國 立 成 功 大 學

電腦與通信工程研究所

碩 士 論 文

在無線感測器網路中的低延遲鏈結排程機制

Low-Latency Link Scheduling in Wireless Sensor Networks

研 究 生: 王 超　　Student: Chao Wang
指導教授: 斯國峰　　Advisor: Kuo-Feng Ssu

Institute of Computer and Communication Engineering
National Cheng Kung University

Thesis for Master of Science
June 2010

中 華 民 國 九 十 九 年 六 月

Low-Latency Link Scheduling in Wireless Sensor Networks

Chao Wang
Institute of Computer and Communication Engineering
National Cheng Kung University

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 the minimum graph coloring. However, such approach is not effective towards low-latency transmissions due to the hidden square phenomenon. The graph coloring approach paints each node with single color, but in fact each node can have multiple colors and still satisify the coloring constraint. When applied the coloring method in wireless scheduling, these additional color assignments, now associated with the time slot assignments, can reduce the data buffering delay.

This thesis studies the hidden square phenomenon, and accordingly proposes methods to identify all hidden squares. With these methods, two link scheduling schemes are proposed. The first scheme is named DCLS, a distributed collision-free low-latency link scheduling. The scheme considers the network snapshot at each time slot, and determines a set of collision-free links 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. The second scheme, named GSA, is a greedy slot assignment scheme. GSA is capable of assigning all feasible time

slots to each node in the network, and the simulation results have shown that it is possible to reduce the delay of conventional coloring approach in half.

The second part of this thesis considers the problem of the upper bound of the maximum degree on inserting vertices in unit disk graphs. This study first introduces the underlying structural properties of UDG insertions, and then proves a least upper bound as the function of the previous maximum degree. Further, the results of this study lead to a surprising necessary condition in the presence of the upper bound, showing that the network should have at least six disconnected components. Finally, the case study gives examples in wireless networks. Applications of the proposed problem include, but not limit to, the scheduling protocols in wireless sensor networks, mobile ad hoc networks, and wireless networks with mobile stations.

Acknowledgments

I wish to express my sincere thanks to my advisor, Professor Kuo-Feng Ssu, for his invaluable direction and assistance during my years at the National Cheng Kung University. Thanks are due to Professors Hao-Hua Chu, Hewijin Christine Jiau, and Chia-Ho Ou for their contributions to the research and for serving on my dissertation committee. I also acknowledge Professor Ssu and Jiau for their inspiring lectures in, respectively, Introduction to Computer Science, and System Analysis and Design Theory.

I am indebted to my colleagues in Dependable Computing Laboratory, including Wei-Tong Wang, Chun-Hao Yang, Yu-Yuan Lin, Wei-Cheng Chu, Meng-Hsiu Kuo, Meng-Chieh Lin, Ting-Hsu Wei, Wei-Chen Song, Yen-Tsung Chung, Hsiu Yu, Chiao-Wei Chih, Hsun-Jui Chiang, and Jun-Ye Liao, for both their sugesstions in my research and company in daily lives. Special thanks should go to Chun-Hao Yang, Meng-Hsiu Kuo, Yen-Tsung Chung, and my flatmate Chi-Sen Chiu for the lively discussions on various research topics. Also, to Grace Chan I owe a sincere debt of gratitude. She made numerous valuable sugesstions both on my English writing and oral presentation.

Finally, I would like to thank my family and all my good friends, who have been a constant source of encouragement and understanding. Last, but not least, I wish to thank Yu-Lin Huang, for making me concentrate on my research, and for countless memories we shared. This thesis is dedicated to her.

Table of Contents

Chapter

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Scheduling in Wireless Sensor Networks

Wireless sensor networks have been a popular research topic, owing to their rich applications such as event detecting, target tracking and other scientific purposes. Each sensor node is powered by limited battery-supplied energy. Due to certain environmental conditions, it may be infeasible to replace or recharge the battery, and sensor nodes are assumed to be disposed after they are out of battery. To prolong the network lifetime, energy efficiency should be taken into great considerations. Besides energy efficiency, the latency of transmission is also important in many scenarios such as the military or life-saving operations.

Packet-collision-avoidance is a primary issue towards energy efficiency in wireless sensor networks [1]. 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 at the same time. For example, Figure 1.1a shows that node B is receiving packets from both node A and node C. Secondary interference

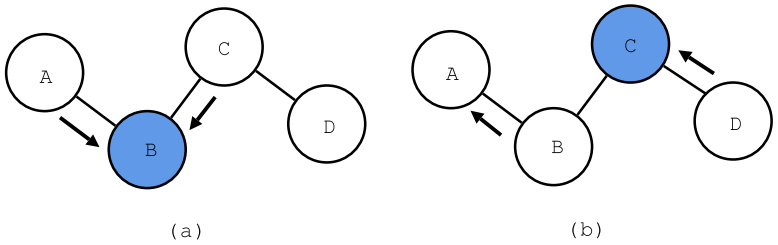

Figure 1.1: (a) The primary interference and (b) the secondary interference.

occurs when a receiver is within the transmission range of other transmitters. For instance, Figure 1.1b shows that node C is receiving a packet from node D, while node A is, simutaneously, receiving a pack from node B. Node C is interfered by the signal from node B, since it is within the transmission range of B. When packet collision occurs, this packet has to be dropped and retransmitted, which not only causes throughput degradation, but also wastes energy.

Time Division Multiple Access (TDMA) is one of the possible methods of solving the collision problem. In TDMA, time is equally divided into intervals called *frames*, and each frame is further divided into *time slots*. Packet collision can be eliminated by making each adjacent node to transmit only during different time slots. Proper scheduling on time slots is necessary both to prevent packet collision as well as to maintain the network connectivity and reasonable throughputs. The collision-free TDMA scheduling protocols can be classified into two categories, including *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 can share the same time slots only if they are neither adjacent nor connected by a third link [2]. With the above constraints, the generated schedule can guarantee collision-free.

Besides preventing packet collisions in order to save energy, IEEE 802.11 Power-Saving mode [3] includes a mechanism that allows nodes to enter into sleep mode when no transmission occurs during the time slot. Given a transmitter, the broadcast scheduling requires all of its neighbor nodes awake to receive packets. Energy is wasted when a node receives packets which are not intended for it. Link scheduling, on ther other hand, assigns the time slot for each pair of transmission, and the node may wake up only in these assigned slots. As a consequence, link scheduling is favored when the unicast or multicast transmission is of concern.

There are strong bonds between the TDMA scheduling and the coloring problem in graph theory [4]. The broadcast scheduling relates to the vertex coloring [5], and the link scheduling could be modeled into the edge coloring. Further, the latter problem could be associated with *valid edge-coloring* for directed links [6] and *strong edge-coloring* for undirected links [7]. In the vertex/edge coloring, each color is mapped to a different time slot, and the schedule length equals the number of colors. The goal of coloring is to properly paint all nodes/edges with minimum number of colors, while meeting the broadcast/link scheduling constraint. Though the concept of graph coloring correctly models the scheduling constraint, but such simple reduction of TDMA scheduling neglects some opportunities of time division. According to our observation, the graph coloring approach does not effectively utilize the network bandwidths due to a phenomenon termed *hidden squares* in this thesis.

This thesis firstly introduces the hidden square phenomenon, and gives a sufficient and necessary condition for it in the link scheduling. Accordingly, a $O(1)$ algorithm is proposed to identify this phenomenon. Two link scheduling schemes, DCLS and GSA, are

presented. DCLS treats the link scheduling problem in an aspect different from the strong edge-coloring. In the conventional strong edge-coloring, the assignments are performed once for all time slots on a given network topology; DCLS considers the network topology at each time slot and determines pairs of collision-free link on each network snapshot. As a result, the transmission delay can be improved, as each node end up with multiple time slots for transmission in one schedule length. DCLS partially exploits the hidden square phenomenon, and it needs to determine the schedule length in advance. Another proposed scheme, termed GSA, integrates the algorithm to identify all hidden squares, and therefore it can assign all feasible time slots to each node. Unlike DCLS, GSA does not require given the schedule length in advance. The analysis and simulation shows that both DCLS and GSA can achieve lower transmission delay than the conventional graph coloring methods.

For the related research on broadcast scheduling, refer to [1, 5, 8–10] and [2]. For the link scheduling problem, Gandham et al. proposed an 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. ρ denotes the time spent on counting the degree of each node in the network. Ramanathan and Lloyd had developed algorithms for link scheduling on both restricted and arbitrary graphs [2]. The schedule length is $O(\theta^2)$

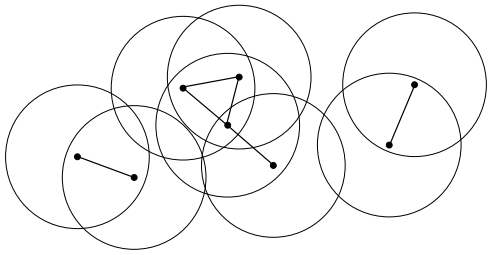

Figure 1.2: A unit disk graph.

times the optimum, where θ is the minimum number of planar graphs decomposed from the network. For the related work on the power-saving protocols, refer to [11–13].

1.2 Unit Disk Graph

In a graph model, each vertex represents an object, and an edge connects two vertices if they hold a certain relation. The graph model is very useful for describing wireless networks [2,6,7,10,14–18]. The unit disk graph (UDG) [19,20] models wireless networks with the assumption that each station has the same communication range. There are three common models of UDG: the *proximity, intersection,* and *containment* models [19]. This thesis exclusively uses the proximity model without loss of generality, since the transformation between three models needs only linear time. Consider a set of equal-sized disks in the Euclidean plane. Each vertex is located at the center of a disk and the radius of each disk is set to r. An edge connects two vertices v, u if and only if their Euclidean distance is within r; the resulting graph is called r-UDG. Figure 1.2 illstrates an example UDG. The terms v and u are *adjacent* if there is an edge between them. The *degree* of a vertex, denoted as δ, is the number of its adjacent vertices. The term $d(v)$ is the set of all adjacent vertices of vertex v, and $\delta_v = |d(v)|$. The maximum degree of the graph is denoted by Δ, and it is the maximum value of δ among all vertices in the graph.

The parameter Δ plays an important role in many applications of the UDG model. In wireless networks, for example, the space complexity in the time division multiple access (TDMA) scheduling algorithm depends on Δ. The degree of a node also directly affects the efficiency of message exchange [2,6,7,10]. Two typical variations of wireless networks involve the alternation of Δ. First, wireless ad hoc networks may employ additional sensors to either extend the network coverage area or replace faulty sensors. Second, some applications in wireless sensor networks have mobile stations that cruise the network to collect data. The UDG model views these mobile stations as inserted vertices. Thus, a knowledge of the maximum degree after vertex insertion (Δ' hereafter) is necessary. If designers are able to estimate the value of Δ', they can conduct more adaptive systems. Furthermore, since the system is working under current Δ, it is advantageous to know the relationship between Δ and Δ'. This raises a critical question: *what is the relationship between Δ and Δ', or what is the growing rate of the maximum degree after inserting additional vertices into UDG?*

This thesis answers the above question with a theoretical upper bound on Δ', represented as the function of Δ. Let the function $f_n(\Delta)$ represent the upper bound of Δ' after inserting n vertices into UDG. This study proves that $f_1(\Delta) = 5(\Delta + 1)$, and this is essentially the least upper bound for general UDGs. Further, when the equation holds, there should be more than six disconnected components in the graph. For connected UDGs, the least upper bound equals $f_1(\Delta) = 5\Delta + 1$. Finally, $f_n(\Delta) = f_1(\Delta) + s_n - 1$, where s_n is the size of the largest connected component in the set of additional vertices. The simulation evaluates the tightness of the upper bound with various graph settings.

Chapter 2

System Model and Problem Formulation

This thesis assumes all nodes are equipped with omnidirectional antenna with transmission range r. 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 it 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 one-hop neighbor nodes. A message could be sent to the specific node by wrapping the receiver's ID within the packet. Before the transmission schedule is arranged, the transmission is not free from interference. To reduce the interference rate, each node backs off a bounded random time before sending message. No back-off time is needed after the scheduling.

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. The node can transmit data only during owned time slots.

The unit disk graph model is assumed. 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.* $|T|$ *denotes 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*

$$T_v \cap T_i \cap T_j = \phi. \tag{2.1}$$

for all v *in* V*;* $i, j \in d(v)$ *and* $i \neq j$*.*

The above equation states 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 Figure 2.1, 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 $T_E = \{1, 4\}$.

It should be pointed out that the link scheduling criteria is, in fact, a special type of quorum systems [21–23]:

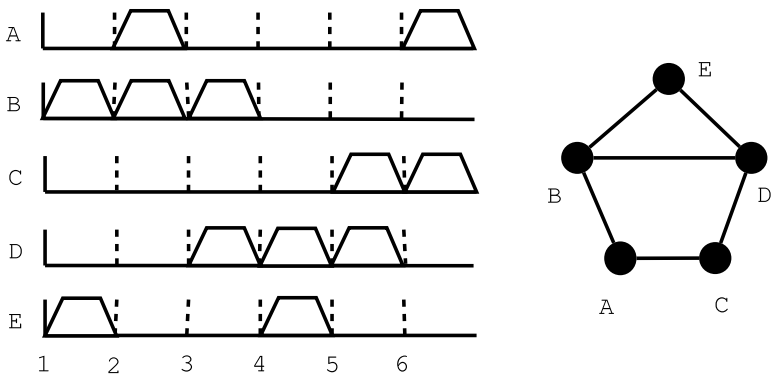

Figure 2.1: An example of a collision-free link schedule.

Definition 3 (Coterie [23]). *Let $U = \{P_0, P1, ..., P_{n-1}\}$ be a set. A set C of subsets of U is a coterie under U if and only if the following conditions are satisfied.*

1. *Nonemptiness. For each $Q \in C$, Q is not empty, and $Q \subseteq U$.*

2. *Intersection property. For any Q, $Q' \in C$, $Q \cap Q'$ is not empty.*

3. *Minimality. For any Q, $Q' \in C$, Q is not a proper subset of Q'.*

An element of C is called a quorum.

Definition 4 (Data Buffering Delay). *Data buffering delay is the number of time slots for which a node has to wait before the next packet transmission to the same receiver takes place.*

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

$$D_{v,u} = \frac{|T|}{|T_v \cap T_u|} \tag{2.2}$$

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

$$D_v = \frac{\sum_{u \in d(v)} D_{v,u}}{\delta_v} \tag{2.3}$$

$$D_{net} = \frac{\sum_{v \in V} D_v)}{|V|}. \tag{2.4}$$

In the conventional strong edge-coloring 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:

$$s_{\chi'}(G) \leq 1.998\Delta^2. \tag{2.5}$$

For a tighter upper bound, there is a still widely open conjecture [24]: for any graph G of maximum degree Δ, $s_{\chi'}(G) \leq f(\Delta)$, where $f(\Delta)$ is defined as follows:

$$f(\Delta) = \begin{cases} \frac{5}{4}\Delta^2 & \text{if } \Delta \text{ is even.} \\ \frac{5}{4}\Delta^2 - \frac{1}{2}\Delta + \frac{1}{4} & \text{if } \Delta \text{ is odd.} \end{cases} \tag{2.6}$$

Chapter 3

The Hidden Square Phenomenon

3.1 Presence of Hidden Squares

Figure 3.1 illstrates the phenomenon. Consider a map with 5×5 squares, the objective is to paint each adjacent square with a different color. Figure 3.1 shows a feasible solution using 4 colors. By examining each color individually, it is clear that there are some squares (marked with X) nonadjacent to any square of the given color. A graph coloring method cannot find these squares, because they are already painted by other colors. The squares with this characteristic are termed *hidden squares* of a given color. Figure 3.2 shows all hidden squares in this example.

Generally, hidden squares are likely to appear in the graph coloring which uses many different colors; given the same graph, more colors involved in the graph coloring indicates that more colors are neglected in the view of a single vertex, because each vertex is painted by the single color. In the strong edge-coloring, the typical model for the link scheduling, it is NP-hard to determine the minimum number of colors [24], and a general upper bound of this number is known only for sufficiently large Δ [25]. But, even in the graphs

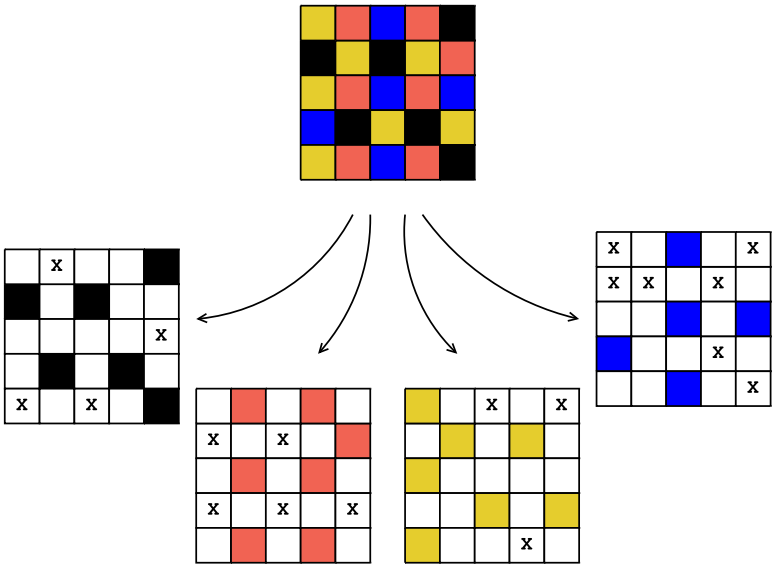

Figure 3.1: The upper part shows a result of graph coloring method. For each color, there are hidden colorable squares (marked with X).

painted by the minimum number of colors, the phenomenon is sometimes inevitable. For example, consider applying strong edge-coloring to the graph shown in Figure 3.3. The digit on each edge represents the assigned type of color. To use minimum number of colors, six colors in this case, the rightmost edge must choose either type 1, 2, or 4. However, choosing one of them leads to the hidden squares of the other two colors.

The hidden square phenomenon causes an undesirable effect in wireless sensor networks. After applying the graph coloring method to schedule the networks, certain time slots will be still unassigned to the nodes, even if such assignments make no collision. With these assignments, corresponding nodes can own more time slots, and thus less transmission delay. That is, by exploiting the hidden squares, we can reduce the time that each node spent on waiting for next available time slot.

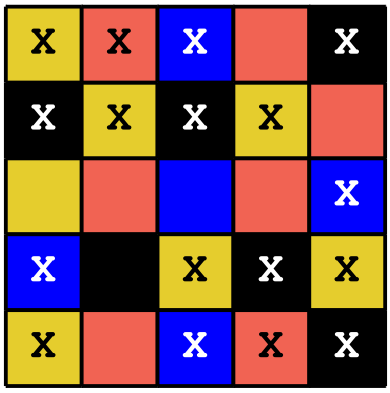

Figure 3.2: All squares which can be painted by more colors.

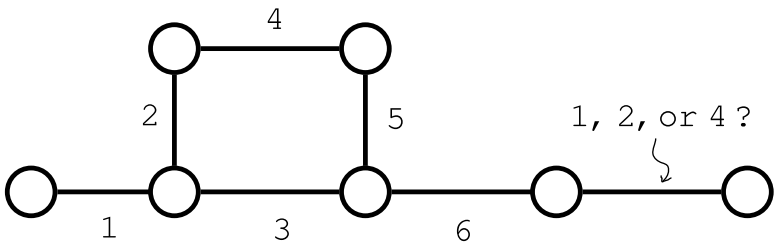

Figure 3.3: The hidden-square phenomenon arises after painting the rightmost edge.

3.2 Hidden Square Identification

Recall from Figure 3.1, the hidden square phenomenon occurs when some squares can be painted by multiple colors. From the view of each square, it is a hidden square if and only if all adjacent squares and it itself are not painted by certain colors. This leads to the following algorithm: *(Algorithm 1) For each square, check all its adjacent squares to see if they have exausted all colors. If not, and that this square has the color different from the remaining colors, then it itself is a hidden square of those unused colors. Repeat the above procedure through all squares.*

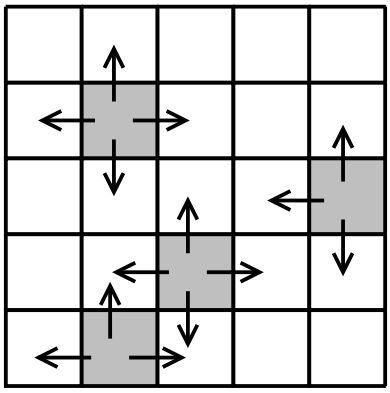

Figure 3.4: The concept of identifying the hidden squares.

Figure 3.4 shows the concept of Algorithm 1. This algorithm has linear time complexity $O(n)$. Further, it can be applied in parallel, since each square only has to check its adjacent squares. The parallel version has $O(1)$ in the running time.

Similarily, in the link scheduling problem, each node can locally check if it suffers from the hidden square phenomenon. Define the term *hidden slot* of a given node to be the collision-free time slot neglected by that node. For the hidden slot of link $(v,u) \in E$, it must both belong to the sets $T-(T_v \cup \bigcup_{i\in d(v)} T_i) \equiv T-\bigcup_{i\in d(v)} T_i$ and $T-(T_u \cup \bigcup_{j\in d(u)} T_j) \equiv T-\bigcup_{j\in d(u)} T_j$. Let $H_{(v,u)}$ be the set containing all hidden slots of link $(v,u) \in E$, then

$$\begin{aligned} H_{(v,u)} &\equiv (T - \bigcup_{i\in d(v)} T_i) \cap (T - \bigcup_{j\in d(u)} T_j) \\ &\equiv T - ((\bigcup_{i\in d(v)} T_i) \cup (\bigcup_{j\in d(u)} T_j)). \end{aligned} \tag{3.1}$$

The following set H_v consists of all hidden slots of node $v \in V$:

$$\begin{aligned} H_v &\equiv \bigcup_{u\in d(v)} H_{(v,u)} \\ &\equiv T - \bigcap_{u\in d(v)} ((\bigcup_{i\in d(v)} T_i) \cup (\bigcup_{j\in d(u)} T_j)) \\ &\equiv T - ((\bigcup_{i\in d(v)} T_i) \cup (\bigcap_{u\in d(v)} (\bigcup_{j\in d(u)} T_j))). \end{aligned} \tag{3.2}$$

Figure 3.5 illstrates the corresponding nodes in $\bigcup_{i\in d(v)} T_i$ and $\bigcap_{u\in d(v)}(\bigcup_{j\in d(u)} T_j)$.

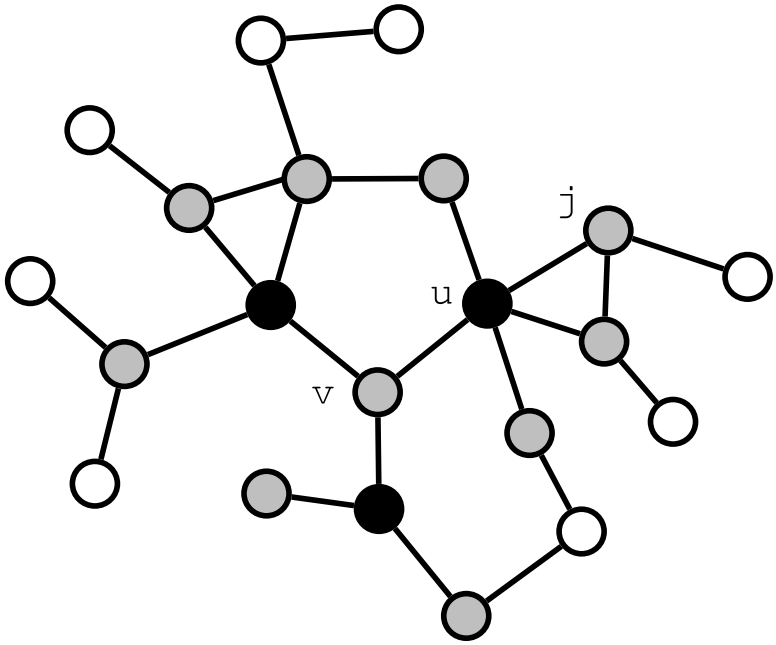

Figure 3.5: The corresponding nodes in $\bigcup_{i \in d(v)} T_i$ and $\bigcap_{u \in d(v)} (\bigcup_{j \in d(u)} T_j)$ are marked black and gray, respectively.

Theorem 1. *The node $v \in G$ is free from the hidden square phenomenon if and only if*

$$H_v \equiv \phi. \tag{3.3}$$

Theorem 1 directly leads to the algorithm to identify all hidden slots. *(Algorithm 2) Each node $v \in V$ exchanges the slot information with its one-hop neighbor nodes twice, so that it has all its two-hops slot information. Compute H_v. If H_v is an empty set, then node v has no hidden slot; otherwise, H_v consists of all hidden slots of v.*

Clearly, Algorithm 2 has $O(1)$ in the running time complexity.

Chapter 4

Low-Latency Link Scheduling

Figure 3.1 motivates a simple strategy to assign hidden slots. For example, if there are 4 colors, then we use them on four different map snapshots, respectively. As shown in Figure 4.1, by concatenateing these snapshots, each square may have mutiple colors. Since on each snapshot the procedure obeys the coloring constraint, the resulting map should satisify the constraint as well. This concept leads to the following distributed link scheduling scheme.

4.1 The Distributed Approach

The scheme first determines the schedule length of network (the number of time slots), then selects pairs of collision-free link over each network snapshot; a snapshot is the network topology at the given time slot. For instance, Figure 4.2 shows two possible sets of collision-free links in a snapshot. In each snapshot every node executes three subroutines: **Initialization**, **Message_passing**, and **State_transition**. The pseudocode is given in Figure 4.6 and the flow diagram is shown in Figure 4.3. The *state* variable is

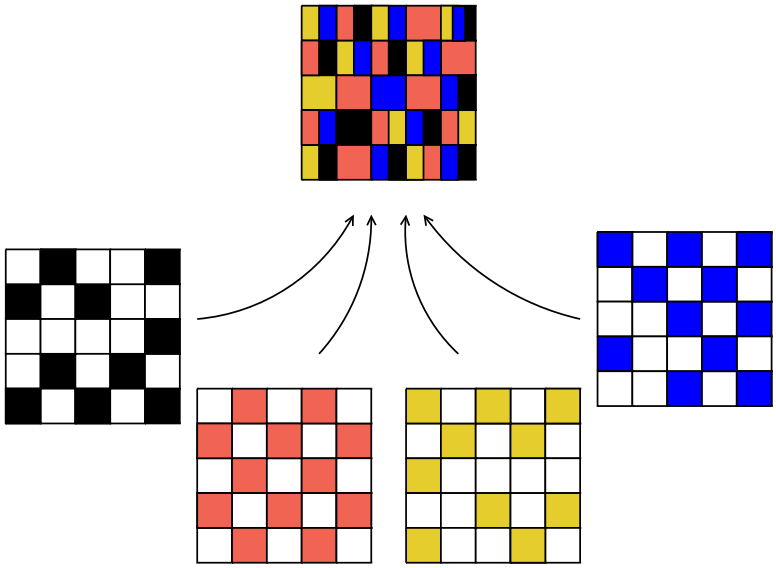

Figure 4.1: The concept of DCLS.

added by 1 at the end of **State_transition**, except it is assigned to a explicit value. All nodes are synchronized at the end of each snapshot. As a 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 link selection is shown in Figure 4.4.

4.1.1 Collision-free link selection

In the first step of the selection phase, all nodes are marked in *white*. DCLS then generates an independent node set based on the method proposed by Luby [26]. Each node generates a unique random number and compares it with all its neighbor nodes. The node with the smallest number is included in the independent set. Repeat the above subroutine $O(1)$ rounds. 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

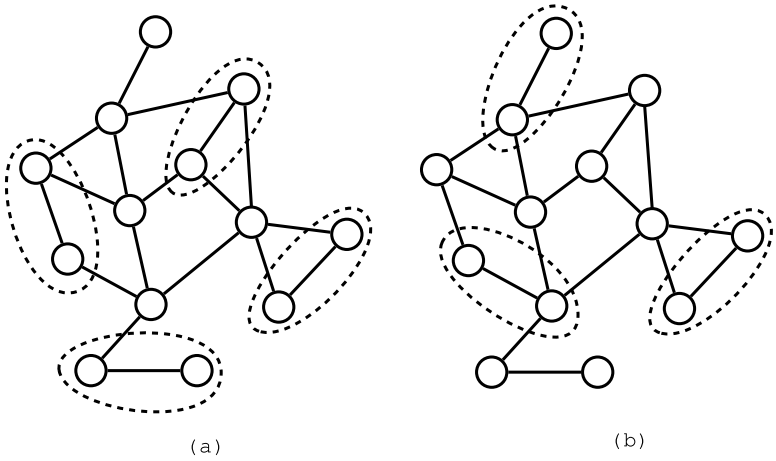

Figure 4.2: Two possible sets of collision-free links on a snapshot. Each link is circled by dotted line.

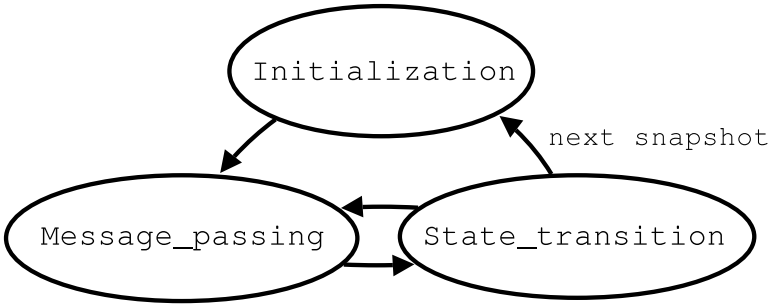

Figure 4.3: The flow diagram of the sub-routines.

maximal independent node set, because it will cost $O(\log^2 |V|)$ rounds to find a maximal independent set [26].

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 to form a collision-free link. If there is no available neighbor node, that black node reverses itself to *white*.

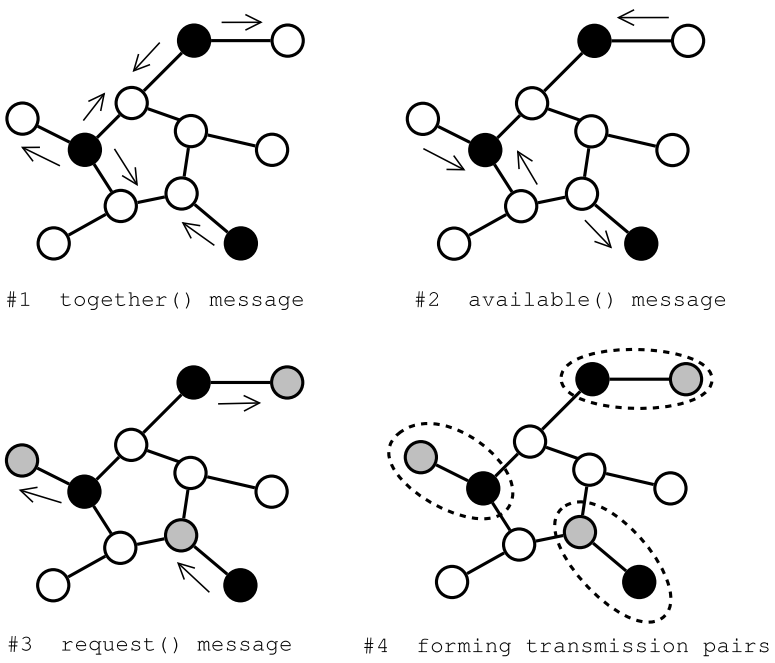

Figure 4.4: The procedure on selecting collision-free links.

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 collision-free links with their requester nodes. The other gray nodes send *reject*() messages back to their requesters, forcing them to request other available nodes. The above scenario is shown in Figure 4.5a. To avoid a black node sending *request*() circularly to some of its gray neighbor nodes (Figure 4.5b), 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 collision-free link. Hence, a set of collision-free links on the snapshot are

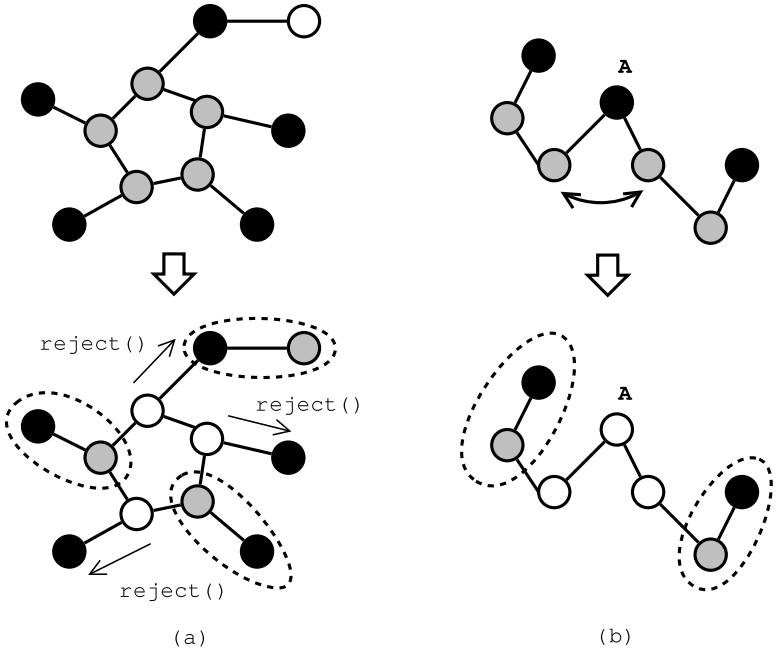

Figure 4.5: (a) DCLS generates an independent set among adjacent gray nodes, and the nodes in the set are able to form collision-free links. The other gray nodes turn *white* and send *reject*() back to their requesters, force which to request other available nodes. (b) Node A circularly requests two neighbors. It will cease and turn *white* after receiving *reject*() α times; other nodes then could form collision-free links.

constructed, and all nodes belonging to these links *own* the time slot. DCLS then repeats the selection phase for the next snapshot.

4.1.2 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 of forming collision-free links. Under such a 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 are then ranged between 1001 to 2000, and therefore all nodes once again have the same probabilities to be included in the independent set.

It is the right time to revisit the claim 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 collision-free link 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 collision-free link 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 $|T_v \cap T_u| \geq 2$.

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**

Figure 4.6: Pseudocode of Collision-Free Link Selection

4.1.3 Analysis

Theorem 2. *DCLS satisfies the link scheduling criterion.*

Proof. All snapshots are orthogonal to each other, so it is sufficient to consider an arbitrary snapshot and prove that the criterion holds. The proof is demonstrated by proving two arguments: 1. there are no overlapped pairs of transmission, and 2. no adjacent link can be selected.

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 collision-free link, and the overlapped pairs of transmission 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 pairs of transmission can be adjacent. With arguments 1 and 2, DCLS is proved to satisfy the link scheduling criterion for a static network. □

Theorem 3. *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 [27]. □

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

$$\mathtt{Pr}(\mathcal{E}(v)) \geq \frac{\epsilon}{\delta_v + 1}. \tag{4.1}$$

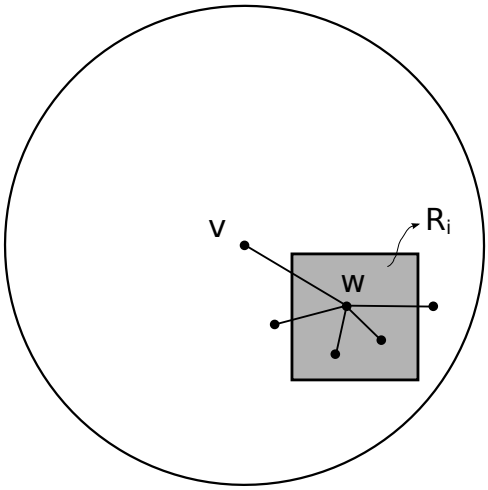

Figure 4.7: Illstration of Lemma 1.

Proof. Because of the independent trails,

$$\texttt{Pr}(\mathcal{E}(v)) = \frac{1}{\delta_v + 1} \prod_{w \in d(v)} (1 - \frac{1}{\delta_w + 1}). \tag{4.2}$$

Partition the disk D_v into a constant number of unit regions $R_1, ..., R_s$, such that these regions are mutually exclusive. Let $|R_i|$ be the number of nodes in R_i. Then for each R_i, we have $\delta_w \geq |R_i|$ for each w in R_i. Figure 4.7 illstrates this property. Hence,

$$\begin{aligned} \texttt{Pr}(\mathcal{E}(v)) &\geq \frac{1}{\delta_v + 1} \cdot \prod_{i=1}^{s} \prod_{w \in R_i} (1 - \frac{1}{\delta_w + 1}) \\ &\geq \frac{1}{\delta_v + 1} \cdot \prod_{i=1}^{s} (1 - \frac{1}{|R_i|})^{|R_i|} \\ &\geq \frac{(\epsilon' e)^s}{\delta_v + 1} = \frac{\epsilon}{\delta_v + 1}\,. \end{aligned} \tag{4.3}$$

□

Theorem 4. *The delay D_{net} obtained by DCLS is*

$$O((\Delta + 1)(\ln \Delta + O(1))). \tag{4.4}$$

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:

$$E[M] = 1/\mathtt{Pr}(\mathcal{E}(v)) \leq (\delta_v + 1)/\epsilon \tag{4.5}$$

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

$$\begin{aligned} Pr(M > c \cdot E[M]) &= (1 - Pr(\mathcal{E}(v)))^{c \cdot E[M]} \\ &= (1 - 1/E[M])^{c \cdot E[M]} \leq e^{-c}. \end{aligned} \tag{4.6}$$

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 [28]. Assume a uniform probability distribution, and the expectation of M' is

$$E[M'] = \delta_v \cdot H_{\delta_v} \approx \delta_v \cdot (\ln \delta_v + O(1)) \tag{4.7}$$

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

Since the computations of DCLS are performed in a distributed manner, the schedule length $|T|$ is $O(E[M] \cdot E[M'])$. From Eq. (2.3),

$$D_v = \frac{|T|}{\delta_v} \cdot \textstyle\sum_{u \in d(v)} \dfrac{1}{|T_v \cap T_u|} \tag{4.8}$$

and from Eq. (2.4),

$$\begin{aligned} D_{net} &= (\textstyle\sum_{v \in V} D_v)/|V| \\ &= O(E[M] \cdot E[M']/\Delta) \\ &= O((\Delta + 1)(\ln \Delta + O(1))). \end{aligned} \tag{4.9}$$

□

Comparing Eq. (4.4) with Eq. (2.5), 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:

$$\rho_v = \frac{\sum_{u \in d(v)} |T_v \cap T_u|}{|T|} . \tag{4.10}$$

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:

$$|T_v \cap T_u|_{avg} = \frac{\rho_v \cdot |T|}{\delta_v} . \tag{4.11}$$

4.1.4 Performance Evaluation

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.

4.1.4.1 Performance on data buffering delay

The simulation statistics are given in Table 4.1. For the data buffering delay, the performance of DCLS is shown in column $\langle 3 \rangle$. The standard deviation of DCLS is given in column $\langle 7 \rangle$. The result shows that the delay value is stable for various network densities.

As mentioned, 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. Figure 4.8

Table 4.1: Simulation statistics

	⟨1⟩	⟨2⟩	⟨3⟩	⟨4⟩	⟨5⟩	⟨6⟩
Δ	Nodes	Delay			$\frac{\langle 2\rangle-\langle 3\rangle}{\langle 2\rangle}$	$\frac{\langle 3\rangle}{\langle 4\rangle}$
		1. GC	2. DCLS	3. $(\Delta+1)(\ln\Delta)$		
6	20	71.93	14.62	12.54	79%	1.17
8	30	127.87	25.35	18.71	80%	1.35
10	40	199.8	31.04	25.33	84%	1.23
12	50	287.71	52.18	32.3	82%	1.62
14	60	391.61	66.61	39.58	83%	1.68
16	70	511.49	91.37	47.13	82%	1.94
18	80	647.35	107.21	54.91	83%	1.95

	⟨7⟩	⟨8⟩	⟨9⟩	⟨10⟩	⟨11⟩
Δ	Std. dev. of ⟨3⟩	Duty cycle	Snapshots	$\lvert T_v \cap T_u\rvert_{avg}$	$\frac{\langle 9\rangle}{2\cdot\langle 10\rangle}$
6	1.06	0.38	100	3.17	15.78
8	1.65	0.3	150	2.81	26.69
10	1.52	0.29	200	2.9	34.48
12	2.12	0.22	400	3.66	54.64
14	1.8	0.18	500	3.22	77.63
16	3.24	0.15	800	3.8	105.26
18	2.77	0.13	1000	3.61	138.5

shows the comparison of data buffering delay under various Δ, in which GC denotes the general upper bound of $s_{\chi'}(G)$ (Eq. (2.5)) and GC-conj represents the conjecture on $s_{\chi'}(G)$ (Eq. (2.6)); the value of DCLS is obtained by running the 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⟩.

4.1.4.2 Performance on power saving

DCLS could be optimized to save energy. Since a node can only transmit 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 4.1. For example, for the topology with $\Delta = 12$, a node has 78% of the time in sleep mode.

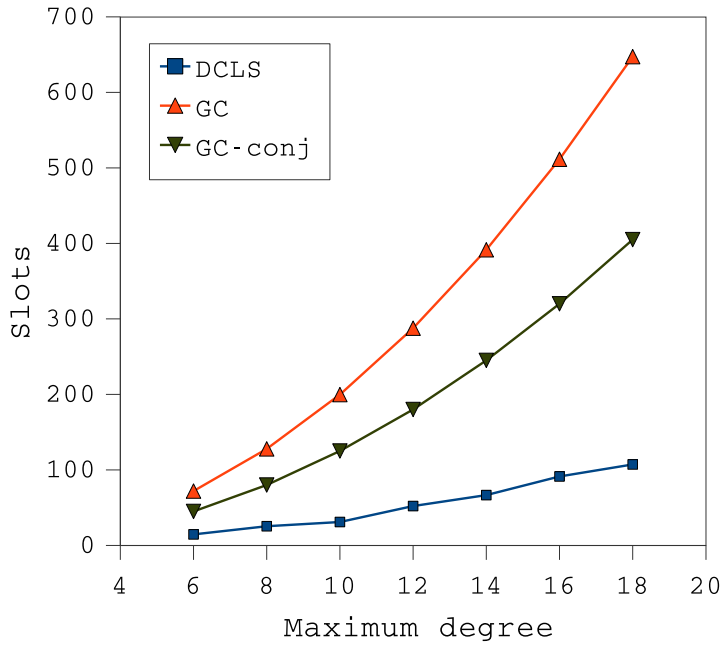

Figure 4.8: Data buffering delay.

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 also gets smaller.

4.1.4.3 Other characteristics

The computation complexity of data buffering delay of DCLS is $O((\Delta+1)(\ln\Delta+O(1)))$. Column $\langle 6\rangle$ in Table 4.1 gives the estimated coefficient of the highest order term in this asymptotic notation. The value is ranged from 1.17 to 1.95.

Eq. (4.11) can be computed with the simulation results, and the value is shown in column $\langle 10\rangle$ in Table 4.1. According to Eqs. (2.2), (2.3) and (2.4),the value of data

buffering delay can be approximately determined:

$$\begin{aligned} D_{net} &\approx \frac{|T|}{\Delta} \cdot \sum_{\Delta} \frac{1}{|T_v \cap T_u|_{avg}} \\ &\approx \frac{|T|}{|T_v \cap T_u|_{avg}} \end{aligned} \tag{4.12}$$

and the value is given in column $\langle 11 \rangle$ of Table 4.1. This value is close to the one obtained by simulation, as displayed in column $\langle 3 \rangle$. Hence, it suggests that the simulation result is in consistent with the analytical analysis.

4.2 The Centralized Method

The DCLS algorithm is the first scheme to deal with the hidden square phenomenon. It is a probabilistic algorithm, and it requires a predetermined schedule length. This section introduces GSA, a simple deterministic method which does not require a predetermined schedule length.

Figure 4.9 and Figure 4.10 gives the pseudocode of GSA. GSA performs in two steps. The first step is to obtain a feasible solution by the greedy coloring, and the second step is to assign the hidden slots. GSA begins at an arbitrary node s. The greedy coloring method associates each time slot with a color. For each of its neighbor v, paint s and v with the same color, and then put v into the queue. Each pair of (s, v) receives different color, which must be different from those adjacent nodes of s or v. After all neighbor nodes of s are colored, mark s as *saturated* and dequeue a node for the next iteration. Once all nodes in V are saturated, set the schedule length of all nodes by the maximum slot number. Now GSA has generated a feasible schedule.

$nbr.i$: the set of one-hop neighbor nodes of i.
$slotnum.i$: the schedule length of i.
$satrd.i$: the indicator of saturation of i.
T_i : the set of owned time slots of i.
s : the starting node.
Q : a queue.

```
GSA-1
1   call GreedyColoring(s)
2   t ← the maximum slot number.
3   for all v ∈ V do
4         slotnum.v ← t
5   end for
6   repeatedly call GreedyColoring() until
       there's no hidden slot in the network.

GSA-2
1   call GreedyColoring(s)
2   t ← the maximum slot number.
3   for all v ∈ V do
4         slotnum.v ← t
5   end for
6   call FillHQ()
```

Figure 4.9: Pseudocode of Greedy Slot Assignment

At the second step, assigning the hidden slots, there are two possible methods. For simplicity, one can repeat the greedy coloring procedure several times. Because we have determined the schedule length, we know how many additional time slots at most can be assigned to the given node. Those time slots are essenially the hidden slots, and therefore GSA repeats the coloring procedure to assign these slots. The pseudocode GSA-1 describes this algorithm. The alternative is to firstly apply Algorithm 3 to identify all hidden slots, and then sequencially assign these hidden slots. GSA-2 gives the pseudecode of this method.

Theorem 5. *GSA satisfies the link scheduling criterion.*

Proof. The first step of GSA iterates through each node in a breadth-first manner. Before the assignment of the time slot c, the algorithm guarantees that $c \cap T_i \equiv \phi$ for each i in

```
GreedyColoring(s)
1    Enqueue(Q, s)
2    while Q ≠ φ do
3       v ← Dequeue(Q)
4       for all u ∈ nbr.v do
5          if (satrd.v = FALSE) and (satrd.u = FALSE)
6             c ← the smallest time slot number
                  which is not in {T_nbr.v ∪ T_nbr.u}.
7             T_v ← T_v ∪ c
8             T_u ← T_u ∪ c
9             Enqueue(Q, u)
10         end if
11         satrd.v ← TRUE
12      end for
13   end while

FillHQ()
1    for all v ∈ V do
2       for all hidden slots of v do
3          c ← the hidden slot number.
4          u ← the corresponding node.
5          T_v ← T_v ∪ c
6          T_u ← T_u ∪ c
7          update the slot information of
             those nodes in nbr.v ∪ nbr.u.
8       end for
9    end for
```

Figure 4.10: The Subroutines of GSA-1 and GSA-2

$d(v)$, and $T_i \cap T_j \equiv \phi$ for each i and j in $d(v)$. Therefore, $c \cap T_i \cap T_j \equiv \phi$ for each slot assignment. After node v is saturated, these assignments lead to $T_v \cap T_i \cap T_j \equiv \phi$, which is essentially the link scheduling crietrion. The `FillHQ()` subroutine assigns all hidden slots. By definition of the hidden slot, the assignment satisifies the link scheduling criterion. And after each assignment on node v, GSA updates the slot information of all nodes in $d(v)$, where the nodes then recompute their hidden slots. The above arguments have proved that GSA satisifies the link scheduling criterion. □

Since each node will be enqueued and dequeued once, the total time devoted to queue operation is $O(n)$. Because GSA scans the adjacency list of each dequeued node, the total

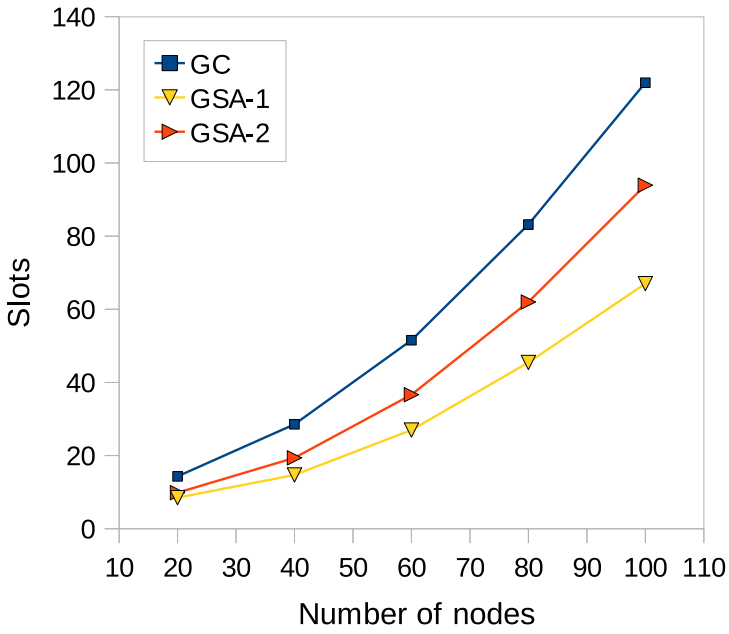

Figure 4.11: The performance of the data buffering delay.

iteration costs $O(nd)$, where d is the maximum degree of the network. The remaining part of GSA takes $O(n)$. So the total running time of GSA is $O(nd)$.

4.2.1 Performance Evaluation

The simulation setups a 250 $\times$ 250 meter-square area, and randomly deploys 20$\sim$100 static nodes. The whole network is connected. Each node has the transmission range setup to 50 meters.

Figure 4.11 shows the data buffering delay obtained by the conventional graph coloring (GC), GSA-1, and GSA-2. Each value is obtained from the average of 200 runs. It is clear that both GSA-1 and GSA-2 outperform GC in the data buffering delay, and the delay of GSA-1 is about half of that of GC. The delay obtained by GSA-2 is higher than that obtained by GSA-1. This is because GSA-2 assigns all feasible slots to a link in

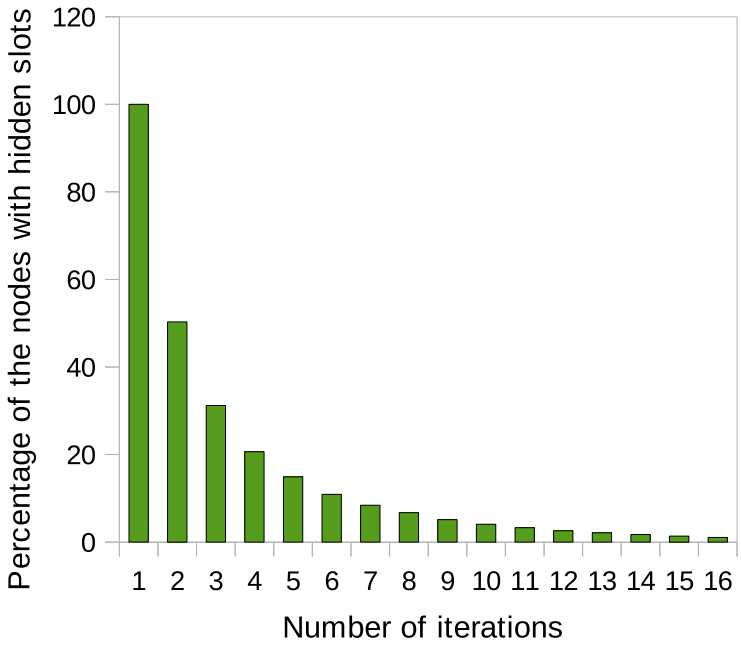

Figure 4.12: The percentage of the nodes with hidden slots after each iteration.

the single run, while GSA-1 assigns single time slot to one link in each of its iterations. Therefore GSA-1 should generate a more balanced slot assignment, and in average the delay will be smaller.

Given 40 nodes in the network, Figure 4.12 shows the percentage of the nodes with hidden slots after a certain number of iterations. Each value is obtained from the average of 500 runs. It can be seen that, in almost every case, all nodes will still have hidden slots after the first iteration. Because of the space limitation, the value of the 16-th iteration represents the summation of the percentage of itself and all subsequent iterations. GSA-1 terminates after assigning all hidden slots. Figure 4.13 gives the probability of termination after each iteration, and Figure 4.14 shows the corresponding CDF function. Again, the value of the 32-th iteration is the summation of itself and all subsequent iterations. These figures suggest that GSA-1 requires 15 or more iterations to assign all hidden slots.

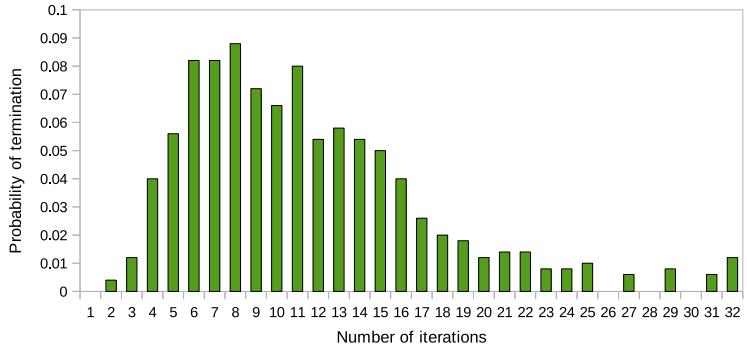

Figure 4.13: The probability of termination after each iteration.

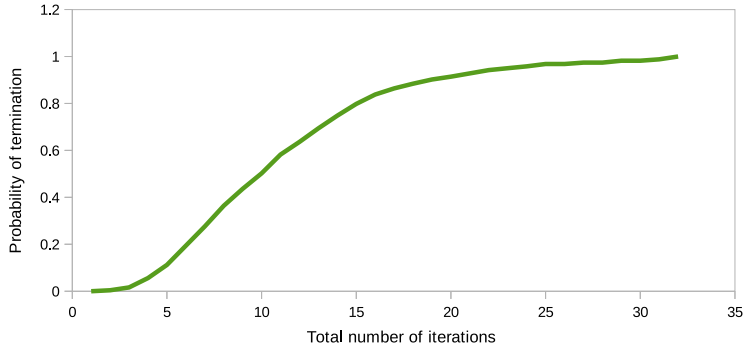

Figure 4.14: The cumulative distribution funtion of the probability of termination.

Finally, although the average delay of GSA-2 on the whole network is higher than GSA-1, it outperforms GSA-1 when applied to single node. The delay, in the average of 200 runs, of each of 40 nodes is shown in Figure 4.15. Each value of GSA-2 is obtained by uniquely executing `FillHQ`() to the given node. In this case, by GSA-2 all nodes have smaller delay compared with those by GSA-1 and GC. This result suggests that GSA-2 is favored when the delay performance of certain nodes are more important than the others. GC has equal delay for all nodes, due to the inherent principle of graph coloring.

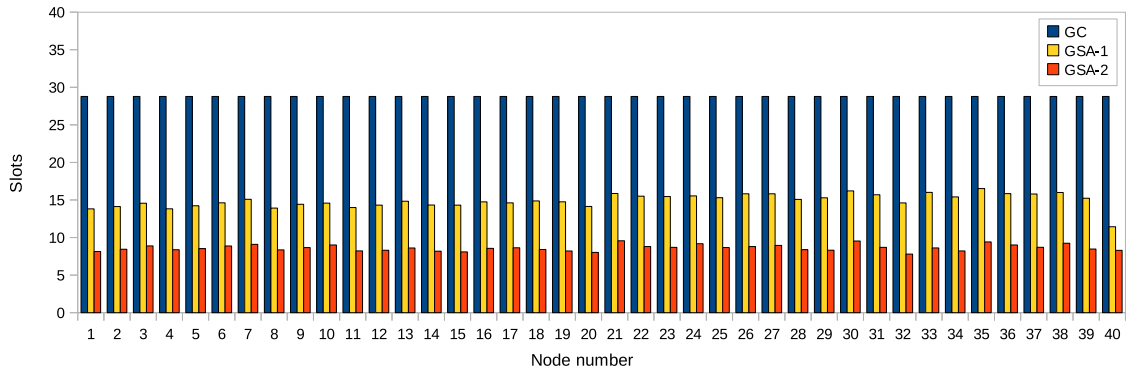

Figure 4.15: The performance of the data buffering delay on each node.

Chapter 5

The Least Upper Bound of Degree on Vertex Insertion

5.1 UDG Structure

Represent the r-UDG as $G = (V, E)$, which consists of a vertex set V and an edge set E. The term V is a finite set and E is a subset of $\{(u, v)|(u, v)$ is a pair in $V\}$. Given a vertex $v \in V$, denote the corresponding r-radius circular disk as $D_{(r,v)}$. For simplicity, this thesis uses D_v instead of $D_{(r,v)}$ and UDG instead of r-UDG if the context is clear. D_ϕ (namely $D_{(r,\phi)}$) indicates an arbitrary r-radius circular disk that does not require a vertex at the center point. $G' = (V', E')$ is the UDG after inserting vertices.

Lemma 2. *If* $\Delta' > \Delta + (|V'| - |V|)$, *then* $\Delta' = \{$*maximum of* $\delta_v \mid \forall v \in \{V' - V\}\}$.

Proof. Define a subset $C \subset V'$, where $C = \{x \cup y \mid \forall x \in \{V' - V\}$ and $\forall y \in d(x)\}$. A subset $\overline{C} = \{V' - C\}$ is the complement of C. Clearly, δ_u remains the same for all u in $\overline{C}$. If $\Delta' > \Delta$, then Δ' is equal to at least one of δ_v where $v \in C$. δ_u increments at

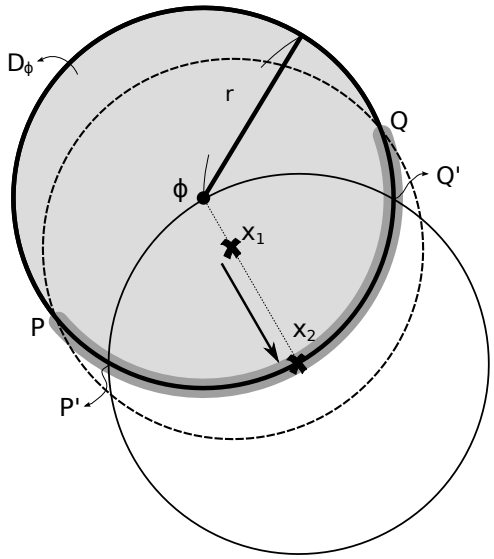

Figure 5.1: All vertices within D_ϕ can be moved to the circumference without increasing Δ.

most $(|V'| - |V|)$ for all u in $\{C - (V' - V)\}$. Thus, if $\Delta' > \Delta + (|V'| - |V|)$, then Δ' must equal the maximum of δ_v, where $v \in \{V' - V\}$. □

Lemma 3. *Let $G = (V, E)$ be a UDG where all vertices are within D_ϕ. With a fixed Δ, the maximum value of $|V|$ equals the maximum number of vertices on the circumference of D_ϕ.*

Proof. Let V_c be the set of vertices on the circumference of D_ϕ, and let V_i be the set of vertices inside D_ϕ such that $|V| = |V_c| + |V_i|$.

Firstly, each vertex in V_i can move to the circumference of D_ϕ without increasing Δ. Figure 5.1 demonstrates this concept. The vertex at position x_1 only influences the degree of vertices in V_c within the shaded arc $\widehat{PQ}$, and moving the vertex from x_1 to x_2 shrinks the arc. Due to $\widehat{PQ} > \widehat{P'Q'}$, the new maximum degree among V_c is smaller than or equal

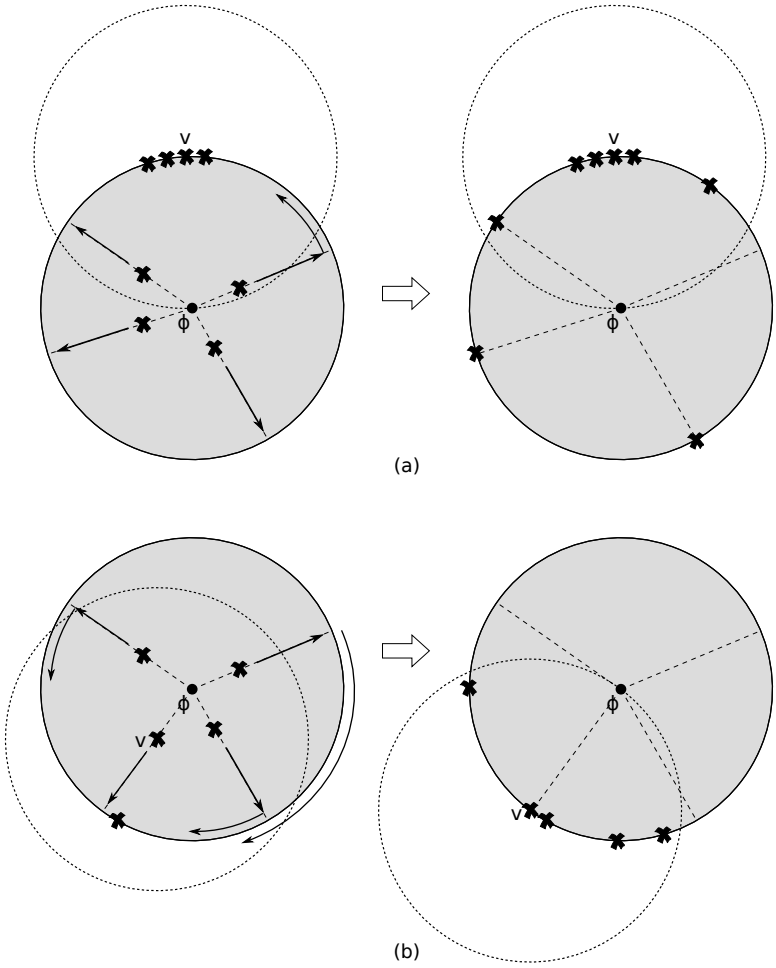

Figure 5.2: Maneuvers to maintain Δ. (a) $\Delta = \delta_v = 5$ for v in V_c. (b) $\Delta = \delta_v = 4$ for v in V_i.

to the previous one. Following the similar argument, the new maximum degree among V_i is no larger than the previous one by the maneuver Thus there is no increase in Δ.

Secondly, each vertex in V_i can move to the circumference of D_ϕ without decreasing Δ. Consider two cases: 1. Δ lies in the vertex in V_c and 2. Δ lies in the vertex in V_i. In both cases, we can first push all vertices to the circumference of D_ϕ, and then adjust these vertices to fix Δ. Figure 5.2 illustrates two examples, respectively.

The above arguments show that given a UDG where all vertices are within D_ϕ, it is always feasible to conduct a graph with Δ and $|V|$ remaining the same and $|V_i|$ decreasing to 0. This implies that $|V| = |V_c|$, which completes the proof. □

To determine $f_1(\Delta)$ where $V' - V = \{v\}$, then it is sufficient to consider the subgraph within D_v, according to Lemma 2. Further, if $\Delta' > \Delta + 1$ then $\Delta' = \delta_v$. The value of δ_v is the number of vertices within D_v. By Lemma 3, the maximum of this number equals the maximum number of vertices on the circumference of D_v. Thus, we can reduce the problem of $f_1(\Delta)$ in a random UDG to the problem of the maximum number of vertices on the circumference of D_ϕ. By inserting a vertex u to the center of D_ϕ, u is adjacent to all other vertices, and $f_1(\Delta) = \delta_u$. This leads to the following Corollary:

Corollary 1. *Given a UDG, the value of $f_1(\Delta)$ equals the maximum number of vertices on the circumference of D_ϕ.*

In the following of this thesis, we assume that all vertices are on the circumference of D_ϕ. Partition the circumference of D_ϕ into segments of equal arc length ε, with $\varepsilon \ll 1$. Each segment can have at most one vertex. Due to the property of UDG, the degree of a vertex is the number of vertices within $\frac{\pi}{3}$ arc length at both sides.

For example, given that $\Delta = 0$, Figure 5.3 shows the configuration on the circumference of D_ϕ, with exactly one vertex at each segment marked X. The regular hexagon in the background acts as a reference, and $|\overline{AB}| = |\overline{BC}| = |\overline{CD}| = |\overline{DE}| = r + \mu$, where $\mu \ll 1$. Except for those vertices on segments marked X, no vertex can occupy the segments on $\widehat{AB}$, $\widehat{BC}$, $\widehat{CD}$, and $\widehat{DE}$, because a vertex in any of these segments causes $\Delta = 2$. No vertex can occupy the segments on $\widehat{AE}$, because $\angle\widehat{AE} = \frac{2}{3}\pi - 4\varepsilon$ and a vertex

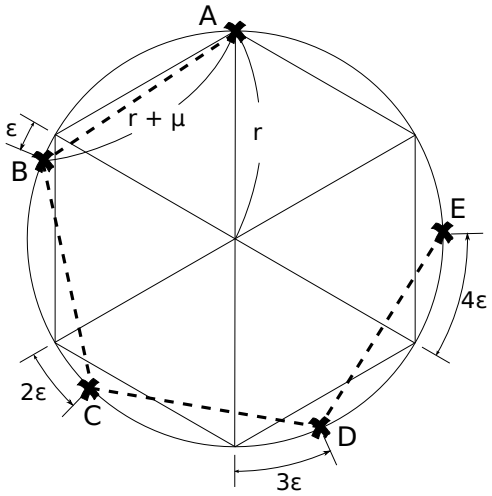

Figure 5.3: The configuration of five vertices when $\Delta = 0$.

in this region causes either $\Delta = 1$ or $\Delta = 2$. Thus a maximum of five vertices can lie on the circumference, occupying segments marked X.

For $\Delta > 0$, the configuration can be conducted by a similar method. This time, each X in the figure includes $\Delta + 1$ segments, and it can have at most $\Delta + 1$ vertices. The vertices in each X are not connected to those vertices positioned at other X, since they have distances longer than r.

Limitation on Placing Vertices

Due to the maximum degree Δ, each vertex can have at most Δ adjacent vertices. Define the term *credit* to indicate this limitation. All segments initially have zero credit, but placing a vertex in a segment increases the credit of all segments within Euclidean distance r by 1. That is, a vertex contributes $\frac{2\pi r}{3\varepsilon} + 1$ credits to the circumference. This study denotes a segment of k credits as a k-segment hereafter, and visualizes the number of credits by the height of the segment, as Figure 5.4 shows. A segment with a vertex is

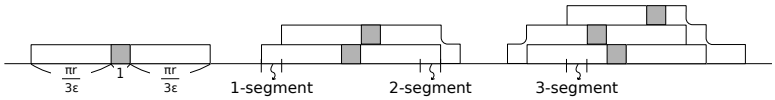

Figure 5.4: Three configuration examples on the circumference of D_ϕ; each gray square represents a vertex.

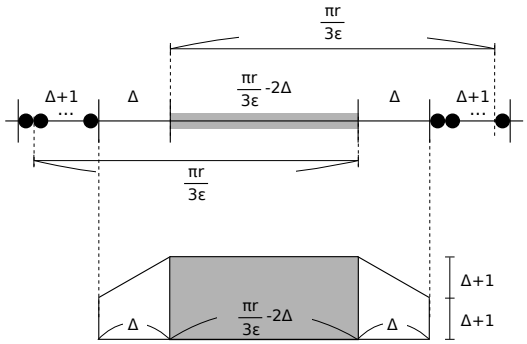

Figure 5.5: The maximum length of a continuous $2(\Delta + 1)$-segments (shaded region) is $\frac{\pi r}{3\varepsilon} - 2\Delta$.

an *occupied segment*, while a segment without a vertex is a *vacant segment*. Clearly, an occupied segment can have at most $\Delta+1$ credits. For vacant segments, there are at most $\Delta + 1$ vertices within distance r in a clockwise direction, and $\Delta + 1$ vertices within r in a counterclockwise direction. Therefore the maximum credit of a segment is $2(\Delta + 1)$. Finally, the absolute difference between adjacent segments is at most one credit, since no two vertices can occupy the same segment.

Any vacant segment has at most $2(\Delta + 1)$ credits, but not all vacant segments can reach this maximum value. There must be at least Δ segments between the occupied segment (i.e. $(\Delta+1)$-segment) and the $2(\Delta+1)$-segment, because the absolute difference between adjacent segments is at most one credit. Further, the occupied segments limit the length of a continuous $2(\Delta + 1)$-segment; if a segment has any credit, then it implies that at least one vertex is within Euclidean distance r. Thus for a $2(\Delta + 1)$-segment,

there must be one vertex within r at each side. The upper part of Figure 5.5 shows that the maximum length of a continuous $2(\Delta+1)$-segment is $\frac{\pi r}{3\varepsilon} - 2\Delta$. The vertices at the left determine the right boundary of the region, and the vertices at the right determine the left boundary.

5.2 The Upper Bound of Maximum Degree

Theorem 6. *In a unit disk graph with the maximum degree* Δ*,*

$$f_1(\Delta) = 5(\Delta+1). \tag{5.1}$$

Proof. Let *capacity* ($\mathcal{C}$) be the upper bound of total credits on a circumference. If there are x vertices, then the following equation holds:

$$\mathcal{C} \geq x(\frac{2\pi r}{3\varepsilon} + 1). \tag{5.2}$$

If $\mathcal{C}$ can be determined, and given x' such that $x'(\frac{2\pi r}{3\varepsilon} + 1) \geq \mathcal{C}$, then x' is the upper bound of x. According to Corollary 1, $f_1(\Delta) = x'$.

The lower part of Figure 5.5 shows the configuration of one continuous $2(\Delta+1)$-segment region and two corresponding intervals, with a total $\frac{\pi r}{3\varepsilon}$ segments. Let this configuration be a building block, and by concatenating these blocks with $(\Delta+1)$-segments in between and at both ends, the resulting region also has maximum credits. For n blocks, the equation below represents the number of segments remaining on the circumference:

$$\begin{aligned} & \frac{2\pi r}{\varepsilon} - n((\frac{\pi r}{3\varepsilon} - 2\Delta) + 2\Delta) - (n+1)(\Delta+1) \geq 0 \\ \Rightarrow \quad & \frac{\pi r(6-n)}{3\varepsilon} - (n+1)(\Delta+1) \geq 0. \end{aligned} \tag{5.3}$$

Recall that there are $\frac{2\pi r}{\varepsilon}$ segments in total on the circumference, and the number of remaining segments should be more than $\frac{\pi r}{3\varepsilon}$; otherwise the occupied segments at both

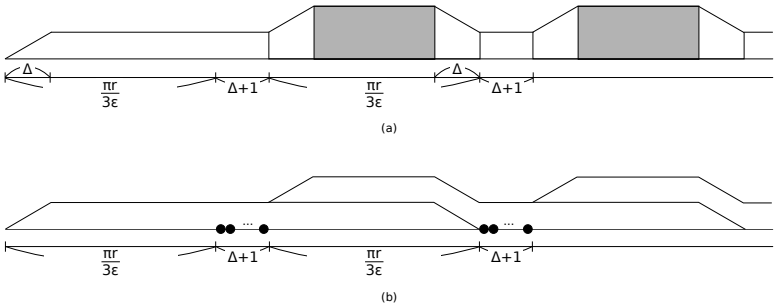

Figure 5.6: The configuration on the circumference.

ends will have more than $\Delta + 1$ credits, which means the maximum degree exceeds Δ. As a result, $n \leq 4$ and there are four such building blocks on the circumference of D_ϕ. Figure 5.6a shows half of this configuration.

No vertex can occupy any of the remaining segments without removing a vertex in blocks; otherwise the maximum degree will exceed Δ. By definition the distance of any two vertices must be greater than $\frac{\pi r}{3\varepsilon}$, a vertex in the remaining region is within this distance of at least one vertex in building blocks. Therefore, although removing one vertex in the blocks enables another vertex to occupy a segment in this region, this maneuver produces no extra credit.

Finally, by adding everything together, the capacity $\mathcal{C}$ can be determined:

$$\begin{aligned}
\mathcal{C} &= 4((\frac{\pi r}{3\varepsilon} - 2\Delta) \cdot 2(\Delta + 1) + 2 \sum_{k=\Delta+2}^{2\Delta+1} k) + 5(\Delta + 1)^2 + 2((\frac{\pi r}{3\varepsilon} - \Delta)(\Delta + 1) + \sum_{k=1}^{\Delta} k) \\
&= \frac{10\pi r(\Delta + 1)}{3\varepsilon} + 4(-4\Delta(\Delta + 1) + 2 \cdot \frac{\Delta(3\Delta + 3)}{2}) + 5(\Delta + 1)^2 \\
&\quad +2(-\Delta(\Delta + 1) + \frac{\Delta(\Delta + 1)}{2}) \\
&= \frac{10\pi r(\Delta + 1)}{3\varepsilon} + 5\Delta + 5 \\
&= (\frac{2\pi r}{3\varepsilon} + 1) \cdot 5(\Delta + 1) \qquad (5.4)
\end{aligned}$$

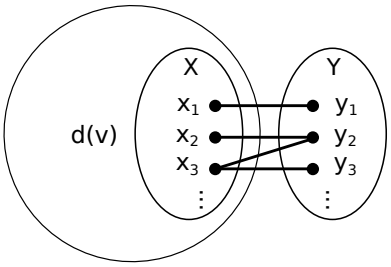

Figure 5.7: Illustration of Lemma 4.

and from Eq. (5.2), $\mathcal{C} \geq x(\frac{2\pi r}{3\varepsilon} + 1) \Rightarrow x \leq \frac{\mathcal{C}}{\frac{2\pi r}{3\varepsilon}+1} = 5(\Delta + 1)$. Hence, $5(\Delta + 1)$ is the upper bound of x and $f_1(\Delta) = 5(\Delta + 1)$. □

Figure 5.6b implies the same setting as in Figure 5.3. Further, the above arguments show that this configuration is unique in the presence of $\Delta' = f_1(\Delta)$. Since there is a feasible configuration for $5(\Delta+1)$, $f_1(\Delta) = 5(\Delta+1)$ is essentially the least upper bound.

Given $f_1(\Delta) = \delta_v$ and $v \in V'$, there should be five disconnected components in the subgraph of G within D_v (refer to Figure 5.3). The following Lemma states that the subgraph within D_v is disconnected from the subgraph outside D_v.

Lemma 4. *Given a vertex $v \in V'$ such that $\Delta' = f_1(\Delta) = \delta_v$, then $d(d(v)) \subset \{d(v) \cup v\}$.*

Proof. The configuration of $f_1(\Delta)$ in Figure 5.3 shows that each vertex in $d(v)$ has Δ adjacent vertices within D_v. If there exists a set $X \subset d(v)$ in which each vertex has at least one edge to another vertex set $Y \subset V$, and $d(v) \cap Y = \phi$ (see Figure 5.7), then vertices in X have a maximum degree larger than Δ, contradicting the maximum degree of Δ. Thus, to obtain $f_1(\Delta)$ with a given Δ, all adjacent vertices of a vertex in $d(v)$ must be either the vertex in $d(v)$ or the vertex v itself. □

Corollary 2. *Given a UDG with the maximum degree Δ, then $\Delta' = f_1(\Delta)$ only if G has more than six disconnected components.*

Practical scenarios usually guarantee the network connectivity, so we will prove that $f_1(\Delta) = 5\Delta + 1$ in the connected networks. Consider all vertices are put either within or on the circumference of D_ϕ.

Lemma 5. *Given the maximum number of vertices on a Δ-fixed UDG, every unit disk D_ψ with its center in D_ϕ contains at least one vertex $v \in V$ such that $\delta_v = \Delta$.*

Lemma 5 stands because if D_ψ contains no such vertex, then an additional vertex can be put at the center of D_ψ, contradicting that there are maximum number of vertices.

Theorem 7. *In a connected unit disk graph with the maximum degree Δ,*

$$f_1(\Delta) = 5\Delta + 1. \tag{5.5}$$

Proof. Lemma 5 implies it is always feasible to generate a set $V_s \subset V$ such that $\bigcap_{v \in V_s} D_v$ covers the whole area in D_ϕ. $\delta_v = \Delta$ for all v in V_s. Further, $\overline{uv} > r$ for arbitrary two vertices u and v in V_s. Let $|V_s| = m$ and $V_s = \{v_1, v_2, ..., v_m\}$. Clearly, $3 \leq m \leq 5$ can be determined. If $m \leq 2$, it is impossible to cover D_ϕ; if $m \geq 6$, it causes $\overline{uv} \leq r$ for some u and v in V_s. Figure 5.8 shows the cases of $m = 5$, 4, and 3, respectively. $R_i \subset D_\phi$ denotes the shaded region which contains vertex v_i. Let $S_i \subset G$ be the subgraph within R_i. The vertices put in R_i do not increase the degree of v_j in R_j, for all v_i in V_s and v_j in $\{V_s - v_i\}$. Thus, to maximize the number of nodes placed in D_ϕ, we should put as many vertices as possible in each shaded region, and place vertices to connect the subgraphs within these regions.

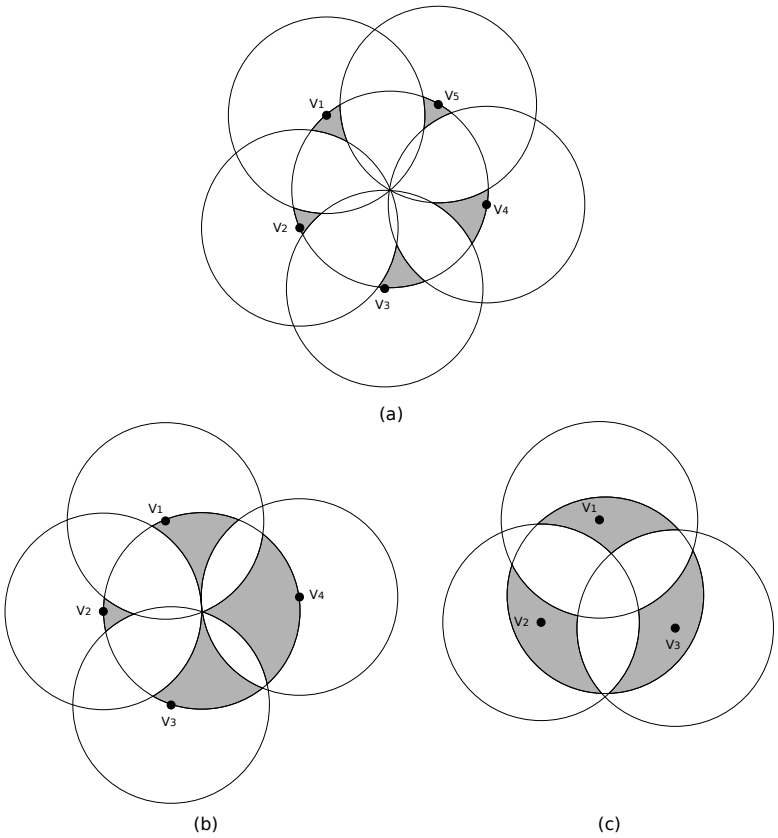

Figure 5.8: Illustration of the proof of Theorem 7 with (a) $m = 5$, (b) $m = 4$ and (c) $m = 3$.

Case 1. $m = 5$ (see Figure 5.8a). Due to that $\delta_v = \Delta$ for all v in V_s, each shaded region can have at most $\Delta + 1$ vertices. Since the shaded region R_i has the diameter shorter than r for $1 \leq i \leq 5$, the vertices within R_i should always form a complete subgraph S_i. Consider the procedure to connect S_1 and S_2. This can be done by placing a vertex in the intermediate region R'. However, to not exceeding the maximum degree, we must accordingly remove one vertex in each of S_1 and S_2. Thus, after connecting all five subgraphs, one in each shaded region, D_ϕ contains at most $5(\Delta+1)+(4-2\cdot4) = 5\Delta+1$ vertices.

Case 2. $m = 4$ (see Figure 5.8b). D_{v_2} and D_{v_4} have occupied more than $\frac{4}{3}\pi$ arc length of D_ϕ, and therefore at least one of the shaded regions R_1 and R_3 should have the arc length less than $\frac{1}{3}\pi$. Assume it is R_1 without loss of generality. Accordingly, R_1 must have its diameter shorter than r and have one complete subgraph S_1. By the similar argument, at least one of the shaded regions R_2 and R_4 has a complete subgraph. Assume it is R_2. To connect all S_i for $1 \leq i \leq 4$, at least two pairs among $(S_1,\ S_2)$, $(S_1,\ S_4)$, and $(S_2,\ S_3)$ should be connected. This leads to the total of vertices no more than $4(\Delta + 1) + (2 - 2 \cdot 2) = 4\Delta + 2$. Remark that $5\Delta + 1$ dominates due to $(5\Delta + 1) - (4\Delta + 2) = \Delta - 1 \geq 0$.

Case 3. $m = 3$ (see Figure 5.8c). After the connection, D_ϕ should contain no more than $3(\Delta + 1)$. Because $(5\Delta + 1) - 3(\Delta + 1) \geq 0$, $5\Delta + 1$ still dominates. This completes the proof. □

Theorem 8. *In a unit disk graph with the maximum degree* Δ*,*

$$f_n(\Delta) = f_1(\Delta) + s_n - 1, \tag{5.6}$$

where s_n *is the size of the largest connected component in* $\{V' - V\}$*.*

Proof. For a connected G', it is sufficient to assume that v is within D_ϕ for all v in V'. In this case, $s_n = n$ and Δ_n is the maximum degree of the graph after inserting the n-th vertex. If $\Delta_1 = f_1(\Delta)$, it is easy to see that $\Delta_2 = f_1(\Delta) + 1$; if $\Delta_1 < f_1(\Delta)$, then $\Delta_2 = \Delta_1 + 1$, which is at most equal to $f_1(\Delta)$. Therefore, $f_2(\Delta) = f_1(\Delta) + 2 - 1$. If $f_k(\Delta) = f_1 + k - 1$ for $k \in N$, then the similar argument leads to $f_{k+1}(\Delta) = f_1(\Delta) + (k + 1) - 1$. Thus, $f_n(\Delta) = f_1(\Delta) + s_n - 1$ by induction.

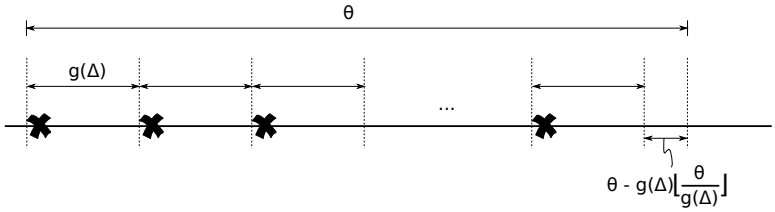

Figure 5.9: Illustration of Theorem 9.

For G' with disconnected components, each vertex in one component has no effect to the degree of vertex in the others, because the Euclidean distance between two vertices in different component is greater than r. As a result, it is sufficient to consider only the largest component, and the proof follows. □

Finally, consider the cast that only a certain angle within $D_{v'}$ contains $v \in V$. The following Theorem generalizes $f_n(\Delta)$ with respect to this angle.

Theorem 9. *Let θ be the minimum spanning radian that covers all $v \in V$ within D_ϕ. Let $g(\Delta) = \frac{\pi}{3} + \frac{(\Delta+1)\varepsilon}{r}$, and the general form of $f_n(\Delta)$ is given by:*

$$f_n(\Delta)\Big|_{0<\theta<\frac{5\pi}{3}} = \begin{cases} \lfloor \frac{\theta}{g(\Delta)} \rfloor (\Delta+1) + \lfloor \frac{\theta - g(\Delta)\lfloor \frac{\theta}{g(\Delta)} \rfloor}{\varepsilon} \rfloor + s_n - 1 & , \text{ if } \lfloor \frac{\theta - g(\Delta)\lfloor \frac{\theta}{g(\Delta)} \rfloor}{\varepsilon} \rfloor \leq (\Delta+1). \\ (\lfloor \frac{\theta}{g(\Delta)} \rfloor + 1)(\Delta+1) + s_n - 1 & , \text{ otherwise.} \end{cases} \tag{5.7}$$

Proof. In the construction example of $5(\Delta+1)$ vertices above, after concatenating $\Delta+1$ vertices, the next vertex must be placed beyond $\frac{\pi}{3}$ radian. In the resulting graph, there are five components on the circumference, each one containing $\Delta+1$ vertices, and the radian between two components is $\frac{\pi}{3}$. Thus, take $g(\Delta) = \frac{\pi}{3} + \frac{(\Delta+1)\varepsilon}{r}$ as the radian of a region containing one component. If θ is the minimum spanning radian that covers all vertices on the circumference, then there are $\lfloor \frac{\theta}{g(\Delta)} \rfloor$ components.

Figure 5.9 shows that the radian of the remaining region is $\theta - g(\Delta)\lfloor\frac{\theta}{g(\Delta)}\rfloor$, and therefore at most $\lfloor\frac{\theta-g(\Delta)\lfloor\frac{\theta}{g(\Delta)}\rfloor}{\varepsilon}\rfloor$ vertices can occupy this region. If $\frac{4\pi}{3} \leq \theta \leq 2\pi$, then no vertex can occupy the remaining region. Hence, $f_1(\Delta)$ can be represented as follows:

$$f_1(\Delta) \underset{0<\theta<\frac{5\pi}{3}}{=} \begin{cases} \lfloor\frac{\theta}{g(\Delta)}\rfloor(\Delta+1) + \lfloor\frac{\theta-g(\Delta)\lfloor\frac{\theta}{g(\Delta)}\rfloor}{\varepsilon}\rfloor & \text{, if } \lfloor\frac{\theta-g(\Delta)\lfloor\frac{\theta}{g(\Delta)}\rfloor}{\varepsilon}\rfloor \leq (\Delta+1). \\ (\lfloor\frac{\theta}{g(\Delta)}\rfloor+1)(\Delta+1) & \text{, otherwise.} \end{cases} \tag{5.8}$$

Theorem 8 leads to the proof of Eq. (5.7). □

Note that if $\theta = 5g(\Delta)$, then $\lfloor\frac{\theta-g(\Delta)\lfloor\frac{\theta}{g(\Delta)}\rfloor}{\varepsilon}\rfloor = 0$, and thus $f_1(\Delta) = \lfloor\frac{\theta}{g(\Delta)}\rfloor(\Delta+1) = 5(\Delta+1)$, which is reduced to Eq. (5.1).

5.3 Discussion

Let $G = (V, E)$ be a UDG and let $G' = (V', E')$ be the UDG after inserting a vertex on G. Besides $\Delta' \leq f_1(\Delta)$, another upper bound on the maximum degree is $\Delta' \leq |V'|-1 = |V|$. The value of $f_1(\Delta) = 5(\Delta+1)$ should be less than or equal to $|V|$, and thus $\Delta \leq \frac{|V|}{5} - 1$. For instance, consider the case of fifty vertices in a UDG ($|V| = 50$). In this case, $f_1(\Delta) \leq |V|$ if and only if $\Delta \leq \frac{50}{5} - 1 = 9$; for $\Delta > 9$, the trivial upper bound $|V|$ is tighter. Similarly, for the case of the connected UDG, $f_1(\Delta) = 5\Delta + 1 \leq |V|$ if and only if $\Delta \leq \frac{|V|-1}{5}$.

A typical wireless network example should demonstrate the tightness of $f_1(\Delta)$ compared with $|V|$. Consider a $x \times x$ square map, in which there are n wireless stations with equal communication ranges r. The scenario can be modeled as a UDG that $G = (V, E)$ and $|V| = n$. Each vertex represents a wireless station and an edge appears between two vertices if the corresponding stations can communicate with each other. The average

degree in G is

$$\delta_{avg} = \frac{r^2\pi(n-1)}{x^2} \tag{5.9}$$

if $\frac{r^2\pi}{x^2} < 1$.

Because $\Delta \geq \delta_{avg}$, the ratio of $f_1(\Delta)$ to the trivial upper bound n can be derived by

$$\begin{aligned} & \Delta \geq \frac{r^2\pi(n-1)}{x^2} \\ \Rightarrow \quad & \frac{5(\Delta+1)}{n} \geq \frac{5r^2\pi}{x^2} + \frac{5}{n} - \frac{5}{nx^2}. \end{aligned} \tag{5.10}$$

Let $k = \frac{r^2}{x^2}$, then

$$\frac{5(\Delta+1)}{n} \geq 5k\pi + \frac{5}{n} - \frac{5}{nx^2}. \tag{5.11}$$

By similar derivation,

$$\frac{5\Delta+1}{n} \geq 5k\pi + \frac{1}{n} - \frac{5}{nx^2}. \tag{5.12}$$

The parameter k is the ratio of the covering area of one wireless station to the area of the map. For example, given a 300×300 (200×200) meter-square area, and the communication range of each station is set to 40 (30) meters, then $k \approx 0.02$ [6, 10]; for a 160×160 meter-square area and the communication range 30 meters, $k \approx 0.03$ [15]; with 70×70 area and communication range 15, $k \approx 0.04$ [14]. It is also possible to have $k \approx 0.004$ or $k = 0.01$ [16, 17].

Figure 5.10 plots $\frac{5(\Delta+1)}{n}$ against k, and Figure 5.11 draws $\frac{5(\Delta+1)}{n}$ against n. For example, suppose that $k = 0.04$ and $n \leq 5$, $|V|$ is tighter than $f_1(\Delta)$ (because $\frac{f_1(\Delta)}{n} > 1$). If $n = 10$ and $k \geq 0.03$, then V is tighter. For the case of $k = 0.02$ and $n \geq 6$, then $|V|$ is not necessarily tighter than $5(\Delta+1)$, and the result is determined by comparing $5(\Delta+1)$ with n. For the same purpose, Figure 5.12 shows $\frac{5\Delta+1}{n}$ against k, and Figure 5.13 gives $\frac{5\Delta+1}{n}$ against n.

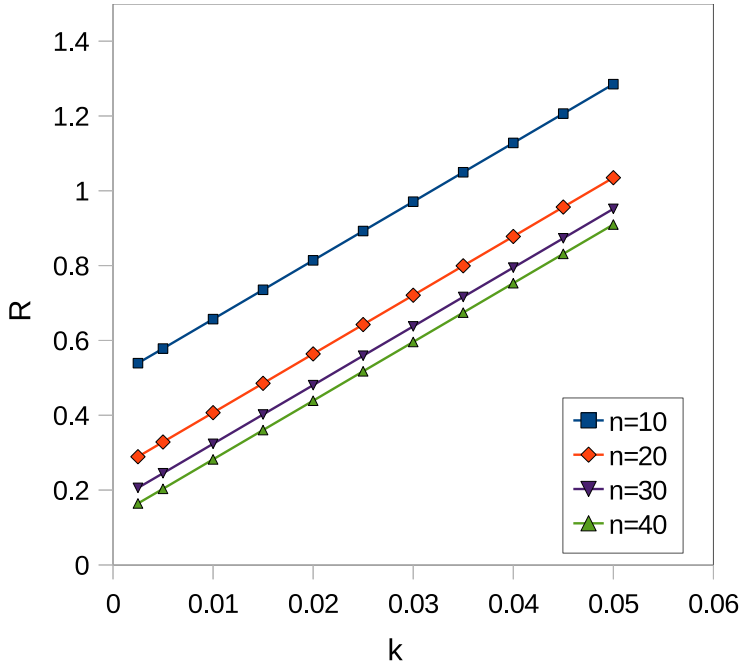

Figure 5.10: $f_1(\Delta) = 5(\Delta + 1)$

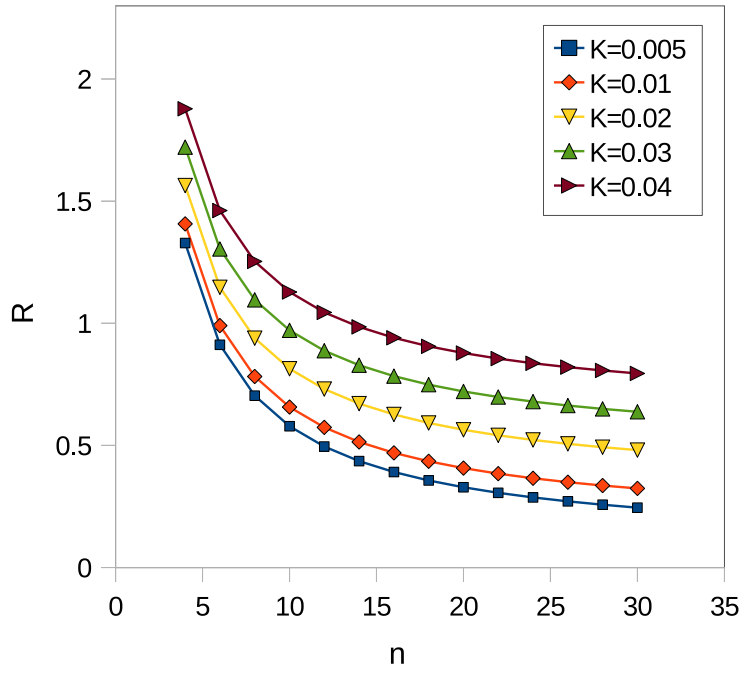

Figure 5.11: $f_1(\Delta) = 5(\Delta + 1)$

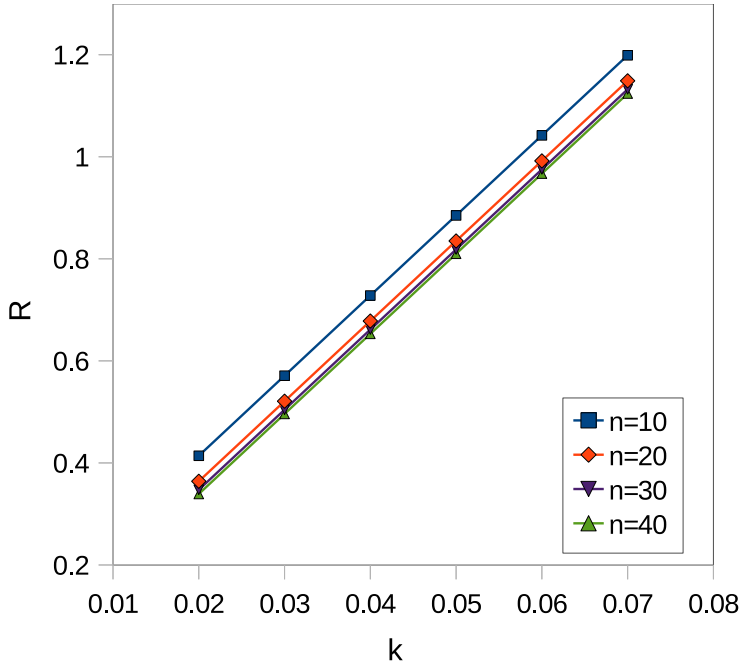

Figure 5.12: $f_1(\Delta) = 5\Delta + 1$

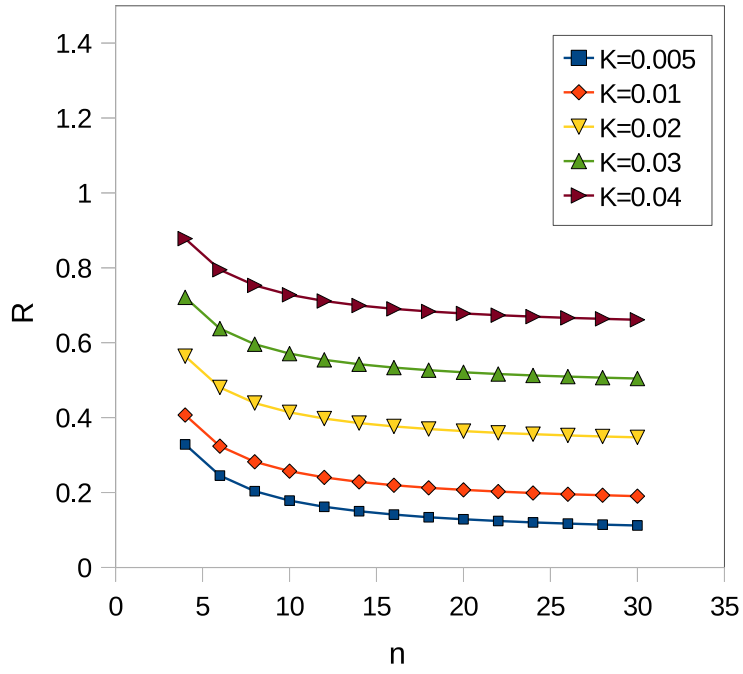

Figure 5.13: $f_1(\Delta) = 5\Delta + 1$

Chapter 6

Concluding Remarks

The graph coloring approach is not effective in the low-latency link scheduling, due to the hidden square phenomenon. The first part of this thesis studies the hidden squares identification problem. Accordingly, a distributed link scheduling scheme called DCLS is developed. With this scheme, the data buffering delay is significantly reduced, as compared to the strong edge-coloring approach. Next, this thesis introduces a centralized link scheduling shceme named GSA. GSA-1 is suitable for the slot assignments of the whole network; GSA-2 outperforms GSA-1 in delay when applied to single node. The performance improvement has been confirmed by both the analysis and simulation.

The second part of this thesis proves that the maximum degree Δ' in unit disk graphs, after inserting one vertex, at most equals $5(\Delta + 1)$. In connected UDGs, the least upper bound of Δ' is $5\Delta + 1$. Further, inserting n vertices leads to $\Delta' \leq 5(\Delta + 1) + s_n - 1$, where s_n is the size of the largest connected component among new vertices. Finally, the case study compares the proved upper bound with $|V|$ in wireless networks.

Chapter 7

Future Work

This Chapter gives brief discussions and rudimentary results on several topics.

7.1 Comparison on DCLS, GSA, and Related Work

DCLS outperforms GSA in that it is a distributed algorithm and the running time complexity is comparably low. However, unlike GSA, DCLS requires a predetermined schedule length, and DCLS may not able to assign all hidden slots. Therefore, it is important to compare the delay performance of two algorithms under various conditions, and to analysis the performance towards energy efficieny. Besides, it is also worthy to compare both DCLS and GSA with the related work.

7.2 A Local Rescheduling Scheme

This section sketches the idea of link scheduling on node insertion. For example as given in Figure 7.1, node A is inserted into the network, and nodes B, C, D, and E are its

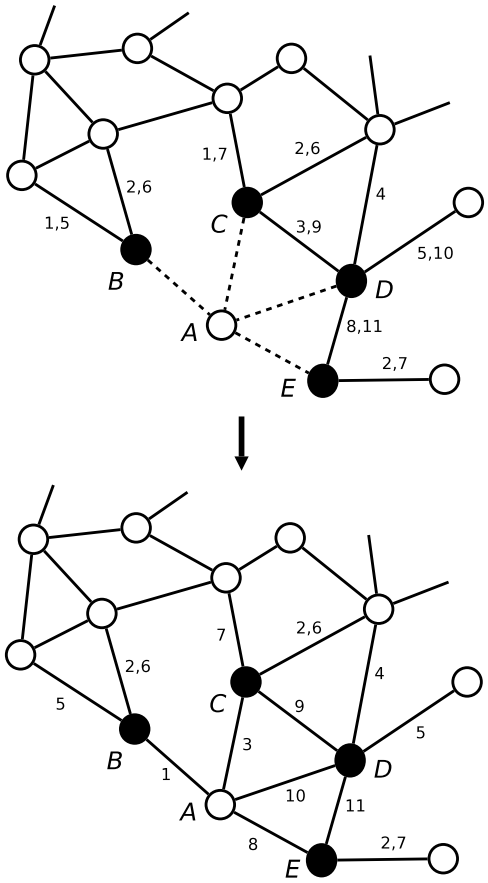

Figure 7.1: An example of the local rescheduling scheme.

one-hop neighbors. Each neighbor sends its slot information to A, so A will have the information as shown in the upper part of Figure 7.2.

Node A chooses one of the provided time slots from each neighbor node respectively, with a constraint that it should not choose the time slot which is the only slot assigned to an edge, because otherwise that edge will be disconnected. Each time slot number can be chosen only once, and the same slot in other edges must be removed in order to meet the link scheduling criterion.

During the algorithm, A maintains a time slot table shown in Table 7.1. The second row gives the number of times the slot appearing in the received information, and in the

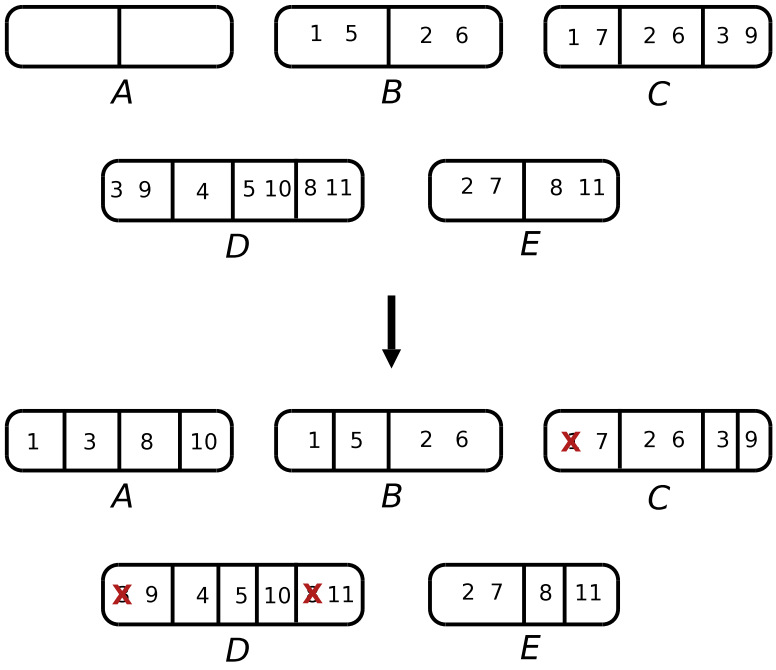

Figure 7.2: The owned time slots of each node before and after rescheduling.

Table 7.1: Slot information collected by node A.

time slot #	1	2	3	4	5	6	7	8	9	10	11
appearing times	2	3	2	1	2	2	2	2	2	1	2
$\zeta(slot)$	2	2	2	1	2	2	2	2	2	2	2
shared	5,7	6,7	9		1,10	2	1,2	11	3	5	8

third row $\zeta(slot)$ represents the least number of time slots in all edges assigned. The fourth row shows other time slots assigned to the same edge as the given time slot.

To begin with, scan through all edges and updates the corresponding time slot entries in the table. For example time slot #2 appears three times in the information, and among all edges it assigned, the minimum number of time slots on an edge is two. Sort the table so that the second row is in the decreasing order. The result is given in Table 7.2.

A then starts choosing the time slot with least appearing times and $\zeta(slot) > 1$. Once the time slot is chosen, $\zeta(slot)$ of all its shared time slots are updated if they are no

Table 7.2: Slot information of node A after sorting.

time slot #	4	10	1	3	5	6	7	8	9	11	2
appearing times	1	1	2	2	2	2	2	2	2	2	3
$\zeta(slot)$	1	2	2	2	2	2	2	2	2	2	2
shared		5	5,7	9	1,10	2	1,2	11	3	8	6,7

Table 7.3: Slot information of node A after local rescheduling.

time slot #	4	10	1	3	5	6	7	8	9	11	2
appearing times	1	1	2	2	2	2	2	2	2	2	3
$\zeta(slot)$	1	2	2	2	1	2	1	2	1	1	2
shared		5	5,7	9	1,10	2	1,2	11	3	8	6,7

longer the least value. For example, suppose that A chose time slots #10, 1, 3, and 8 for nodes D, B, C and E respectively, and the resulting table will be as shown in Table 7.3

After choosing the time slots for each neighbor node, A broadcasts the information to all its neighbors, which then accordingly adjust their schedules. As shown in the lower part of Figure 7.2, the neighbor of A either transfers the ownership of time slot to A, or gives up its ownership of that slot. The whole rescheduling process can be done locally, and other parts of the network are not affected.

If a inserted node cannot choose at least one time slot per neighbor, then it should query its neighbor nodes for the time slot. Figure 7.3 gives an example. The conflict color information are piggy-backed in the packet. Once the time slot is allocated, the ownership of that time slot is transfered along the path back to the inserted node and its neighbor. The detailed algorithm requires further studies.

Theorem 10. *This local rescheduling algorithm satisifies the link scheduling criterion.*

Proof. When a new node is added to the network and that it has two neighbor nodes u and v, then from the link scheduling criterion: $T_u \cap T_v \cap T_{u'} = \phi$, $\forall u' \in d(u)$, and

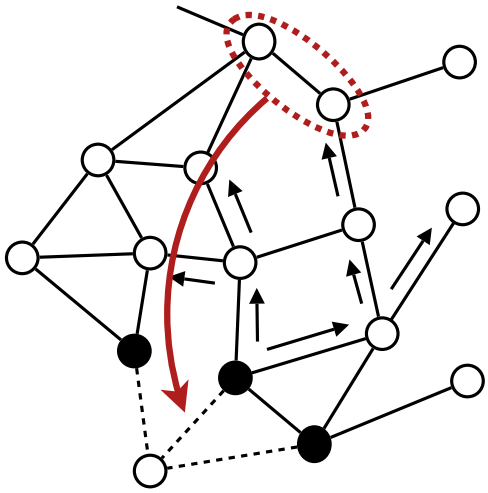

Figure 7.3: Queries for time slots.

$T_v \cap T_u \cap T_{v'} = \phi$, $\forall v' \in d(v)$. The equation directly follows: $T_u \cap T'_u \cap T_v \cap T'_v = \phi$. Because u and v simply transfer the ownerships of time slot to the new node, the criterion is still hold after the rescheduling, according to the above equation. □

7.3 Subnetwork Concatenation

Consider a group of nodes are inserted into the existing network from the boundary, shown in Figure 7.4. In this case, the network concatenation can be done in the following manner. Firstly, apply the local rescheduling to the nodes which are one-hop from the existing network. Based on the slot information acquired by these boundary nodes, the rest of nodes in the subnetworks can be scheduled by the GSA algorithm proposed in Chapter 4. For the related work, refer to [29]

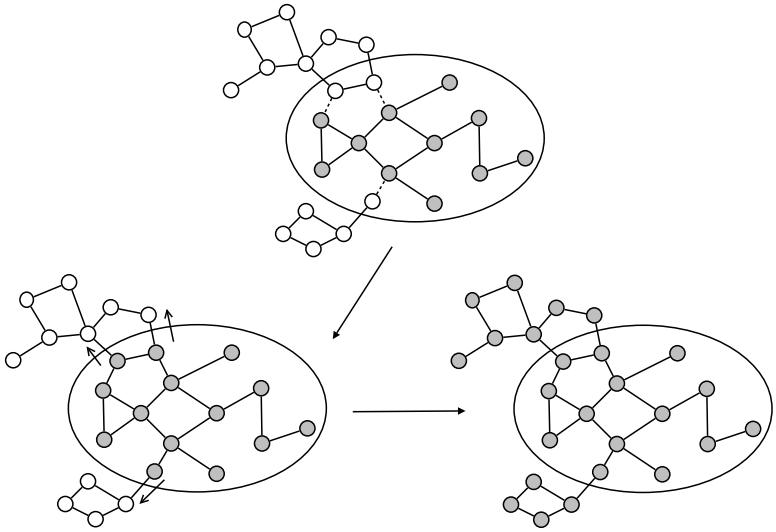

Figure 7.4: Illustration of the subnetwork concatenation.

7.4 Scheduling in Mobile Networks

In mobile networks, the degree of each wireless node varies due to the frequent changes in network topology. This characteristic makes the reserved time slot only valid in a short period of time, and, in addition, time slots should be assigned to the newly established communication link. As a result, it leads to heavy overhead in rescheduling. Therefore, mobile network protocols often apply the carrier sense technology instead of TDMA approach.

With the knowledge of degree upper bound on vertex insertion, proposed in Chapter 5, it is possible to design a TDMA scheduling protocol in mobile netowrks with lower overhead. Given a node v, let d denote the number of its one-hop neighbor nodes, and let s denote the number of its *shadow nodes*. The shadow node acts as a virtual identity that indicates the potential neighbor node of v. v then joins the slot assignment with the degree set to be $d + s$. Let S_v be the set of time slots assigned to shadow nodes.

After the slot assignment, v designates those slots in S_v to its one-hop neighbor nodes. Suppose that a new communication link is established sometime in the future, the slot assignment can be done by simply transfering the slot in S_v to the new neighbor node of v. Since the time slots in S_v satisify the link scheduling constraint, the redesignation satisifies the constraint as well. As compared with the rescheduling among all neighbor nodes, the procedure above should reduce overhead. For the related work on the mobile network scheduling, refer to [30–36].

7.5 The Maximum Degree on Vertex Insertion in Other Graphs

Chapter 5 gives the least upper bound of degree in the case of unit disk graphs. It is interesting to study the upper bound on double disk graphs or unit sphere graphs, which are, respectively, the interference model and the model of three dimensional wireless networks.

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] *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, IEEE 802 LAN/MAN Standards Commitee, 1999.

[4] D. B. West, *Introduction to Graph Theory.* Prentice-Hall, 2001.

[5] I. Slama, B. Jouaber, and D. Zeghlache, "A Free Collision and Distributed 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 Annual Joint Conference of IEEE Computer and Communications Societies (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 Annual IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOMW'06)*, Mar. 2006, pp. 105–110.

[8] V. Rajendran, K. Obraczka, and J. J. Garcia-Luna-Aceves, "Energy-Efficient Collision-Free Medium Access Control for Wireless Sensor Networks," in *Proceedings of International Conference on Embedded Networked Sensor Systems (SenSys'03)*, Nov. 2003, pp. 181–192.

[9] Y. Wang and I. Henning, "A Deterministic Distributed TDMA Scheduling Algorithm for Wireless Sensor Networks," in *Proceedings of IEEE Internation Conference on Wireless Communications, Networking and Mobile Computing (WICOM'07)*, Sept. 2007, pp. 2759–2762.

[10] I. Rhee, A. Warrier, J. Min, and L. Xu, "DRAND: Distributed Randomized TDMA Scheduling for Wireless Ad-Hoc Networks," *IEEE Transactions on Mobile Computing*, vol. 8, no. 10, pp. 1384–1396, Oct. 2009.

[11] S. H. Wu, M. S. Chen, and C. M. Chen, "Fully Adaptive Power Saving Protocols for Ad Hoc Networks Using the Hyper Quorum System," in *Proceedings of International Conference on Distributed Computing Systems (ICDCS'08)*, June 2008, pp. 785–792.

[12] J. R. Jiang, Y. C. Tseng, C. S. Hsu, and T. H. Lai, "Quorum-Based Asychronous Power-Saving Protocols for IEEE 802.11 Ad Hoc Networks," *Mobile Networks and Applications*, vol. 10, no. 1-2, pp. 169–181, Feb. 2005.

[13] J. Ma, W. Lou, Y. Wu, X.-Y. Li, and G. Chen, "Energy Efficient TDMA Sleep Scheduling in Wireless Sensor Networks," in *Proceedings of Conference on Computer Communications (INFOCOM '09)*, Apr. 2009, pp. 630–638.

[14] H. A. B. F. de Oliveira, A. Boukerche, E. F. Nakamura, and A. A. F. Loureiro, "An Efficient Directed Localization Recursion Protocol for Wireless Sensor Networks," *IEEE Transactions on Computers*, vol. 58, no. 5, pp. 677–691, May 2009.

[15] J. Kang, Y. Zhang, and B. Nath, "TARA: Topology-Aware Resource Adaptation to Alleviate Congestion in Sensor Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 7, pp. 919–931, July 2007.

[16] F. Ye, G. Zhong, S. Lu, and L. Zhang, "GRAdient Broadcast: A Robust Data Delivery Protocol for Large Scale Sensor Networks," *Wireless Networks*, vol. 11, no. 3, pp. 285–298, May 2005.

[17] A. Mei and J. Stefa, "Routing in Outer Space: Fair Trafc Load in Multi-Hop Wireless Networks," in *Proceedings of ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'08)*, May 2008, pp. 23–32.

[18] S. C. H. Huang, P. J. Wan, J. Deng, and Y. S. Han, "Broadcast Scheduling in Interference Environment," *IEEE Transactions on Mobile Computing*, vol. 7, no. 11, pp. 1338–1348, Nov. 2008.

[19] B. N. Clark, C. J. Colbourn, and D. S. Johnson, "Unit Disk Graphs," *Discrete Mathematics*, vol. 86, no. 1-3, pp. 165–177, Dec. 1990.

[20] M. V. Marathe, H. Breu, H. B. H. Iii, S. S. Ravi, and D. J. Rosenkrantz, "Simple Heuristics for Unit Disk Graphs," *Networks*, vol. 25, no. 2, pp. 59–68, 1995.

[21] H. Garcia-Molina and D. Barbara, "How to Assign Votes in a Distributed System?" *Journal of the ACM (JACM)*, vol. 32, no. 4, pp. 841–860, Oct. 1985.

[22] S. Y. Cheung, M. Ahamad, and M. H. Ammar, "The Grid Protocol: a High Performance Scheme for Maintaining Replicated Data," in *Proceedings of International Conference on Data Engineering*, Feb. 1990, pp. 438–445.

[23] H. Kakugawa, S. Kamei, and T. Masuzawa, "A Token-Based Distributed Group Mutual Exclusion Algorithm with Quorums," *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, no. 9, pp. 1153–1166, Sept. 2008.

[24] M. Mahdian, "The Strong Chromatic Index of Graphs," Master's thesis, Department of Computer Science, University of Toronto, 2000.

[25] 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.

[26] M. Luby, "A Simple Parallel Algorithm for the Maximal Independent Set Problem," in *Proceedings of ACM Symposium on Theory of Computing (STOC'85)*, Dec. 1985, pp. 1–10.

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

[28] 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.

[29] A. Kanzaki, T. Hara, and S. Nishio, "An Adaptive TDMA Slot Assignment Protocol in Ad Hoc Sensor Networks," in *Proceedings of ACM Symposium on Applied Computing*, Mar. 2005, pp. 1160–1165.

[30] I. Chlamtac and A. Faragó, "Making Transmission Schedules Immune to Topology Changes in Multi-hop Packet Radio Networks," *IEEE/ACM Transactions on Networking*, vol. 2, no. 1, pp. 23–29, 1994.

[31] S. Boztas, "A Robust Multi-Priority Topology-Independent Transmission Schedule for Packet Radio Networks," *Information Processing Letters*, vol. 55, no. 5, pp. 291–295, 1995.

[32] V. R. Syrotiuk, C. J. Colbourn, and A. C. Ling, "Topology-Transparent Scheduling for MANETs Using Orthogonal Arrays," in *Proceedings of Joint Workshop on Foundations of Mobile Computing (DIALM-POMC '03)*, Sept. 2003, pp. 43–49.

[33] T. Herman and S. Tixeuil, "A Distributed TDMA Slot Assignment Algorithm for Wireless Sensor Networks," *Algorithmic Aspects of Wireless Sensor Networks*, pp. 45–58, July 2004.

[34] F. Farnoud and S. Valaee, "Reliable Broadcast of Safety Messages in Vehicular Ad Hoc Networks," in *Proceedings of Conference on Computer Communications (INFOCOM '09)*, Apr. 2009, pp. 226–234.

[35] S. Basagni and D. Bruschi, "A Logarithmic Lower Bound for Time-Spread Multiple-Access (TSMA) Protocols," *Wireless Networks*, vol. 6, no. 2, pp. 291–295, May 2000.

[36] A. E. F. Clementi, A. Monti, and R. Silvestri, "Distributed Broadcast in Radio Networks of Unknown Topology," *Theoretical Computer Science*, vol. 302, no. 1-3, pp. 337–364, 2003.

Vita

Chao Wang was born on December 8, 1986, in Kaohsiung City, Taiwan. He received the B.S. degree in Electrical Engineering from National Cheng Kung University, Taiwan, in 2009. In September 2009, he enrolled at the National Cheng Kung University as a master student in the Institute of Computer and Communication Engineering.