\Delta = H - 2$. Such shifting avoids nodes nearby the BS from colliding with each other. When $H \geqslant 5$, the value of k is defined differently. However, shifting by $\Delta = H - 2$ still helps avoid collision. Note that when $H = 5$ and 6, there are 6 and 8 groups, respectively, where these numbers of groups are least for grouping on both chains to transmit without congestion. We will explain later on.

When $H \leqslant 4$, the collision-free property of the above grouping can be proved by case-by-case validation. When $H \geqslant 5$, Fig. 4 gives a general proof. There are $k = 2H - 4$ groups. Consider nodes SS_i and SS'_i, $i = 1, \ldots, 2H-4$ on C_L and C_R. Assume that the former i $(1 \leqslant i \leqslant 2H-4)$ SSs are grouped from G_0 and the remaining $2H-4-i$ SSs are grouped from G_i. We prove it in two aspects. First, since the indexes of groups on both two chains are increasing toward BS, if the pair of SS_x and SS_y, where they are in the same group but on different chains, have no interference to each other that will make all SS_i, $i \geqslant x$ and SS'_i, $i \geqslant y$ be interference free. By Eq. (7), all SSs on C_R are shifted by $H - 2$ of grouping sequence, the critical pairs (i.e., SS_1 and SS'_{H-1}) can be in same groups while keep a distance of H hops. That means those SS_i, $i \geqslant 1$ and SS'_i, $i \geqslant (H-1)$ can be grouped successfully and will not cause any interference when those SSs are in their groups. On the other hand, as we know that SS'_1 is grouped by G_{i+H-3}. Since SS_{2H-4} is grouped in G_i, the smallest index of SS grouped by G_{i+H-3} will be SS_{H-1}, which has a distance of H hops to SS'_1. By these two aspects, all SS_i, $i \geqslant (H-1)$ and SS'_i, $i \geqslant 1$ will not cause any interference when those SSs are in their groups. Then, we can promise that each SS_i and each SS'_i, $i \geqslant 1$ can be grouped by Eqs. (6) and (7) to transmit without any interference and congestion.

Theorem 1. *By Eqs. (6) and (7), SSs in the same group can transmit concurrently without collision for any value of H.*

With Theorem 1, we can allow C_L and C_R to transmit concurrently without collision. So the results in Section 3.2 can be applied here. (The only exception is that the first transmitting node in C_R, i.e., SS'_r, may not start from group G_0. In this case, we can add some virtual nodes to C_R and make one start from G_0.) For C_L, there will exist an optimal value of $\hat{b}_\ell$ such that $L_2(\ell, p_1, p_2, \ldots, p_\ell, \hat{b}_\ell)$ is smallest. Similarly, for C_R, there will exist an optimal value of $\hat{b}_r$ such that $L_2(r, p'_1, p'_2, \ldots, p'_r, \hat{b}_r)$ is smallest. Plugging in any possible b, we can formulate the latency as follow:

$$L_3(\ell, p_1, p_2, \ldots, p_\ell, r, p'_1, p'_2, \ldots, p'_r, b) = \max\{L_2(\ell, p_1, p_2, \ldots, p_\ell, b), L_2(r, p'_1, p'_2, \ldots, p'_r, b)\} \tag{8}$$

The best value of b, called $\hat{b}$, may appear nearby or between $\hat{b}_\ell$ and $\hat{b}_r$.

3.4. A general grid/triangle topology

Here we show how to extend our scheduling scheme to a grid/triangle topology. The scheduling is built on top of the previous chain scheduling results. First, we will construct a *fishbone-like tree* from the grid/triangle network. The fishbone-like tree is further decomposed into horizontal and vertical chains. For example,

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

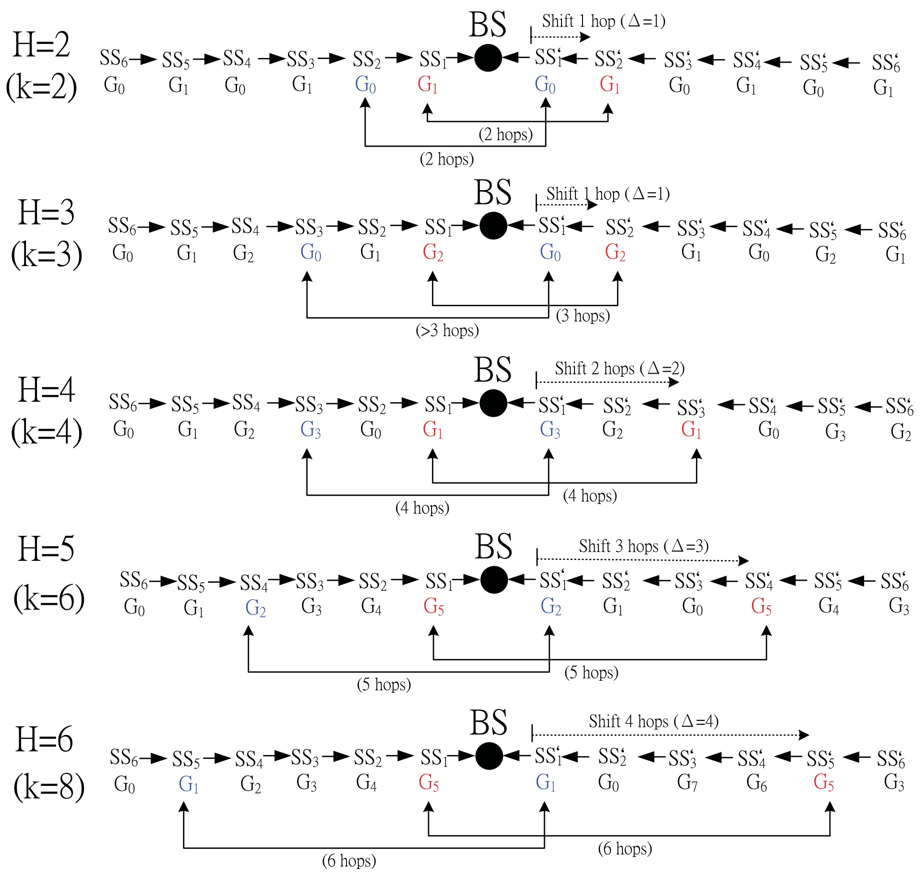

Fig. 3. The arrangement of transmission-able groups for $H = 2, 3, 4, 5$, and 6 when BS is in the middle.

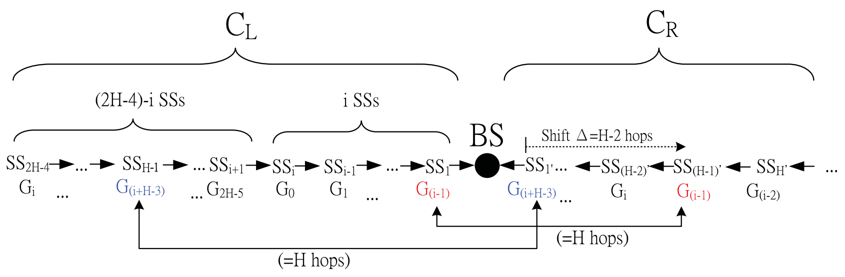

Fig. 4. General collision-free proof for the cases of $H \geqslant 5$.

Fig. 5(a) shows how such trees are 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.

Details of the scheme are as follows. We consider a $X \times Y$ grid/triangle topology. Without loss of generality, we assume $X \leqslant Y$ 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. 5(b), the seven branch chains are numbered by $1, \ldots, 3$ 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.

4. Performance evaluation

In this section, we present our simulation results to verify the effectiveness of the proposed scheme. The simulator is written in JAVA language. Unless otherwise stated, the default parameters

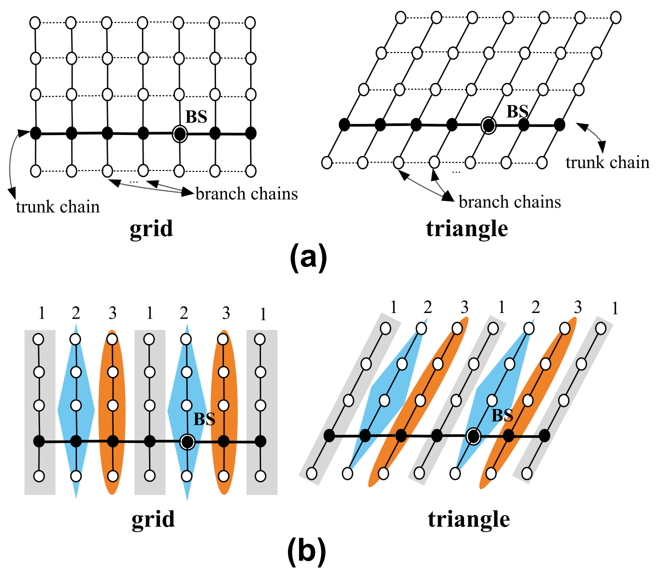

Fig. 5. (a) A fishbone-like tree on the 5 × 7 grid/triangle topology. (b) The grouping of branch chains when $H = 3$.

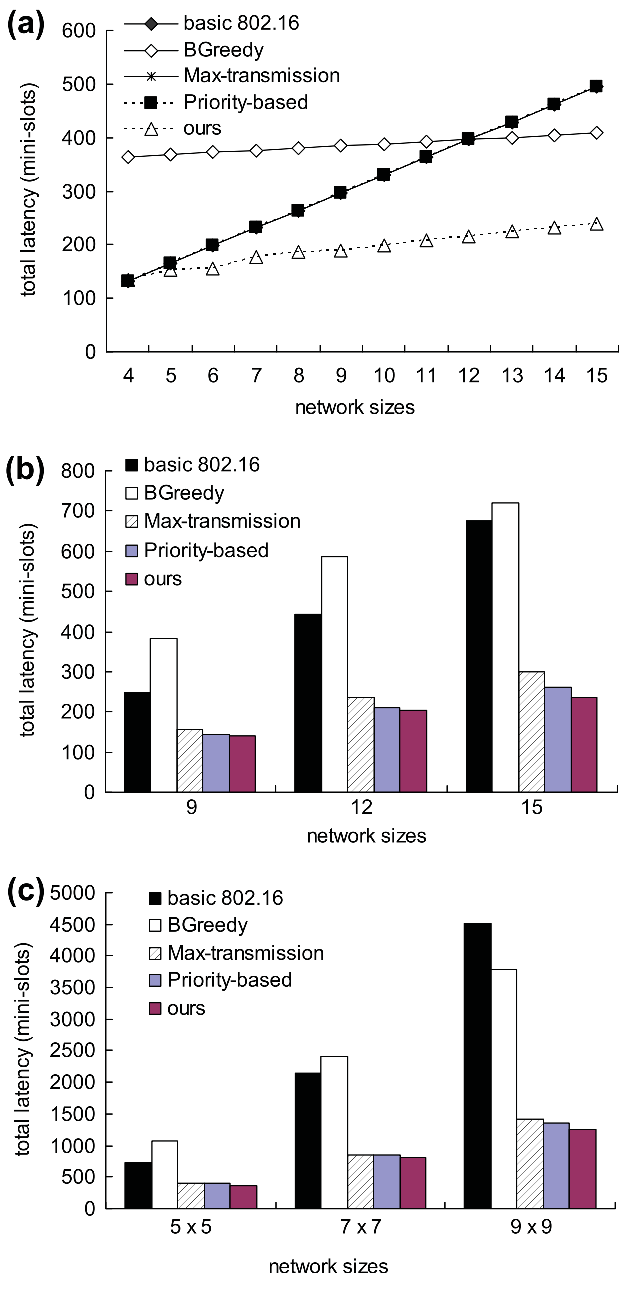

Fig. 6. The impact of network size on total latency in scenarios SN1, SN2, and SN3.

used in our simulation are $d = 1$ byte, $\alpha = 3$ mini-slots [4,5], and $H = 3$ hops.

We compare our scheme against four schemes, named the basic IEEE 802.16d mesh operation [1], the **BGreedy** scheme [14], the **Max-transmission** scheme [15], and the **Priority-based** scheme [15]. The reason for selecting the **BGreedy** scheme for comparison is that it considers the pipeline efficiency, while that for selecting the **Max-transmission** scheme and the **Priority-based** scheme for comparison is that they consider the transmission overhead. 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. **Max-transmission** scheme always finds the maximal number of concurrent transmission links in the network round by round and assigns those links to transmit the minimal buffered data plus a transmission overhead. **Priority-based** scheme first finds all available links sets, which can transmit without interference, and chooses one with the maximal predefined priority. Then, it assigns those links to transmit the minimal buffered data plus a transmission overhead. Here, we adopt **LQR** priority, which performs the best performance in [15]. Such priority consults some network information, such as the hop counts, queue lengths, and transmission rates of the links. Then, except **BGreedy** scheme, we construct our fishbone-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 three scenarios: a chain with a single request (SN1), a chain with multiple requests (SN2), and a grid with multiple requests (SN3). Unless otherwise stated, we use a 15-node chain and a 7 × 7 grid for the last two scenarios with BS in the middle. We remark that since the results of the triangle topology are almost the same as those in grid topology, we will only discuss the grid case.

4.1. Impact of network size

First, we investigate the effect of network size on the total latency (in mini-slots). Fig. 6(a) considers scenario SN1 with $p_n = 30$ bytes by varying n. Clearly, the total latencies of all schemes increase as n increases. **BGreedy**, **Max-transmission**, and **Priority-based** schemes perform the same because when the traffic demand is from only one SS, they schedule one transmission each time without any spatial reuse. Although **BGreedy** tries to maximize spatial reuse, its latency is still higher than ours because it disregards the transmission overhead. Ours has the best performance. This indicates the necessity of balancing between transmission overhead and pipeline efficiency. This effect is more significant when n is larger. Fig. 6(b) and (c) shows our results for SN2 and SN3, respectively. Each SS has a randomly traffic demand from 0 to 20 bytes. Similar to SN1, the total latencies of all schemes increase as the network size increases. Although all schemes will exploit spatial reuse when there are multiple traffic demands, our scheme still outperforms all the other schemes. In addition, it is to be noted that the schedules generated by our scheme are regular and periodical, which is not so for other schemes.

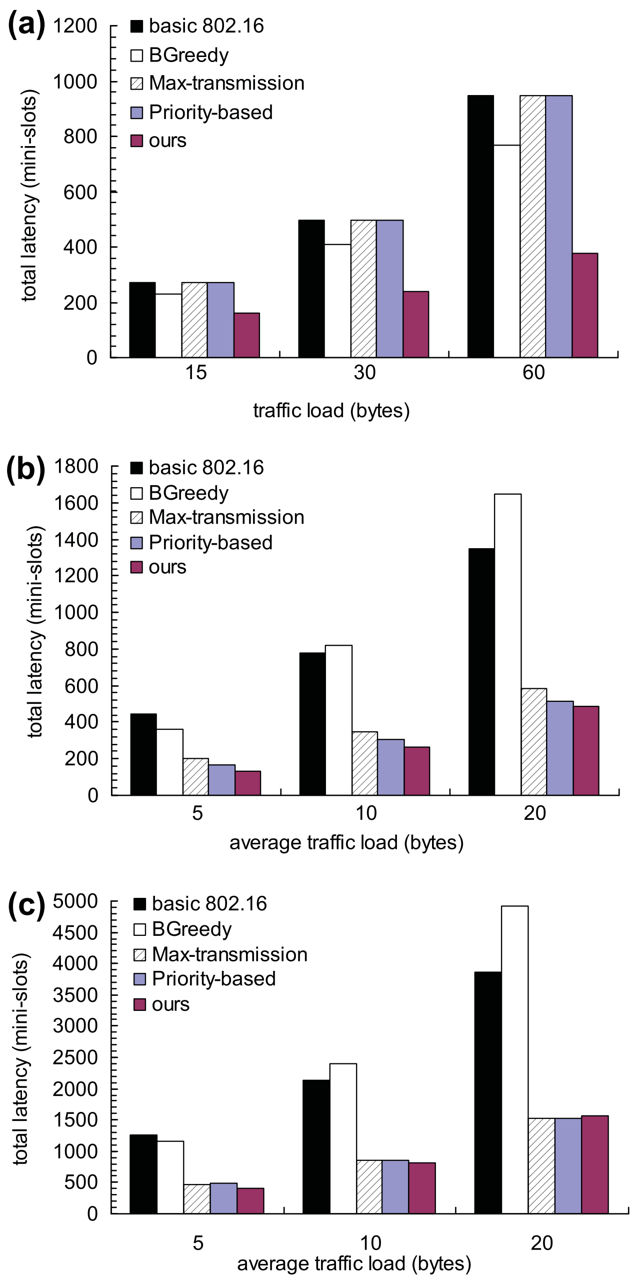

Fig. 7. The impact of traffic load on total latency in scenarios SN1, SN2, and SN3.

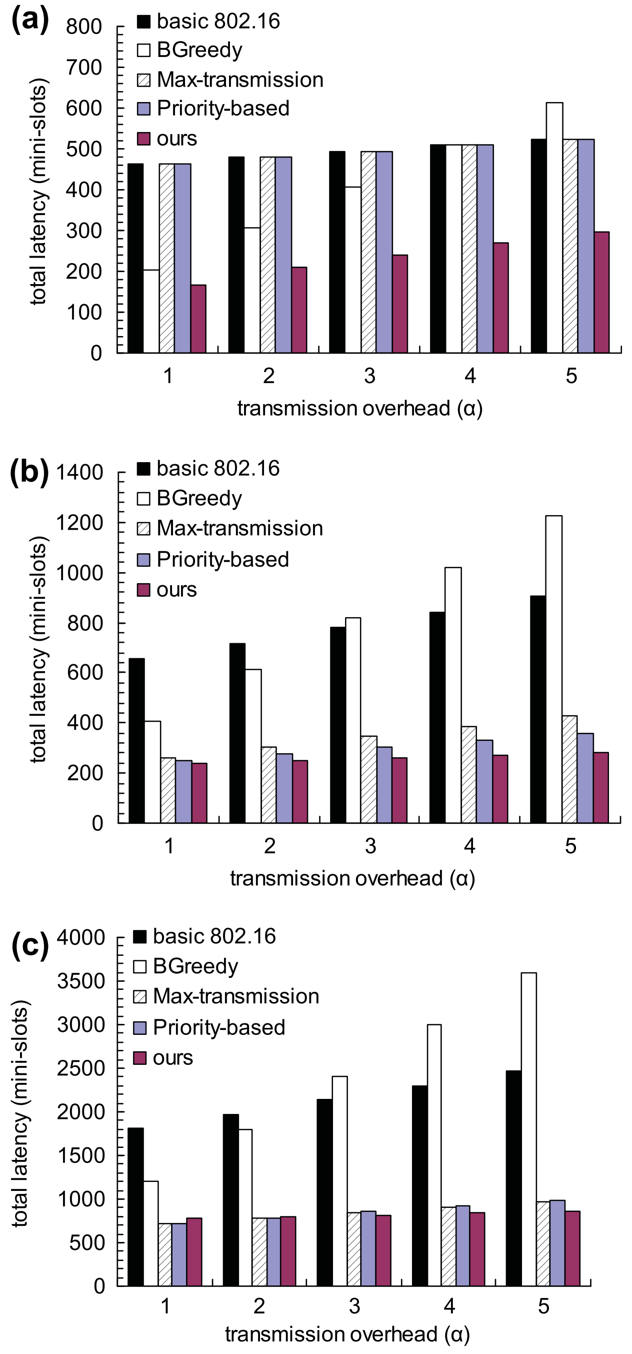

Fig. 8. The impact of transmission overhead (α) on total latency in scenarios SN1, SN2, and SN3.

4.2. Impact of traffic load

Next, we investigate the effect of average traffic load on total latency. Fig. 7(a) shows the results for SN1 when $n = 15$. Fig. 7(b) and (c) shows the results for SN2 and SN3, respectively, where each SS has a random traffic of 0 to 10, 0 to 20, and 0 to 40 bytes (thus the averages are 5, 10, and 20 bytes, respectively). The trends are similar to the previous cases; our scheme outperforms the other schemes significantly in SN1 and slightly in SN2 and SN3.

4.3. Impact of transmission overhead

Next, we investigate the impact of transmission overhead (α) on total latency. Fig. 8 shows our results. The simulation environment is similar to the previous cases except that we vary the value of α. Naturally, the total latencies of all schemes increase as α increases. In SN1, our significantly outperforms the other schemes in all values of α. In SN2 and SN3, the average traffic load of each station is 10 bytes. We see that a larger α will favor our scheme as compared to **Max-transmission** and **Priority-based** schemes. This is because our scheme enforces a regular schedule for each SS, thus losing some degree of pipeline efficiency. The impact is higher when $\alpha = 1$ and 2. When $\alpha \geqslant 3$, the transmission overhead is too high to be neglected. Thus, balancing between transmission overhead and pipeline efficiency becomes quite important. In practice, α is greater than 2 [5].

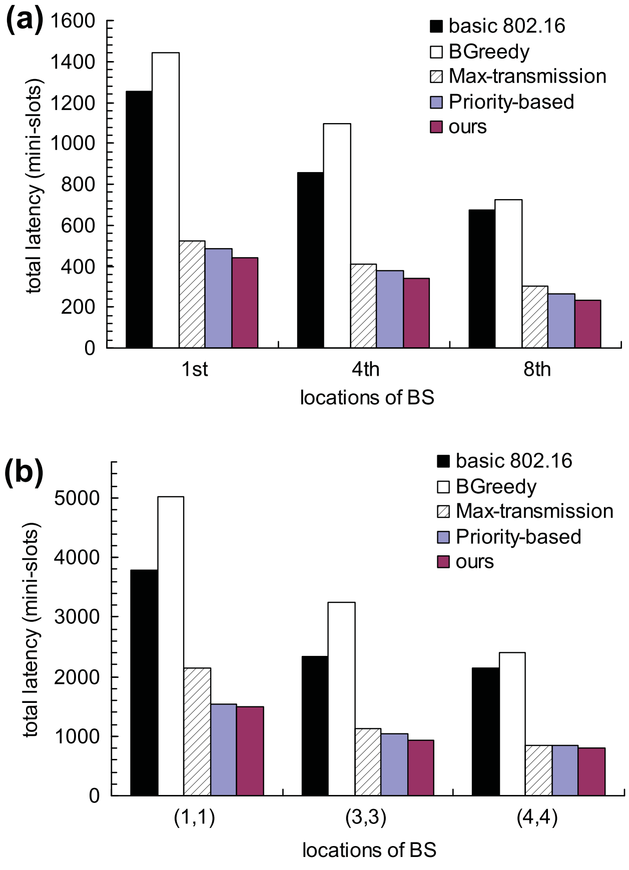

Fig. 9. The impact of locations of BS on total latency in scenarios SN2 and SN3.

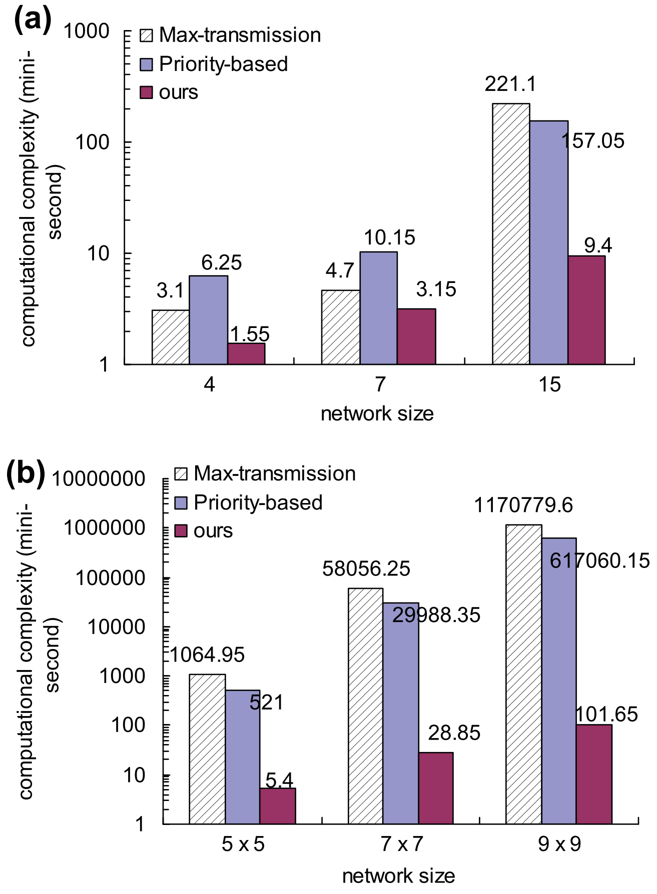

Fig. 10. The impact of network size on computational complexity in scenarios SN2 and SN3.

4.4. Impact of BS location

Next, we move the location of the BS around, as shown in Fig. 9. In SN2, we place the BS in the first, 4th, and 8th position of the chain ($n = 15$). In SN3, we place the BS at (1,1), (3,3), and (4,4) of the 7×7 grid network. In both cases, the total latencies of all schemes reduce as the BS is moved toward the center of the network because SSs are closer to the BS.

4.5. Computational complexity

Finally, we investigate the computational complexities of different schemes. We mainly compare our scheme against **Max-transmission** and **Priority-based** schemes. Note that it has been proved in [16] that the problem that **Max-transmission** and **Priority-based** schemes intend to solve is NP-hard. So we are interested in seeing the total CPU time incurred by these schemes as compared to ours. (The computation time is measured by the platform of IBM R61 with Intel Core 2 Duo T7300 2.0 GHz and DDR2-800 SDRAM 2 GB). From Fig. 10, we see that the computational complexities of all schemes increase as the network size increases. Since both **Max-transmission** and **Priority-based** schemes try to find out the maximal concurrent transmission set of SSs round by round until all data are delivered to BS, the processing time increases exponentially as n grows (note that the y-axis is drawn with exponential scales). For example, the computation costs of **Priority-based** are 4.03, 3.22, and 16.7 times of ours when $n = 4$, 7, and 15, respectively, in SN2 and 96, 1039, and 6070 times of ours in the 5×5, 7×7, and 9×9 grid topologies, respectively. Because our scheme simplifies the grouping of SSs in both chain and grid topologies, and schedules transmissions in quite a regular way, it achieves a much lower cost. This further verifies that our scheme is more practical and is easier to implement, even when the network scales up.

5. Conclusions

This paper addresses the scheduling problem in chain- and grid-based WMNs. While most existing solutions try to address this NP-hard scheduling problem by searching for the sequence of concurrent transmission-able sets to maximize the spatial reuse factor, our approach tries to identify regular patterns that SSs can follow and repeatedly transmit easily. One special feature of our scheme is that it tries to balance transmission overhead and pipeline efficiency. In particular, our scheme tries to fill up the pipeline as full as possible to improve the pipeline efficiency. With these designs, our scheme does achieve better or equal total latency as compared to existing schemes, incurs much low computational cost as compared to existing schemes, and allows an easy implementation of the scheduler.

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, W. Wang, Wireless mesh networks: a survey, Computer Networks 47 (4) (2005) 445–487.
[3] D. Niyato, E. Hossain, J. Diamond, IEEE 802.16/WiMAX-based broadband wireless access and its application for telemedicine/e-health services, IEEE Wireless Communications 14 (1) (2007) 72.

[4] P. Djukic, S. Valaee, Delay aware link scheduling for multi-hop TDMA wireless networks, IEEE/ACM Transactions on Networking (TON) 17 (3) (2009) 870–883.
[5] S. Ahson, M. Ilyas, WiMAX: Standards and Security, CRC Press, 2007.
[6] Y. Xiao, WiMAX/MobileFi: Advanced Research and Technology, Auerbach Publications, 2008.
[7] A. Taha, H. Hassanein, IEEE 802.16 mesh schedulers: issues and design challenges, IEEE Network 22 (1) (2008) 58–65.
[8] H.-Y. Wei, S. Ganguly, R. Izmailov, Z. Haas, Interference-aware IEEE 802.16 WiMax mesh networks, Vehicular Technology Conference (2005) 3102–3106.
[9] J. Tao, F. Liu, Z. Zeng, Z. Lin, Throughput enhancement in WiMax mesh networks using concurrent transmission, International Conference on Wireless Communications, Networking and Mobile Computing (2005) 871–874.
[10] M. Cao, V. Raghunathan, P. Kumar, A tractable algorithm for fair and efficient uplink scheduling of multi-hop WiMAX mesh networks, IEEE Workshop on Wireless Mesh Networks (2006) 93–100.
[11] H. Shetiya, V. Sharma, Algorithms for routing and centralized scheduling in IEEE 802.16 mesh networks, Wireless Communications and Networking Conference (2006) 147–152.
[12] L. Fu, Z. Cao, P. Fan, Spatial reuse in IEEE 802.16 based wireless mesh networks, IEEE International Symposium on Communications and Information Technology (2005) 1358–1361.
[13] B. Han, W. Jia, L. Lin, Performance evaluation of scheduling in IEEE 802.16 based wireless mesh networks, Computer Communications 30 (4) (2007) 782–792.
[14] F. Jin, A. Arora, J. Hwan, H. Choi, Routing and packet scheduling for throughput maximization in IEEE 802.16 mesh networks, in: Proceedings of IEEE Broadnets, 2007.
[15] J. Zhang, H. Hu, L. Rong, H. Chen, Cross-layer scheduling algorithms for IEEE 802.16 based wireless mesh networks, Wireless Personal Communications 51 (3) (2009) 375–378.
[16] K. Jain, J. Padhye, V. Padmanabhan, L. Qiu, Impact of interference on multi-hop wireless network performance, Wireless Networks 11 (4) (2005) 471–487.
[17] D. Johnson, Evaluation of a single radio rural mesh network in South Africa, International Conference on Information and Communication Technologies and Development (2007) 1–9.
[18] N. Bayer, B. Xu, V. Rakocevic, J. Habermann, Application-aware scheduling for VoIP in wireless mesh networks, Computer Networks 54 (2) (2010) 257–277.
[19] D. Johnson, G. Hancke, Comparison of two routing metrics in OLSR on a grid based mesh network, Ad Hoc Networks 7 (2) (2009) 374–387.
[20] J. Robinson, E. Knightly, A performance study of deployment factors in wireless mesh networks, International Conference on Computer Communications (INFOCOM) (2007) 2054–2062.
[21] K. Ramachandran, I. Sheriff, E. Belding, K. Almeroth, Routing stability in static wireless mesh networks, Lecture Notes in Computer Science 44 (2007) 73–82.