A Distributed Collision-Free Low-Latency Link Scheduling Scheme in Wireless Sensor Networks

Chao Wang Kuo-Feng Ssu
Institute of Computer and Communication Engineering, National Cheng Kung University, Tainan, Taiwan 70101
sean@dcl.ee.ncku.edu.tw, ssu@ee.ncku.edu.tw

***Abstract*—In order to guarantee collision-free transmissions in TDMA-based wireless sensor networks, a substantial amount of work in literature has been done by modeling the problem into minimum graph coloring. However, our observation reveals that such approach is not effective towards low-latency transmissions due to the inherent principle of the graph coloring.**

This paper introduces DCLS, a distributed collision-free low-latency link scheduling scheme. The scheme considers the network snapshot at each time slot, and determines a set of collision-free transmission pairs on each snapshot. With DCLS, the delay is asymptotically smaller than that with the graph coloring model, and the running time complexity on each snapshot is $O(diam)$, where $diam$ is the diameter of the network graph. From the simulation result, the delay is significantly reduced in the network of maximum degree Δ ranged from 6 to 18, and the duty cycle is 0.23 in average.

I. Introduction

Wireless sensor networks have been a lively research topic in recent years owing to their versatile applications. These networks are employed for tasks such as event detecting, target tracking and other scientific purposes. Each sensor node is powered by limited battery-supplied energy. Nodes are assumed to be disposed after they are out of battery. In order to achieve longer network lifetime, the design of an energy efficiency transmission protocol should be taken into great considerations.

Packet-collision-avoidance is one of the fundamental issues towards energy efficiency in wireless sensor networks [1]. Collided packets have to be discard and re-transmitted so those packets not only cause throughput degradation, but increase energy consumption. Packets collide due to signal interferences, which can be divided into two types: *primary* and *secondary interference* [2]. Primary interference occurs when a node performs multiple operations in the same time. For example, a node receives packets from multiple transmitters. Secondary interference occurs when a receiver v is within the ranges of other transmitters which are not intended to transmit data to v.

Time Division Multiple Access (TDMA) is a technique to manage media access control, and it is also a sensible method to reduce packet collision. In TDMA, time is equally divided into intervals called *frames*, and each frame is further divided into *time slots*. The collision-free TDMA protocols can be classified into two categories: *broadcast* and *link* scheduling [2]. In broadcast scheduling, a scheduled node requires that all nodes within its two hops be assigned different time slots. In link scheduling, two links which are adjacent or connected by a third link are not allowed to share the same time slots. With the above constraints, the generated schedule can guarantee collision-free.

This research was supported in part by Delta Electronics, Inc. and in part by the Taiwan National Science Council (NSC) under contracts NSC 97-2628-E-006-093-MY3, 97-2221-E-006-176-MY3, and 97-2221-E-006-146-MY3.

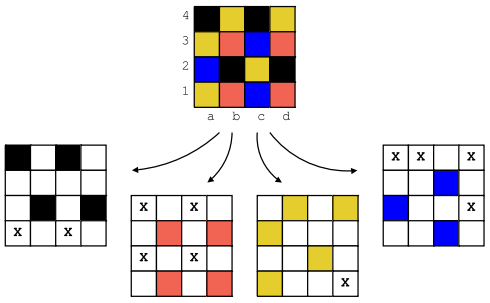

Fig. 1. On the top is a possible result of coloring. For each kind of color there are hidden colorable squares. By examining each color individually, all hidden squares are exposed (marked with X).

Many TDMA scheduling algorithms apply the concept of *graph coloring*. The goal is to properly paint all nodes/edges with minimum number of colors, and also meet the broadcast/link scheduling constraint. Each color is mapped to a different time slot, and the schedule length is equivalent to the number of colors. The concept of graph coloring correctly models the scheduling constraint, but the simple reduction of TDMA scheduling neglects some opportunities of time division. Specifically, the graph coloring approach does not effectively utilize the network bandwidths due to a phenomenon that we name *hidden squares*.

The phenomenon is illustrated by an example shown in Fig. 1. Consider a map with 4×4 squares, the goal is to paint each adjacent square with a different color. Fig. 1 shows a possible solution using 4 colors. By examining each color individually, it is clear that there are some squares nonadjacent to any square of given color. For the black color, the squares positioned at a1 and c1 are nonadjacent to any black square. Because the squares are already painted by other colors, they cannot be painted black. The squares with such characteristic are called *hidden squares* of a given color.

Hidden squares likely appear in the graph coloring which uses many different colors. In strong edge-coloring variation, the typical model for TDMA link scheduling, it is NP-hard

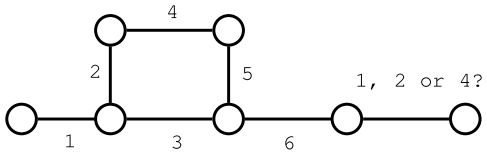

Fig. 2. The digit on each edge represents its color type. To use minimum number of colors, the rightmost edge must choose either type 1, 2 or 4; however choosing one of them leads to hidden squares of the other two colors.

to determine the minimum number of colors [3], and only a general upper bound of the number is known for sufficiently large Δ [4]. Moreover, even in graphs which use the minimum number of colors, the phenomenon is sometimes inevitable. For example, the graph in Fig. 2 requires at least six colors, and the hidden-square phenomenon arises after painting the rightmost edge.

Hidden-square phenomenon causes an undesirable effect in wireless sensor networks. Certain time slots could not be assigned to nodes even if such assignments make no collision; if these assignments were done, corresponding nodes could have less transmission delay, specifically the time spent on waiting for next available time slot to transmit. Since the transmission delay is critical in many applications of wireless sensor networks, it is worth studying an alternative TDMA scheduling approach for low-latency transmissions.

This paper introduces a distributed collision-free low-latency link scheduling scheme called DCLS. The scheme treats the scheduling problem in an aspect different from strong edge-coloring. In strong edge-coloring, algorithms are performed on a given network topology; DCLS considers the network topology at each time slot, and finds a set of collision-free transmission pair on each network snapshot. As the result, each node may have multiple time slots for transmission in one schedule length, and the delay can be improved.

II. Related Work

There are strong relations between TDMA scheduling and graph coloring. Specifically, the broadcast scheduling relates to vertex coloring [5], and the link scheduling could be modeled into edge coloring. Furthermore, the latter problem could be associated with *valid edge-coloring* for directed links [6] and *strong edge-coloring* for undirected links [7]. In this paper, the focus is on undirected link scheduling.

Gandham et al. proposed a link scheduling algorithm which uses at most $2(\Delta + 1)$ time slots for directed acyclic network topologies [6], and the schedule length is close to $2(\Delta + 1)$ for directed sparse graphs with cycles. Barrett et al. presented sequential and distributed algorithms for channel assignment in wireless radio networks [7], in which the goal is to construct a $O(1)$-approximation to *maximum distance-2 matching* so as to achieve possible maximum concurrent transmissions. The running time complexity is $O(\rho \log |V|)$ rounds for each color.

Ramanathan and Lloyd developed algorithms for link scheduling and broadcast scheduling on both restricted and arbitrary graphs [2]. The schedule length is $O(\theta^2)$ times the optimum for link scheduling, where θ is the minimum number of planar graphs decomposed from the network.

III. Preliminaries

The unit disk model of the network is assumed. All nodes are equipped with omnidirectional antenna with transmission range r, so each node v covers a circular region D_v with radius r. Node v, w can communicate with each other if and only if $v \in D_w$ and $w \in D_v$.

Transmission by a node is always a broadcast and all nodes within the transmission range can receive the signal. Node v can receive the message from node w if and only if itself keeps silent and w is the only transmitting node among neighbor nodes of v. Otherwise node v hears only noise. Each node has a unique ID and knows the IDs of its neighbors, i.e. its one-hop neighbor nodes. A message could be sent to the specific node by wrapping the receiver's ID within the packet.

In link scheduling, time slots are assigned to *transmission pairs*. A transmission pair involves one transmitting node and one receiving node. A node is said to *own* the time slot if that slot is assigned to the node. Assigning a time slot to an edge is equal to assigning that slot to the nodes at both ends of that edge. The node can transmit data only during owned time slots.

Let $G = (V, E)$ be an undirected graph corresponding to the network topology. For each node $v \in V$, let $d(v)$ be the set of one-hop neighbor nodes of v and $\delta_v = |d(v)|$. The $distance(v, u)$ is the shortest path length from node v to node u. The $diameter$ (or $diam$) is the maximum $distance(v, u)$ taken over all pairs (v, u). Denote T_v as the set of owned time slots of v.

Definition 1 (Saturation Condition)**.** *A node is said to be* saturated *if each of its edges has been assigned at least one time slot respectively. The notation $|T|$ means the number of time slots needed for all nodes to be saturated.*

Definition 2 (Link Scheduling Criteria)**.** *The link scheduling is said to be* collision-free *if the schedule satisfies $\forall v \in V$, $i, j \in d(v)$, $i \neq j$ such that:*

$$T_v \cap T_i \cap T_j = \phi. \tag{1}$$

The above equation means that the owned time slots are mutually exclusive among all transmission pairs involving node v. No two transmission pairs of the same time slot could be overlapped or adjacent.

For example, in Fig. 3, one possible collision-free link schedule is: $T_A = \{2, 6\}$, $T_B = \{1, 2, 3\}$, $T_C = \{5, 6\}$, $T_D = \{3, 4, 5\}$, and $T_E = \{1, 4\}$.

Definition 3 (Data Buffering Delay)**.** *Data buffering delay is the number of time slots for which a node has to wait before next packet transmission to the same receiver.*

Given node v and node $u \in d(v)$, the data buffering delay $D_{v,u}$ is:

$$D_{v,u} = |T|/|T_v \cap T_u| \tag{2}$$

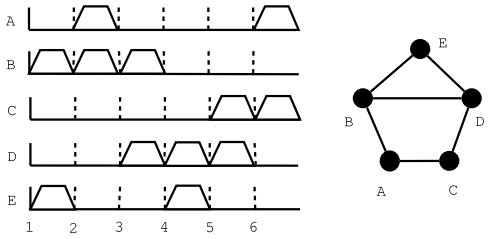

Fig. 3. An example of a collision-free link schedule.

and the average delay on node v and on the network are:

$$D_v = (\sum_{u \in d(v)} D_{v,u})/\delta_v \quad (3)$$

$$D_{net} = (\sum_{v \in V} D_v)/|V|. \quad (4)$$

In the strong-edge-coloring-based algorithm, $|T_v \cap T_u| = 1$ for each transmission pair (v, u), so accordingly $D_{v,u} = |T|$, $D_v = |T|$, and $D_{net} = |T|$. However, $|T_v \cap T_u| \geq 1$ in the proposed scheme, so with the same $|T|$ the data buffering delay is smaller.

The schedule length of strong edge-coloring is $O(s_{\chi'}(G))$, where $s_{\chi'}(G)$ is called *strong chromatic index*. Molloy and Reed proved a general upper bound of $s_{\chi'}(G)$ for sufficiently large Δ:

$$s_{\chi'}(G) \leq 1.998\Delta^2. \quad (5)$$

For a tighter upper bound, there is a still widely open conjecture [3]: for any graph G of maximum degree Δ, $s_{\chi'}(G) \leq f(\Delta)$, where $f(\Delta)$ is defined as follows:

$$f(\Delta) = \begin{cases} 5\Delta^2/4 & \text{if } \Delta \text{ is even.} \\ 5\Delta^2/4 - \Delta/2 + 1/4 & \text{if } \Delta \text{ is odd.} \end{cases} \quad (6)$$

IV. The Scheme

The scheme first determines the schedule length of network (the number of time slots). DCLS then selects collision-free transmission pairs over each network snapshot; a snapshot is the network topology at the given time slot. For instance, Fig. 4 shows two possible sets of transmission pairs in a snapshot. In each snapshot every node executes three sub-routines: **Initialization**, **Message_passing**, and **State_transition**. The pseudocode is given in Table I and the flow diagram is shown in Fig. 5. The $state$ variable is added by 1 at the end of **State_transition**, except it is assigned a value explicitly. All nodes are synchronized at the end of each snapshot. As the result of the mechanism on all snapshots, each node owns some time slots and can transmit data during these slots without collision. The sketch of collision-free transmission pair selection is shown in Fig. 6.

A. Collision-free transmission pair selection

In the first step of the selection phase, all nodes are marked the color *white*. DCLS then generates an independent node set based on a straightforward method [8]. Each node generates a unique random number and compares it with all neighbor nodes. The node with the smallest number is included in the independent set. Repeat the above subroutine $O(1)$ rounds.

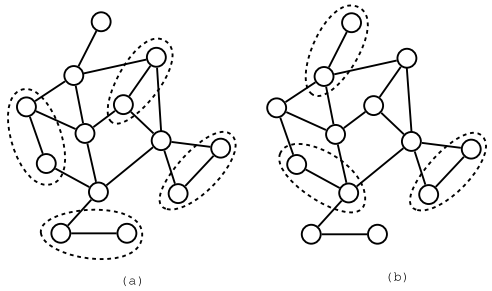

Fig. 4. Two possible sets of transmission pairs on a snapshot. Each transmission pair is circled by dotted line.

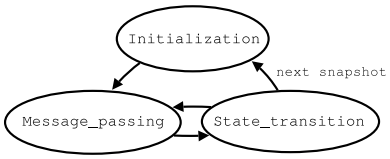

Fig. 5. The flow diagram of the sub-routines.

TABLE I
Pseudocode of Collision-Free Transmission Pair Selection

Initialization$_i$:
1 $color \leftarrow$ *white*
2 $state \leftarrow 1$
3 $status \leftarrow$ *idle*
4 $requester \leftarrow$ NIL
5 $reject\_count \leftarrow 0$
6 generate an independent node set.
7 **if** *i* is in the independent set **then**
8 $color \leftarrow$*black*

Message_passing$_i$:
1 **case** $state$
2 1 : each *black* node sends *together(i)* to its neighbors.
3 2 : each *available* node sends *available(i)* to j.
4 3 : each *black* node sends *request(i)* to j.
5 4 : **if** $\{color =$*white*$\} \wedge \{requester \neq$ NIL$\}$ **then**
6 send *reject(i)* to *requester*.
7 **else**
8 send *color* to all neighbors.
9 5 : each node sends *prepared(i)* to its neighbors.
10 **endcase**

State_transition$_i$:
1 **case** $state$
2 1 : **if** receive *together(j)* from only one neighbor j **then**
3 $status \leftarrow$ *available*
4 2 : **if** $\{color =$ *black*$\} \wedge \{$receive *available(j)*$\}$ **then**
5 $choice \leftarrow j$ // randomly pick one of the senders
6 **else** $color \leftarrow$ *white*
7 3 : **if** receive *request(j)* **then**
8 $color \leftarrow$ *gray*
9 $requester \leftarrow j$
10 4 : **if** $\{color =$ *gray*$\} \wedge \{$ receive color *gray*$\}$ **then**
11 generate an independendt node set among adjacent gray nodes.
12 $state \leftarrow 4$
13 5 : **if** receive *reject(j)*
14 **if** $reject\_count < \alpha$
15 $reject\_count \leftarrow reject\_count + 1$
16 $state \leftarrow 1$
17 **else** $color \leftarrow$*white*
18 **if** receive *prepared()* from all neighbors **then**
19 **if** $color =$ *black* $\vee$ *gray* **then**
20 own the current time slot.
21 $status \leftarrow$ *done*
22 synchronize with all nodes, and then begin a new snapshot.
23 **else if** receive *together(j)* from only one neighbor j **then**
24 $status \leftarrow$ *available*
25 $state \leftarrow 2$
26 **endcase**

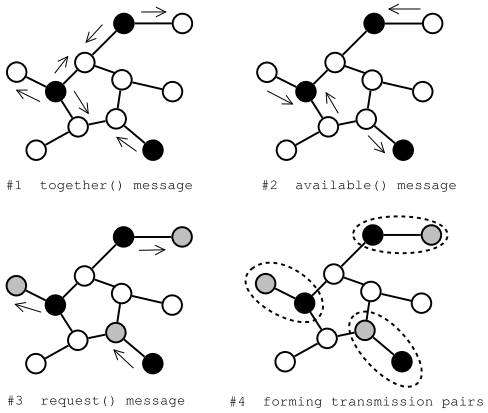

Fig. 6. The procedure on selecting collision-free transmission pairs.

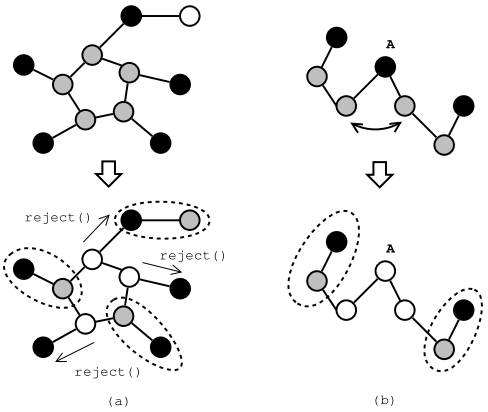

Fig. 7. (a) DCLS generates an independent set among adjacent gray nodes, and the nodes in the set could form transmission pairs. The other gray nodes turn *white* and send *reject*() back to their requesters, force which to request other available nodes. (b) Node *A* circularly requests two neighbors. It will cease and turn *white* after receiving *reject*() α times; other nodes then could form transmission pairs.

At each round all nodes in the set and their neighbors are excluded from joining the next round. Eventually, all nodes in the independent set mark themselves the color *black*. Note that in this step DCLS does not try to form a maximal independent node set, because it will cost $O(\log^2 |V|)$ rounds to find a maximal independent set [8].

Next, all black nodes broadcast *together*() messages to their neighbor nodes. For a node which has received *together*() and has no other black neighbors, the node is called an *available* node. Each available node sends *available*() message back to the corresponding black nodes. Each black node then sends *request*() message to one of the available neighbors to request for forming a transmission pair. If there is no available neighbor node, that black node marks itself back to *white*.

Once a node has received *request*() , it marks itself *gray* and denotes the sender as its requester. Each node then exchanges color information with its neighbors. If a gray node receives some *gray*s (which means some of its neighbors are requested by black nodes elsewhere), all adjacent gray nodes will form a maximal independent node set. The gray nodes in the set are said to be *prepared* to form transmission pairs with their requester nodes. The other gray nodes send *reject*() messages back to their requesters, forcing which to request other available nodes. The above scenario is shown in Fig. 7a. To avoid a black node circularly sending *request*() to some of its gray neighbor nodes (Fig. 7b), after receiving *reject*() α (a constant) times, the black node will cease and then mark itself *white*.

When all gray nodes are *prepared*, each black node and the corresponding gray node then form a transmission pair. Hence a set of transmission pairs on the snapshot are constructed, and all nodes belonged to transmission pairs *own* the time slot. DCLS then repeats the selection phase for next snapshot.

B. Consideration on independent sets

When an independent node set is generated randomly, it is possible that some nodes are hardly included in the set and thus have little chance to form transmission pairs. Under such situation, it is difficult to know whether the determined schedule length is long enough to exceed $|T|$. Choosing a fairly long schedule length could solve the dilemma, but such approach requires the heavier computation.

DCLS overcomes this difficulty by introducing an offset to the random number. Once the node is *saturated* (refer to Definition 1), it should leave the competitions of independent node set in the subsequent snapshots, and therefore gives other nodes higher probabilities to be included in the set. Based on above consideration, DCLS shifts the range of random number of a saturated node by an offset. For example, each node initially generates a random number between 1 and 1000; once after the node is saturated, 1000 is added to the random number for computing an independent set, and hence it certainly will not be included in the set. When all nodes are saturated, all random numbers then are ranged from 1001 to 2000, and therefore all nodes once again have the same probabilities to be included in the independent set.

Now it is the proper time to revisit the claim in Section III that $|T_v \cap T_u| \geq 1$ for DCLS. Two reasons support the claim. First, a node is possible to be included in an independent set and marked *black* in multiple snapshots, and each time a black node has chance to form a transmission pair with the same neighbor node. Second, since the nodes at both ends of each edge are likely to become black nodes (at different snapshots), a node is possible to form a transmission pair due to its black neighbor. Based on above reasons, it is likely that $|T_v \cap T_u| \geq 1$. Moreover, if the saturation condition includes one more prerequisite that at each time slot assignment the node itself is *black*, then $|T_v \cap T_u| \geq 2$. Further analysis on the value of $|T_v \cap T_u|$ is provided in Section V.

V. ANALYSIS

Theorem 1. *DCLS satisfies the link scheduling criteria.*

Proof: All snapshots are orthogonal to each other, so it is sufficient to consider an arbitrary snapshot and prove that the criteria holds within. The proof is demonstrated by proving

two arguments: 1. there are no overlapped transmission pairs, and 2. no adjacent transmission pairs can exist.

First, since if v receives multiple *together*() messages from different neighbors in the same snapshot, v will not be *available*. As a result, no two or more nodes will request v and form a transmission pair, and the overlapped transmission pairs cannot be formed.

Second, if two gray nodes are adjacent, only one of them can join the corresponding requester node. The other node will send *reject*() to its requester, forcing which to request another neighbor. Thus, no two or more transmission pairs can be adjacent. With arguments 1 and 2, DCLS is proved to satisfy the link scheduling criteria for a static network. ■

Theorem 2. *The running time complexity of DCLS is* $O(diam)$ *for each snapshot.*

Proof: For each snapshot, the independent node set is obtained within constant time. All five control messages, including *together*(), *available*(), *request*(), *reject*(), and *prepared*(), are sent $O(1)$ times. The typical synchronization process at the end of each snapshot needs $O(diam)$ rounds [9]. ■

Lemma 1. *Let* $\mathcal{E}(v)$ *denote the event that node* v *is black and* $\forall w \in d(v)$*, w is white. Then the discrete probability* $\texttt{Pr}(\mathcal{E}(v)) \geq \frac{\epsilon}{\delta_v+1}$, $0 < \epsilon < 1$.

Proof: Because of the independent trails, $\texttt{Pr}(\mathcal{E}(v)) = \frac{1}{\delta_v+1}\Pi_{w\in d(v)}(1-\frac{1}{\delta_w+1})$. Partition D_v into a constant number of unit regions $R_1, ..., R_s$, such that these regions are mutually exclusive. Then for each R_i, and $\forall w \in R_i,\ \delta_w \geq |R_i|$. Hence,

$$\begin{aligned}\texttt{Pr}(\mathcal{E}(v)) &\geq \frac{1}{\delta_v+1}\cdot\prod_{i=1}^{s}\prod_{w\in R_i}(1-\frac{1}{\delta_w+1})\\ &\geq \frac{1}{\delta_v+1}\cdot\prod_{i=1}^{s}(1-\frac{1}{|R_i|})^{|R_i|}\\ &\geq \frac{(\epsilon' e)^s}{\delta_v+1} = \frac{\epsilon}{\delta_v+1}. \end{aligned} \quad (7)$$

■

Theorem 3. *The delay* D_{net} *obtained by DCLS is*

$$O((\Delta+1)(\ln\Delta+O(1))). \quad (8)$$

Proof: Let M be the number of snapshots before a node v could be marked black. Use the result of lemma 1, the expectation of M is:

$$E[M] = 1/\texttt{Pr}(\mathcal{E}(v)) \leq (\delta_v+1)/\epsilon \quad (9)$$

and the probability that v is not marked black for a period c times longer than the expected is

$$\begin{aligned}Pr(M > c\cdot E[M]) &= (1-Pr(\mathcal{E}(v)))^{c\cdot E[M]}\\ &= (1-1/E[M])^{c\cdot E[M]} \leq e^{-c}. \end{aligned} \quad (10)$$

For a given black node v, let M' be the number of snapshots that all neighbors of v were chosen at least once. The problem of determining the expectation of M' could be directly modeled as the *coupon collectors problem* in literature on probability analysis [10]. Assume a uniform probability distribution, and the expectation of M' is

$$E[M'] = \delta_v \cdot H_{\delta_v} \approx \delta_v\cdot(\ln\delta_v + O(1)) \quad (11)$$

where H_{δ_v} is the δ_v-th harmonic number.

Since the computations of DCLS are performed in distributed manner, the schedule length $|T|$ is $O(E[M]\cdot E[M'])$. From (3),

$$D_v = \frac{|T|}{\delta_v}\cdot\textstyle\sum_{u\in d(v)}\frac{1}{|T_v\cap T_u|} \quad (12)$$

and from (4),

$$\begin{aligned}D_{net} &= (\textstyle\sum_{v\in V} D_v)/|V|\\ &= O(E[M]\cdot E[M']/\Delta)\\ &= O((\Delta+1)(\ln\Delta+O(1))). \end{aligned} \quad (13)$$

■

Comparing (8) with (5), it is clear that the delay obtained by DCLS is asymptotically smaller than the one obtained by applying the strong-edge-coloring model.

To analyze the value of $|T_v \cap T_u|$, let ρ_v be the ratio of owned time slots of node v to the schedule length, namely:

$$\rho_v = \frac{\sum_{u\in d(v)}|T_v\cap T_u|}{|T|}. \quad (14)$$

If the value of ρ_v is known, $\forall u \in d(v)$ the average value of $|T_v \cap T_u|$ could be obtained:

$$|T_v\cap T_u|_{avg} = \frac{\rho_v\cdot|T|}{\delta_v}. \quad (15)$$

VI. Simulation Results

The effectiveness of DCLS is evaluated by a simulator written in C language. The simulation scenario is a 200 × 200 meter-square area, and 20∼80 static nodes are randomly deployed with a constraint that the network is fully-connected. The transmission range of each node is set to 50 meters. The maximum degree Δ is ranged from 6 to 18. For each network topology, the experiment result is obtained from the average of 30 runs. To guarantee the schedule length, namely the number of snapshots, is larger than $|T|$, DCLS uses about $2\cdot|T|_{avg}$ snapshots for each run.

A. Performance on data buffering delay

The simulation statistics are given in Table II. For the data buffering delay, the performance of DCLS is shown in column ⟨3⟩. The standard deviation of DCLS is given in column ⟨7⟩. The result shows that the delay value is stable for various network densities.

Recall that the delay obtained by applying strong edge-coloring is equal to $|T|$, which is $O(s_{\chi'}(G))$, so $s_{\chi'}(G)$ is used as the comparison value in simulation. Fig. 8 shows the comparison of data buffering delay under various Δ, in which GC denotes the general upper bound of $s_{\chi'}(G)$ (equation (5)) and GC-conj represents the conjecture on $s_{\chi'}(G)$ (equation (6)); the value of DCLS is obtained by running the

TABLE II
SIMULATION STATISTICS

	⟨1⟩	⟨2⟩	⟨3⟩	⟨4⟩	⟨5⟩	⟨6⟩	⟨7⟩	⟨8⟩	⟨9⟩	⟨10⟩	⟨11⟩
Δ	Nodes	Delay			$\frac{\langle 2\rangle-\langle 3\rangle}{\langle 2\rangle}$	$\frac{\langle 3\rangle}{\langle 4\rangle}$	Std. dev. of ⟨3⟩	Duty cycle	Snapshots	$\lvert T_v \cap T_u\rvert_{avg}$	$\frac{\langle 9\rangle}{2\cdot\langle 10\rangle}$
		1. GC	2. DCLS	3. $(\Delta+1)(\ln\Delta)$							
6	20	71.93	14.62	12.54	79%	1.17	1.06	0.38	100	3.17	15.78
8	30	127.87	25.35	18.71	80%	1.35	1.65	0.3	150	2.81	26.69
10	40	199.8	31.04	25.33	84%	1.23	1.52	0.29	200	2.9	34.48
12	50	287.71	52.18	32.3	82%	1.62	2.12	0.22	400	3.66	54.64
14	60	391.61	66.61	39.58	83%	1.68	1.8	0.18	500	3.22	77.63
16	70	511.49	91.37	47.13	82%	1.94	3.24	0.15	800	3.8	105.26
18	80	647.35	107.21	54.91	83%	1.95	2.77	0.13	1000	3.61	138.5

proposed scheme. It is clear that the delay of DCLS is much smaller than GC or GC-conj. Ratio of the reduced delay is shown in column ⟨5⟩.

B. Performance on power saving

DCLS could be optimized to save energy. Since a node can transmit only at its owned time slots, it is harmless to put a node in sleep mode at other time slots, so that ρ_v is equivalent to the duty cycle. The duty cycle is shown in column ⟨8⟩ of Table II. For example, for the topology with $\Delta = 12$, a node has 78% of the time in sleep mode.

The value of duty cycle increases when Δ decreases, because for each node the probability to be included in an independent set is inversely proportional to the degree of the node. Besides, it is a natural trade-off between low latency and low duty cycle, and with smaller Δ the data buffering delay is also smaller.

C. Other characteristics

The computation complexity of data buffering delay of DCLS is $O((\Delta+1)(\ln\Delta+O(1)))$. Column ⟨6⟩ in Table II gives the estimated coefficient of the highest order term in this asymptotic notation. The value is ranged from 1.17 to 1.95.

Equation (15) can be computed with the simulation results, and the value is shown in column ⟨10⟩ in Table II. According to equations (2), (3) and (4),the value of data buffering delay can be approximately determined:

$$D_{net} \approx \frac{|T|}{\Delta}\cdot\sum_{\Delta}\frac{1}{|T_v\cap T_u|_{avg}} \approx \frac{|T|}{|T_v\cap T_u|_{avg}} \quad (16)$$

and the value is given in column ⟨11⟩ of Table II. This value is close to the one obtained by simulation, as displayed in column ⟨3⟩, and hence it suggests that the simulation result is in consistent with the analytical analysis.

VII. Concluding Remarks

Our observation reveals that graph coloring approach on link scheduling is not effective towards low-latency transmissions. Motivated by the need of low-latency transmissions in wireless sensor networks, this paper develops a distributed collision-free low-latency link scheduling scheme called DCLS. With the scheme, the data buffering delay is significantly reduced,

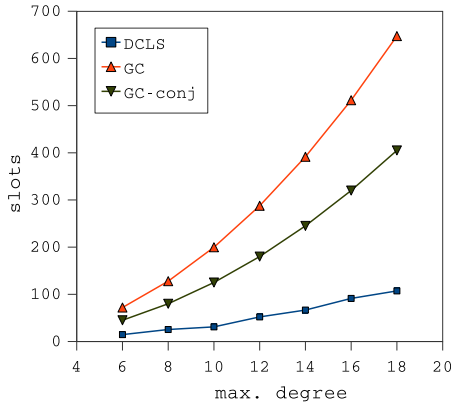

Fig. 8. Data buffering delay.

compared to the strong edge-coloring approach. The execution time complexity of DCLS is $O(diam)$ for each network snapshot. The performance improvement was confirmed by both analysis and simulation.

References

[1] I. Demirkol, C. Ersoy, and F. Alagöz, "MAC protocols for wireless sensor networks: a survey," *IEEE Communications Magazine*, vol. 44, no. 4, pp. 115–121, Apr. 2006.

[2] S. Ramanathan and E. L. Lloyd, "Scheduling algorithms for multihop radio networks," *IEEE/ACM Transactions on Networking*, vol. 1, no. 2, pp. 166–177, Apr. 1993.

[3] M. Mahdian, "The strong chromatic index of graphs," Master's thesis, Department of Computer Science, University of Toronto, 2000.

[4] M. Molloy and B. Reed, "A bound on the strong chromatic index of a graph," *Journal on Combinatorial Theory, Series B*, vol. 69, no. 2, pp. 103–109, Mar. 1997.

[5] I. Slama, B. Jouaber, and D. Zeghlache, "A free collision and ditributed slot assignment algorithm for wireless sensor networks," in *Proceedings of IEEE Global Telecommunications Conference (GLOBECOM'08)*, Dec. 2008, pp. 1–6.

[6] S. Gandham, M. Dawande, and R. Prakash, "Link scheduling in sensor networks: Distributed edge coloring revisited," in *Proceedings of IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Socities (INFOCOM'05)*, vol. 4, Mar. 2005, pp. 2492–2501.

[7] C. L. Barrett, G. Istrate, V. S. A. Kumar, M. V. Marathe, S. Thite, and S. Thulasidasan, "Strong edge coloring for channel assignment in wireless radio networks," in *Proceedings of Fourth Annual IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOMW'06)*, Mar. 2006, pp. 105–110.

[8] M. Luby, "A simple parallel algorithm for the maximal independent set problem," in *Proceedings of the 17th Annual ACM Symposium on Theory of Computing (STOC'85)*, Dec. 1985, pp. 1–10.

[9] N. A. Lynch, *Distributed Algorithms*. Morgan Kaufmann, 1996.

[10] P. Flajolet, D. Gardy, and L. Thimonier, "Birthday paradox, coupon collectors, caching algorithms and self-organizing search," *Discrete Applied Mathematics*, vol. 39, no. 3, pp. 207–229, Nov. 1992.