A Distributed Maximal Link Scheduler for Multi Tx/Rx Wireless Mesh Networks

He Wang, Kwan-Wu Chin, Raad Raad
School of Electrical, Computer and Telecommunications Engineering
University of Wollongong, NSW, Australia
Email: hw407@uowmail.edu.au, {kwanwu, raad}@uow.edu.au

Sieteng Soh
Department of Computing
Curtin University of Technology, WA, Australia
Email: s.soh@curtin.edu.au

***Abstract*—Recently, researchers have developed Wireless Mesh Networks (WMNs) where each router is capable of performing multiple transmissions or receptions concurrently; aka Multi Tx-Rx (MTR) WMNs. Consequently, each node is able to transmit (Tx) or receive (Rx) to/from its neighbors simultaneously. A fundamental problem in such WMNs is to derive a transmission schedule with minimal superframe length to maximize network capacity and minimize end-to-end delays. Unfortunately, deriving a minimal superframe length is equivalent to solving the NP-complete, MAX-CUT problem. To this end, there are a number of centralized schedulers, but only but only one distributed scheduler, called JazzyMAC. Henceforth, in this paper, we add to the state-of-the-art by proposing Algo-d, a novel distributed scheduler that solves the MAX-CUT problem using only local information. Experiment results show Algo-d generates superframes that are 37.5% shorter and it activates 264% more links as compared to JazzyMAC. Lastly, as compared to centralized schedulers, Algo-d schedules 50% more links than Algo-1 and at most 7% fewer links than Algo-2.**

I. Introduction

Wireless Mesh Networks (WMNs) have developed rapidly in recent years to provide 'last miles' connectivity in both urban and rural areas [10]. In this paper, we consider Multi-Transmit-Receive (MTR) WMNs. The nodes in these WMNs are able to transmit to or receive from multiple neighbors concurrently. Nodes with such capability can be found in [13] and [16]. For example, in [13], each node is equipped with multiple radios, each of which is connected to a directional antenna, which allows nodes to direct their transmission to one of their neighbors. A key constraint is that nodes, due to side-lobes, are not allowed to transmit *and* receive concurrently, which we define as *no Tx-Rx* constraint; see Figure 1.

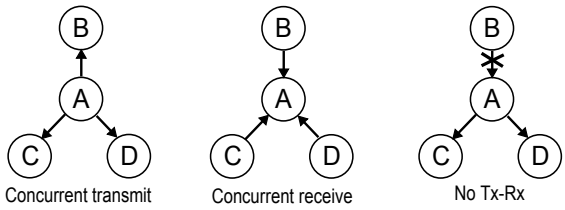
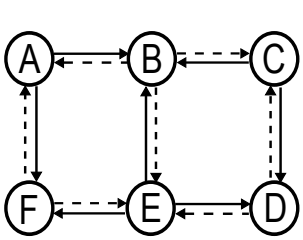


Fig. 1: An example of how MTR works

A fundamental problem in MTR WMNs is link scheduling. The aim is to derive the shortest possible superframe or use the minimal number of slots that allows each link to be activated at least once. Figure 2 shows an example WMN with six nodes and the corresponding link schedule. Note that the resulting schedule follows the *no Tx-Rx* constraint. The challenge is deriving a link schedule that adheres to the said constraint and also maximizes the number of links in each time slot. Under such a link schedule, the network capacity is maximal, which also helps reduce end-to-end delay.



Superframe	
slot-1	slot-2
B→A	A→B
B→C	C→B
B→E	E→B
D→C	C→D
D→E	E→D
F→E	E→F
F→A	A→F

Fig. 2: An example WMN and its corresponding link schedule

To date, several centralized schedulers for MTR WMNs have been proposed [4], [7], [6], [5], [13], [8], [11], [12]; see Section II for more details. However, to the best of our knowledge, there is only one directly relevant work, JazzyMAC [9]. Further, a node using JazzyMAC can only transmit when it has the token of all its links, and thus in each slot, the number of activated links is not maximized. Henceforth, our paper makes the following contributions. We present a distributed link scheduling algorithm called Algo-d that has the following key features: (i) it minimizes the number of idle links in each time slot; (ii) it activates each link at least once and minimizes the resulting superframe length. Our experiment results show that Algo-d shortens the superframe length by 37.5% and increases the number of activated links in each time slot by 264% as compared to JazzyMAC.

This paper has the following structure. Section II reviews prior works. Section III presents our network model. A description of the problem is shown in Section IV. Our solutions are outlined in Section V and experiment results are shown in Section VI. Our conclusions are presented in Section VII.

II. Related Work

To date, there are many link scheduling algorithms or Medium Access Control (MAC) protocols for WMNs. We will only review those developed for MTR WMNs. Readers interested in other algorithms/protocols are referred to [3].

In [4], the authors propose a cluster-based link scheduling algorithm to maximize network capacity. Their algorithm aims to construct a maximum parallel transmission set. Hung et al.'s work [7] extends that of [4] to consider delay between nodes. However, both [4] and [7] do not consider communication between clusters and more importantly, do not outline how clusters are formed nor studied the impact of different cluster policies on capacity. Raman et al. [13] propose a link scheduling algorithm called 2P to maximize network capacity. The 2P MAC switches nodes between two phases: SynRx and SynTx. If a node is transmitting on all links (SynTx), all of its neighbours must be in reception mode (SynRx), thus the topology must be bipartite. Chin et al. [6] [5] relaxed this assumption by proposing Algo-1, a solution that derives a schedule for arbitrary topologies. At every other slot, Algo-1 recursively divides the topology into two disjoint, maximally connected sets. Nodes that transmit in a time slot switch to receiving in the subsequent time slot. In addition, Algo-1 adds opportunistic links to the derived sets; these are links that have been activated in prior slots. However, the throughput achieved by Algo-1 and 2P has been shown to be sub-optimal [8]. To this end, Loo et al. [8] propose a link scheduling algorithm called Algo-2, which recursively divides the topology into two disjoint, maximally connected sets in every slot. To generate these two sets, all nodes are initially in one set. Algo-2 moves a node to the other set if it creates more links between the two sets. Nodes in one set transmit and nodes in the other set receive. It then removes activated links and generates a new MAX-CUT for the next time slot. All these works are centralized solutions, meaning they are impractical in large scale WMNs changes frequently. Thus we seek a distributed solution.

In terms of distributed schedulers, Bao et al. [2] propose a distributed link scheduling protocol called Receiver-Oriented Multiple Access (ROMA) for smart antenna systems. ROMA adopts the neighbour-aware contention resolution algorithm (NCR) proposed in [1] to derive a random channel access schedule for each node. With the use of a hash function, current time, link weight and node ID, ROMA determines the state of nodes and transmitting links in each slot. However, the aim of ROMA is different from ours as we seek to derive a minimal superframe in a distributed manner. Furthermore, as links are scheduled pseudo-randomly, the resulting network capacity is unlikely to be optimal. Rhee et al. [14] present a distributed randomized time slot scheduling algorithm, called DRAND, to maximize capacity. Each node sets itself a probability, which is dependent on the number of two hops neighbors that have yet to receive a slot, to broadcast a transmission request to its neighbors at the beginning of each frame. A node that receives a transmission request from a neighbor sends back a grant message containing its free slots. Upon receiving the grant message, a node compares its own free slots with those specified in the grant message. It then broadcasts a release message containing its busy slots and intended receivers to its neighbors. However, DRAND does not guarantee the maximum number of links is activated in each slot nor ensures every link is activated at least once.

The only directly relevant work to us is JazzyMAC [9]; that is, it is a MAC developed specifically for MTR WMNs. Indeed, JazzyMAC is a distributed version of 2P [13] and addresses the following limitations of 2P: (i) the network topology must be bipartite, (ii) the use of fixed length transmission slots, meaning 2P cannot adapt to dynamic traffic loads. In JazzyMAC, each link is associated with a token. Only the node holding a token can transmit on the associated link. However, JazzyMAC has a number of limitations. In particular, while a node is waiting for tokens, some of its links are idle. Hence, in each slot, the network capacity is less than optimal. As we will see in Section V, Algo-d creates a maximal MAX-CUT in every slot, similar to [8] but in a distributed manner. Different from JazzyMAC, Algo-d maximizes the number of transmitting nodes in each slot, and thus maximizes the number of activated links.

III. Network Model

Consider a MTR WMN modelled as a directed graph $G(V, E)$, where V denotes the set of vertices/nodes/routers and E denotes the set of directional links. Let $v_i \in V$ denote a node i with k_i radios, and $e_{ij} \in E$ represents a directional link from node i to j. We will denote N_i to be the set of node i's neighbours. Assume each TDMA superframe S contains $|S|$ time slots and each slot is sufficient for data packet and ACK. Each slot t contains a $|E|$ dimensional activation vector $\mathbf{e^t}$, where each element corresponds to a link (i,j) and is set to one if said link is active in slot t. That is, the vector $\mathbf{e^t}$ denotes the set of links that adhere to the no-Tx-Rx constraint. Note, we define a transmission set $\mathbf{e^t}$ to be $maximal$ if no other links can be added into it without violating the said constraint. Let $\mathcal{B}$ be the set containing all maximal feasible transmission sets. Also define λ_t to be a binary variable, i.e., $\lambda_t \in \{0, 1\}$, that indicates whether transmission set $\mathbf{e^t}$ is included in the superframe S. Also $\mathbf{1}$ is a $|E|$ dimensional vector containing all 1s.

Notation	Definition
V	The set of vertices/nodes/routers
v_i	Node i
E	The set of directional links
e_{ij}	Directional link from v_i to v_j
$\mathbf{e^t}$	Set of links that can transmit concurrently in slot t
S	TDMA superframe
(x_i, y_i)	The variable x_i denotes the total number of nodes in $S1$ that v_i points to, and y_i denotes the total number of nodes in $S2$ that v_i connects to.
Δ_i	The difference between x_i and y_i, i.e., x_i - y_i
$\mathcal{B}$	The set containing all maximal feasible transmission sets
λ_t	A binary variable that indicates whether transmission set $\mathbf{e^t}$ is included in the superframe S

TABLE I: Notations and definitions

Let S_1 and S_2 be two maximally disjoint connected sets. Initially all nodes are in S_1, and S_2 is an empty set. A node i can be either in S_1 or S_2 in each transmission schedule. That is, for a given slot t, S_2 contains all nodes that transmit in slot t and S_1 contains all nodes that receive in slot t.

Each node v_i is associated with two variables: x_i and y_i. The former denotes the total number of nodes in S_1 with a link to v_i. On the other hand, the variable y_i denotes the total number of nodes in S_2 that link to node v_i. As an example, consider the network shown in Figure 2. Initially all nodes are in S_1. The (x_i, y_i) value of each

node is $(2,0),(3,0),(2,0),(2,0),(3,0),(2,0)$ for node A to F respectively. After slot-1, the (x_i, y_i) value for node A to F becomes $(2,0),(0,3),(2,0),(0,2),(3,0),(0,2)$, respectively. Lastly, we define Δ_i to be the difference between x_i and y_i; i.e., $\Delta_i = x_i - y_i$. Table I gives a summary of the notations used throughout this paper.

IV. The Problem

The link scheduling problem is to maximize network capacity by maximizing the number of links activated in each time slot and to compute a minimal superframe length. Then, the problem at hand can be formulated as follows:

$$\textbf{MIN} \sum_{k=1}^{|\mathcal{B}|} \lambda_k \tag{1}$$

$$\sum_{t=1}^{|\mathcal{B}|} \lambda_t e^t \geq \mathbf{1} \tag{2}$$

In the said problem, the goal is to determine a combination of transmission sets that ensure all links are activated once. However, deriving a maximum transmission set $\mathbf{e^t}$ constitutes solving the NP-complete, MAX-CUT [6] problem. In this paper, we are interested in computing the optimal family of transmission sets in a distributed manner. Specifically, we seek to derive the minimal transmission sets, or equivalently the shortest superframe length, as reflected in Equ. 1, that affords each link at least one activation slot, see Equ. 2 using only one hop neighbor information. Note, in this paper, we do not consider link load or queue length, and defer the development of a suitable scheduler to a future work.

V. Solution

Our solution is based on Algo-2, a centralized scheduler [8]. Our solution, called Algo-d, divides the topology into two maximally connected sets S_1 and S_2, but in a distributed manner. The key idea is to determine and update the x and y value of each node using only local information. The node i with the maximum and positive Δ_i value amongst its 1-hop neighbors broadcast a message to all its neighbors informing them that it will move to S_2 to become a transmitter. Then all nodes update their (x, y) value. A node concludes it has the schedule, i.e., $\mathbf{e^t}$, for a slot t when the Δ value of all one-hop neighbors is equal or less than zero. The algorithm contains two layers: channel access and link scheduling. We now present the details of each layer.

A. Channel Access

Nodes use a random channel access scheme to exchange information with their neighbors. This is required as nodes do not yet have an assigned slot. We will denote this scheme as $TRANSMIT(msg, t_s)$, where *msg* is any message listed in Table II, and t_s is the time slot in which the schedule generated with this message will be executed. Note that channel access is only used by the link scheduler, meaning once a slot is allocated, nodes transmit/receive in their allocated slot(s).

The scheme works as follows. Assume at time slot t, node i needs to send information to its neighbors. It, therefore, sets itself to transmit in a slot chosen randomly in the range $[t, t + kW]$. Here W is the maximum degree of nodes. The term k is a constant. For a node i, the probability of a successful transmission in each slot is

$$P_s = 1 - (\frac{1}{kW})^(N-1) \tag{3}$$

Here N is the number of contending nodes. If a collision occurs, i.e., there is no acknowledgement, when node i transmits in slot t_1, it sets itself to transmit again in another slot chosen in the range $[t_1, t_1 + kW]$. This is carried out for a maximum of MAX_{rtx} times. In our simulations, described in Section V, we used $k = 2$ and $MAX_{rtx} = 3$.

B. Link Scheduling

This section explains how Algo-d determines the link schedule. Each node can be in any of these five states: **BootUp**, **TxXY**, **WaitXY**, **TxInset2**, **ScheduleEnd** and **ScheduleComplete**. Figure 3 shows the corresponding state diagram. Each node maintains the following tuple: $[ID, Set, State, Slot]$. Each node has a unique ID. The *Set* variable stores the set which it belongs to, i.e., Set=1 if it is in S1 or Set=2 if in S2. The variable $State$ stores a node's current state, and $Slot$ represents the time slot it is deriving a link schedule for. Table II shows the list of messages exchanged between nodes and a short description of each state is shown in Table III.

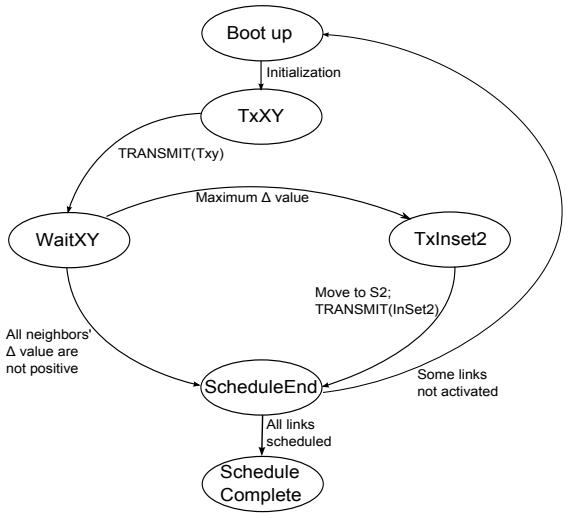


Fig. 3: State diagram of Algo-d

Message	Description
Txy	To transmit (x_i, y_i) value to all neighbours.
InSet2	To inform all neighbors that the sender is moving to S2.
Updxy	To inform all neighbors that the sender's (x_i, y_i) value has changed.
SchComp	To inform all neighbors the schedule is generated

TABLE II: Description of messages

Consider node i. To generate the link schedule for slot t_s, it starts from the **BootUp** state, and creates the tuple

State	Description
BootUp	Initial state or fictional state
TxXY	Tuple $[i, 1, BootUp, t_s]$ has been initialized. Ready to exchange (x,y) with neighbors
WaitXY	Have sent (x, y) to all neighbours; wait to receive (x, y) or Δ from neighbours.
TxInset2	Ready to send InSet2 to all neighbors
ScheduleEnd	Wait for other nodes to finish their schedule for the current slot
ScheduleComplete	Wait for other nodes to activate all their links

TABLE III: Description of each state

$[i, 1, BootUp, t_s]$. After initialization, it moves to the **TxXY** state. It then transmits its (x, y) value to all its neighbors; i.e., by calling $TRANSMIT(Txy, t_s)$. After that, it moves to the **WaitXY** state. Node i then checks if it has received all its neighbors' (x, y) value. If it has, node i will then check if its Δ value and all its neighbors' Δ value are equal to or less than zero. If both are true, then node i moves to the **ScheduleEnd** state. On the other hand, if node i's Δ value is positive, it will check if its Δ value is maximum within its one-hop neighbors. If it is, it then checks whether it has the lowest ID among its neighbors that have the same Δ value. If there are no neighbors with the same Δ value or node i has the lowest ID, it will move to the **TxInset2** state. Otherwise, node i will stay in the **WaitXY** state. If node i is in the **TxInset2** state, it firstly moves itself to S_2, meaning it will transmit in slot t_s. Then node i sends an InSet2 message to all its neighbors to inform them that it will be in S_2. After that node i's neighbors that received the InSet2 message update their Δ value and inform their neighbors via $TRANSMIT(Updxy, t_s)$.

When all nodes enter the **ScheduleEnd** state, they have the link schedule for slot t_s. That is, in slot t_s, nodes in S_2 transmit, and those in S_1 receive. Each node then removes the links that are activated in slot t_s. After that, each node checks if all its links have been activated. If no, it moves to the **BootUp** state to generate the link schedule for slot $t_s + 1$. Otherwise, it moves to the **ScheduleComplete** state.

If node i is in the **ScheduleComplete** state and all its neighbors are in this state, it uses $TRANSMIT(SchComp, t_f)$ to tell all its neighbors that the scheduled slots will start in t_f slots time. The value of t_f is set to the network diameter.

We will now show how Algo-d determines the schedule for the 'two-boxes' topology shown in Fig 4a. Initially, all nodes are in the **BootUp** state and each of them sets a tuple $[i, 1, BootUp, 1]$. Then node i transmits its (x_i, y_i) value to its neighbors, i.e., $TRANSMIT(Txy = (x_i, y_i), 1)$, and moves to the **WaitXY** state. The (x_i, y_i) value of each node is $A(3, 0)$, $B(5, 0)$, $C(3, 0)$, $D(3, 0)$, $E(5, 0)$ and $F(3, 0)$. After each node receives all its neighbors' (x_i, y_i) value, it finds that all its neighbors' Δ_i value is positive. Each node then checks if it has the maximum Δ_i value among its neighbors. Node B realizes that it has the maximum Δ_i value and lowest ID among its neighbors. It thus moves to S_2 and $TRANSMIT(InSet2, 1)$ to all its neighbors. As node A, C, D and F do not have a maximum Δ_i value as compared to their neighbors, they remain in the **Waitxy** state. Although node E knows it has the maximum Δ_i value, its ID is not the lowest, i.e., larger than B. So it also stays in the **Waitxy** state. Nodes which receive the $InSet2$ message then updates its own (x_i, y_i) value as $A(2, 1)$, $C(2, 1)$, $D(2, 1)$, $E(4, 1)$ and $F(2, 1)$. After updating, each node informs its neighbors its new (x_i, y_i) value using $TRANSMIT(Updxy, 1)$. Node E then realizes that it has the largest and positive Δ_i value and moves to S_2 and use $TRANSMIT(InSet2, 1)$ to inform all its neighbors. Each of node E's neighbors then updates its own (x_i, y_i) value. As all nodes observe the Δ_i value of their respective neighbors to be less than zero, the schedule for slot 1 is determined. That is, nodes in $S_2 = \{B, E\}$ will transmit in slot-1 while those in $S_1 = \{A, C, D, F\}$ receive. After that, each node removes the links activated in slot-1; see Figure 4b. Nodes that have un-activated links move to the **BootUp** state and restart the process to generate a link schedule for slot-2.

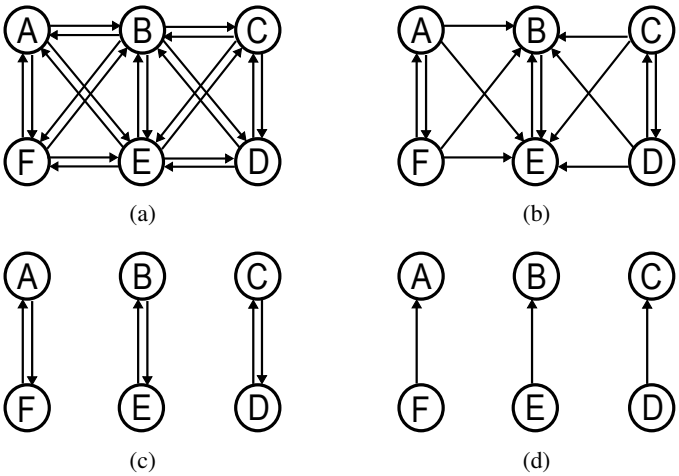


Fig. 4: Two-boxes topology

Each node then update its tuple as $[i, 1, BootUp, t_s]$ and repeat previous procedure to generate the schedule for slot 2, 3 and 4 respectively, as shown in Figure 4b, Figure 4c and 4d. This procedure repeats until all nodes enter the **ScheduleComplete** state. As a result, in slot 1, node B and E transmits and other node receives. In slot 2, node A, C, D and F transmits and node B, E receives. In slot 3, node A, B, and C transmits and other node receives. In slot 4, node D, E and F transmits. The final superframe length and links activated in each slot is exactly the same as Algo-2. Thus in this topology, Algo-d is as optimal as Algo-2.

We note that the number of time slots used to generate the schedule is associated with the maximum degree or neighbors. Consider a network consisting of $|V|$ nodes with a maximum degree N_v. In the worst case, only one neighbor of v moves to S_2 each time, meaning it takes $N_v - 1$ information exchanges for node v to work out a schedule.

VI. Evaluation

We evaluated the performance of Algo-d in Matlab using the Matgraph [15] toolkit. All nodes are stationary, randomly connected and each node has sufficient number of radios to communicate with all its neighbors. We compare Algo-d

to Algo-1 [6], Algo-2 [8] and JazzyMac [9]. Briefly, these algorithms work as follows.

- Algo-1, a centralized scheduler, recursively generates a MAX-CUT in every other slot. Nodes that transmit in slot t_i become receivers in slot $t_i + 1$.
- Algo-2, a centralized scheduler, generates a MAX-CUT in every slot. Nodes in one set transmit and nodes in the other set receive. It then removes activated links and generate a new MAX-CUT for the next time slot.
- JazzyMac, a distributed scheduler, assigns each link a token. A node transmits only when it has the token for all its links.

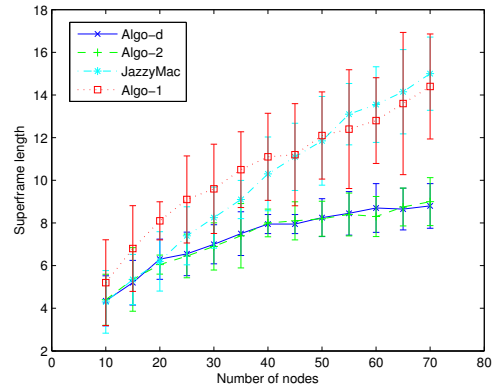
We vary the number of nodes, from 10 to 70, and degree of each node, from 3 to 7, to compare their performance. Our results are an average of 20 simulation runs, each experiment with a different topology. In each experiment, we compute the following metrics:

- *Superframe length*, which corresponds to the number of slots required to activate all links at least once.
- *Network capacity*. This is the average number of links activated in each time slot, and is calculated by summing the number of links activated in each slot and dividing the resulting sum by the superframe length.

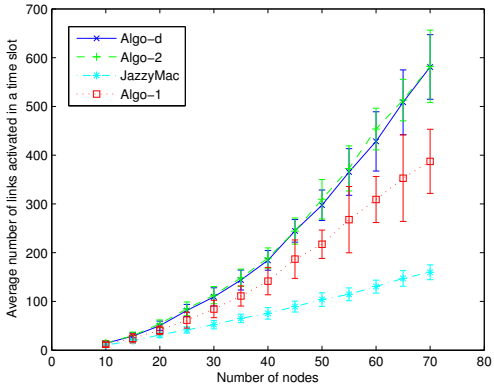
We also plot the confidence interval of 20 simulation runs, where 95% of the results are within the indicated error bar.

A. Node Density

We first study the impact of node density. In this experiment, we tested the algorithms on a topology with 10 to 70 nodes. Each node establishes a bidirectional link to another node with probability 0.5. From Figure 5, we can see that Algo-1, Algo-2 and our algorithm have similar performance when there are 10 to 20 nodes. The superframe length and the average number of links activated in a time slot increase with the number of nodes. However, for Algo-1 and JazzyMac, their superframe length increases much quicker than Algo-2 and Algo-d. When the network size is 70, the superframe length of Algo-1 and JazzyMac is about 60% longer than the superframe length generated by Algo-d. Also, Algo-2 and Algo-d can schedule more links in each slot than Algo-1 and JazzyMac. When the network size is 70, Algo-2 and Algo-d can activate 50% more links than Algo-1 and 264% more links than JazzyMac. This is because Algo-1 uses a 2-phase transmit/receive scheme that generates link schedule every other time slot. This results in an inefficient slot usage and longer superframe length. The 2-phase transmit/receive scheme doubles the standard deviation which is shown in the figure with a wide error bar. The poor performance of JazzyMac is due to the fact that a node must wait until it holds the token of all its links before it can start transmission. This results in large number of idle links. Thus the superframe length increases quicker and the number of links activated grows slower than other algorithms.



(a)



(b)

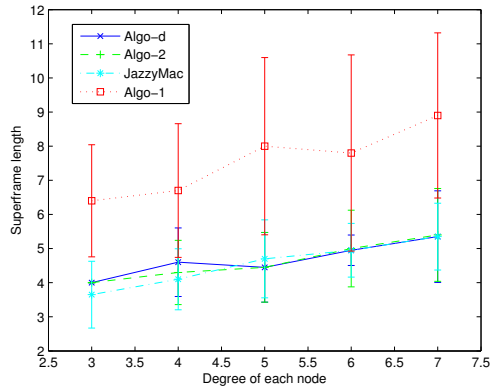
Fig. 5: Performance under different node densities

B. Node Degree

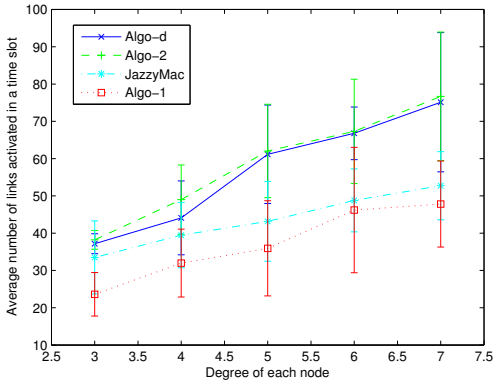
In the second experiment, we study the effect of node degree. We fix the network density at 40 nodes. We then vary the node degree from 3 to 7. Results are shown in Fugure 6. From Figure 6 we can see that Algo-d, Algo-2 and JazzyMac have similar superframe lengths in all experiments and Algo-1 does not perform well. Algo-1's superframe length is about 1.8 times of other algorithms. JazzyMAC, Algo-2 and our algorithm schedule similar number of links when the degree of each node is three. However, Algo-2 and Algo-d schedule 46% more links than JazzyMAC and 58% more links than Algo-1 when each node has seven neighbors. This is because Algo-1's two-phase transmit/receive scheme leads to a longer superframe length and wider confidence interval. As mentioned earlier, nodes JazzyMac are only able to transmit when they hold all tokens. Consequently, there are fewer links activated per slot.

VII. Conclusion

In this paper, we have proposed a distributed link scheduling algorithm called Algo-d to generate the superframe with minimal length for a given MTR WMN topology, and also to maximize the number of links activated in each time slot. Our



(a)



(b)

Fig. 6: Performance under different node degrees

results show that Algo-d has similar performance to Algo-2, which is a centralized algorithm that solves the said problem in a centralized manner. Algo-d activates at most 7% fewer links than Algo-2. Also, Algo-d schedules 50% and 264% more links in each slot as compared to Algo-1 and JazzyMac respectively. As a future work, we plan to modify Algo-d to guarantee end-to-end throughput.

REFERENCES

[1] L. Bao and J. Garcia-Luna-Aceves. A New Approach to Channel Access Scheduling for Ad Hoc Networks. In *ACM MOBICOM*, Rome, Italy, July 2001.

[2] L. Bao and J. Garcia-Luna-Aceves. Receiver-Oriented Multiple Access in Ad Hoc Networks with Directional Antennas. *Wireless Networks*, 11:67–79, 2005.

[3] O. Bazan and M. Jaseemuddin. A Survey on MAC Protocols for Wireless Ad Hoc Networks with Beamforming Antennas. *IEEE Communications Surveys & Tutorials*, 14(2):216–239, 2012.

[4] C.-T. Chang, C.-Y. Chang, and Y.-J. Lu. Maximizing Throughput by Exploiting Spatial Reuse Opportunities with Smart Antenna Systems. In *IEEE International Conference on Communications (ICC)*, Cape Town, South Africa, May 2010.

[5] K.-W. Chin, S. Soh, and C. Meng. A Novel Scheduler for Concurrent Tx/Rx Wireless Mesh Networks with Weighted Links. *IEEE Communications Letters*, 16:246–248, 2012.

[6] K.-W. Chin, S. Soh, and C. Meng. Novel Scheduling Algorithms for Concurrent Transmit/Receive Wireless Mesh Networks. *Computer Networks*, 56:1200–1214, 2012.

[7] L.-L. Hung, S.-H. Wu, and C.-T. Chang. Maximizing Throughput for Delay-Constraint Transmissions with Smart Antenna Systems in WLANs. In *Fifth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS)*, Seoul, Korea, June 2011.

[8] H.-Y. Loo, S. Soh, and K.-W. Chin. On Improving Capacity and Delay in Multi Tx/Rx Wireless Mesh Networks. In *IEEE Asia Pacific Conference on Communications (APCC)*, Bali, Indonesia, Aug 2013.

[9] S. Nedevschi, R. K. Patra, S. Surana, S. Ratnasamy, L. Subramanian, and E. Brewer. An Adaptive, High performance MAC for Long-Distance Multihop Wireless Networks. In *Proceedings of the 14th ACM international conference on Mobile computing and networking*, San Francisco, USA, Sept. 2008.

[10] P. H. Pathak and R. Dutta. A survey of network design problems and joint design approaches in wireless mesh networks. *IEEE Communications Surveys & Tutorials*, 13(3):396–428, 2011.

[11] J. Qiao, L. X. Cai, J. W. Mark, et al. STDMA-Based Scheduling Algorithm for Concurrent Transmissions in Directional Millimeter Wave Networks. In *IEEE International Conference on Communications (ICC)*, Ottawa, Canada, June 2012.

[12] J. Qiao, L. X. Cai, X. S. Shen, and J. W. Mark. Enabling Multi-Hop Concurrent Transmissions in 60 GHz Wireless Personal Area Networks. *IEEE Transactions on Wireless Communications*, 10:3824–3833, 2011.

[13] B. Raman and K. Chebrolu. Design and Evaluation of a New MAC Protocol for Long-Distance 802.11 Mesh Networks. In *Proceedings of the 11th annual international conference on Mobile computing and networking*, Cologne, Germany, Aug. 2005.

[14] I. Rhee, A. Warrier, J. Min, and L. Xu. DRAND: Distributed Randomized TDMA Scheduling for Wireless Ad-Hoc Networks. In *Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing*, Florence, Italy, May 2006.

[15] E. R. Scheinerman. Matgraph: a matlab toolbox for graph theory. *Software available at: http://www. ams. jhu. edu/˜ ers/matgraph*, 2007.

[16] Y. Wang, D. M. Chiu, and J. C. Lui. Characterizing the Capacity Gain of Stream Control Scheduling in MIMO Wireless Mesh Networks. In *IFIP NETWORKING*, volume 4479, pages 311–321. Springer, 2007.