# Effects of vertex insertion on local rescheduling in wireless sensor networks

Chun-Hao Yang*

Department of Electrical Engineering,
National Cheng Kung University
Tainan 970, Taiwan
Email: jeffy@dcl.ee.ncku.edu.tw
*Corresponding author

Kuo-Feng Ssu

Department of Electrical Engineering,
National Cheng Kung University,
Tainan 701, Taiwan
Email: ssu@ee.ncku.edu.tw

Chao Wang

Department of Computer Science and Engineering,
Washington University in St. Louis,
One Brookings Drive, St. Louis, MO 63130, USA
Email: chaowang@wustl.edu

**Abstract:** Rescheduling has abundant issues yet to be explored. The *local rescheduling problem* in wireless sensor networks (WSNs) have been firstly addressed and investigated in this paper. The algorithms of local rescheduling have been proposed and evaluates the performance of reschedule solutions with different metrics. The solutions have to be under the limitation that the network should stay connected after the process of rescheduling. This paper introduces a theoretical bound of maximum degree after node insertion. Along with empirical results in real world settings, the results motivate the design of algorithms and give possible reasons why existing rescheduling algorithms do not work efficiently. Two local link rescheduling algorithms and one local broadcast rescheduling algorithm are developed as improvements. With different node densities and other critical parameters, simulations show that the developed algorithms greatly improve the ratio of finding *proper* solutions successfully in both types of scheduling compared with other existing simple algorithms.

**Keywords:** local rescheduling; WSN; wireless sensor network; local link scheduling; local broadcast scheduling; maximum degree variation.

**Reference** to this paper should be made as follows: Yang, C-H., Ssu, K-F. and Wang, C. (2018) 'Effects of vertex insertion on local rescheduling in wireless sensor networks', *Int. J. Ad Hoc and Ubiquitous Computing*, Vol. 27, No. 3, pp.195–209.

**Biographical notes:** Chun-Hao Yang received his BS in Electrical Engineering and the MS degree in Computer and Communication Engineering, both from the National Cheng Kung University, Tainan, Taiwan. He received his PhD in Computer Science in the Institute of Computer and Communication Engineering at the National Cheng Kung University. His research interests include underwater sensor networks, time synchronisation, and routing protocols analysis in sensor networks.

Kuo-Feng Ssu received his BS in Computer Science and Information Engineering from the National Chiao Tung University and the PhD in Computer Science from the University of Illinois, Urbana-Champaign. He is a Professor in the Department of Electrical Engineering, National Cheng Kung University, Tainan, Taiwan. He was a Visiting Associate Professor in the School of Electrical and Computer Engineering, Cornell University, Ithaca, New York. He was a Visiting Scholar in the Department of Computer and Information Sciences, University of Delaware, Newark. His research interests include mobile computing, dependable systems, and distributed systems. His research awards include the Ta-You Wu Memorial Award, the K.T. Li Research Award, and the Lam Research Thesis Award. He is a Member of the IEEE, the ACM, and the Phi Tau Phi Honor Scholastic Society.

Chao Wang received his BS in Electrical Engineering and the MS from the Institute of Computer and Communication Engineering, both from the National Cheng Kung University, Taiwan. He is a PhD student in the Department of Computer Science and Engineering, Washington University in St. Louis. He works in cyber-physical systems.

## 1 Introduction

Rapid developments in wireless technologies have enabled many applications in wireless sensor networks (WSNs). Sensor nodes may be placed in a wild area to collect geological information or be positioned around a facility to detect invasion. Recent research work extends the applications to three-dimensional space; for example, underwater sensor networks are deployed to monitor aquatic activities. Topics on aerial sensor networks also populate the literature (Ravindran and Narayanasamy, 2011; Dhurandher et al., 2010; Chen et al., 2007; Yang and Ssu, 2008; Chen et al., 2011; Ou and Ssu, 2008).

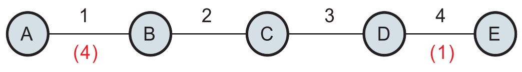
Wireless nodes operating at the same frequency may interfere with each other. The situations are classified into either *primary interference* or *secondary interference* (Ramanathan and Lloyd, 1993). Interference occurs if the receiver is within the transmission ranges of multiple transmitters. Interference may lead to longer delay and less successful delivery ratio in communication. A well-established strategy to resolve the interferences is to construct a schedule of transmission. The schedule is composed of *time slots*, and each pair of nodes is allowed to communicate at the time slots which were assigned to it. A carefully-chosen slot assignment has higher possibility for interference-free communications.

The scheduling protocols can be divided into two categories: link scheduling and broadcast scheduling. In link scheduling, time slots are assigned to both nodes involved in a transmission. In broadcast scheduling, slots are assigned to the node that sends messages to all other nodes within its radio coverage. The call for rescheduling arises in, but it is not limited to the following situations. First, a mobile node cruising around the network collecting information. Next, a new node being introduced to replace the faulty node. Third, a pile of new nodes being deployed to extend a network or to concatenate multiple networks. Most situations do not favour a global rescheduling operation since the nodes involved cannot continue their regular tasks during the operation. In addition, it takes huge efforts to reschedule the existing schedule of the whole network.

Local rescheduling algorithms have been arisen by exploiting the *hidden square phenomenon* (Wang and Ssu, 2010). The phenomenon states that each link/node can be assigned multiple time slots in a single schedule. This is the first paper which formulates and studies the problems of local rescheduling. In this paper, two delicate algorithms are developed considering the phenomenon by using extra available time slots. The algorithm is novel because it accounts for both link scheduling and broadcast scheduling and offers local reschedule solution for packet transmission that improves successful scheduling rate. Besides, a new analysis technique has been presented subject to link connectivity's guarantee. It is expected that the theoretical analysis result will help enable the use of WSNs in real-time applications.

Figure 1 gives an example of link rescheduling. There are five nodes positioned along a line. A schedule with four time slots is shown above the links. Suppose a new node is introduced to the left of the network. Then either Slot #1 or Slot #4 can be transferred to the new node.

**Figure 1** An example of transmission schedule (see online version for colours)



To study the influences where a node enters a scheduled network, a theoretical upper bound of maximum degree after node insertion has been proved in this paper. As inducted in Appendix, $5(\Delta + 1)$ is the maximum degree for any topologies of the networks with any positions that new node enters, where $\Delta$ is the maximum degree of the original network without the new vertex. Empirical results of the maximum degree increases had also been performed. In simulations, all nodes are randomly deployed with uniform distribution throughout the network. Note that the event of theoretical bound barely happens, unless the locations of nodes including new coming node are perfectly arranged. To further understand the maximum degree variance that corresponds to reality, the statistical results from empirical simulations are then collected. Nevertheless, even with smaller maximum degree variance in real world compared with that in theory, *proper* solutions cannot be found with ease. A rescheduling algorithm is said to produce a *proper* solution if no interference happens where, in link rescheduling, the nodes that share a common link preserves at least one identical time slot; in broadcast rescheduling, each node preserves at least one time slot. The local rescheduling algorithm is said to be *failed* in situations where no proper solution can be found.

The performance in local rescheduling relates to some other factors. For example, cycle length of the schedule, slot request from the new vertex, currently occupied or used slots of each node, and node density of the network. These factors are all playing important roles in finding a proper solution. This paper introduces algorithms to local link rescheduling and local broadcast rescheduling respectively, followed by a simple and unified strategy that will always produce proper solutions in both types of rescheduling. With the idea of slot transferring from a slot-spare node to the new coming node, simulation results represent that the proposed algorithms greatly improve the ratio of finding *proper* solutions successfully in both types of scheduling.

This paper is organised as follows. In Section 2 the related work is presented. Some preliminaries, common assumptions, definitions, and notations that are used throughout the paper are presented in Section 3. Moreover, since this is the first paper which addresses the local rescheduling (LR) problem, the intuitive and simple solutions to the LR problem are developed for further comparison. In Section 4, refined local link rescheduling and broadcast rescheduling algorithms are presented. Section 5 established theoretical lower bound on the need of the number of the redundant time slots for 100-percent acquiring *proper* solutions. In addition, the overhead of the theoretical solutions are discussed. Section 6 presents our simulation results with distinct metrics. Section 7 draws our conclusions and the Appendix gives the detail of the proofs to the proposed theorems.

## 2 Related work

In wireless networks, wireless nodes may cooperate in routing packets of interest. To increase the network throughput, the reduction of collisions is one of the main issues. Therefore, some related papers were proposed which take advantage of the asynchronised medium access control (MAC) protocols, or synchronised TDMA, CDMA, FDMA, and the dynamic assignment of frequency bands to avoid possible packets collisions (Pantelidou and Ephremides, 2011; Doudou et al., 2013b; Ahn et al., 2006; Wang et al., 2006)

LLMAC uses simple asynchrony message package for frame schedule between neighbour nodes instead of SYNC package in S-MAC, and brings in the stagger active schedule mechanism to maintain the data forwarding transmission (Yu et al., 2007). Duo-MAC is an asynchronous cascading wake-up MAC protocol for WSNs (Doudou et al., 2013a). It achieves energy-time constrained data delivery, and balances energy-efficiency with delay-minimisation by adaptively switching between two states according to the dominating traffic in the network. Park et al. (2011) uses a randomised routing, a CSMA/CA mechanism at the MAC, radio power control at the physical layer, and sleeping disciplines. This paper considers duty cycle, routing, MAC, and physical layers all together to maximise the network lifetime by taking into account the trade-off between energy consumption and application requirements for control applications. CyMAC adopts a receiver-initiated beacon-based strategy. Each node periodically wakes up and sends out a short beacon to explicitly notify its neighbours that it is ready to receive data (Peng et al., 2011). This asynchronised protocol provides the desired relative delay bound guarantee for data delivery services via planning the rendezvous schedules carefully, and adjusts the sensor nodes' duty cycles dynamically to the varying traffic condition. Another async protocol Diff-MAC integrates different methods to meet the requirements of QoS provisioning to deliver heterogeneous traffic (Yigitel et al., 2011). Diff-MAC coordinates the medium access of each traffic class by using effective service differentiation mechanisms. Fragmentation and message passing feature of Diff-MAC reduces the retransmission cost and CW size adaptation mechanism tries to balance both energy consumption and delay.

Time-based MAC has potential advantages over FDMA and CDMA approaches in terms of hardware simplicity, energy efficiency, and low delay time (Ye et al., 2002; Yackoski and Shen, 2008). A number of centralised TDMA-based research were proposed with many applications (Ramanathan, 1997; Hossain and Bhargava, 2001). However, a centralised control mechanism requires at least a powerful node which recognises the behaviours and mobilities of nodes and it is costly to reach the requirements in empirical scenarios. Besides, centralised systems fail easily if some nodes break down during operation and lead to lower reliability. Nevertheless, concerned with energy issue in wireless networks, the amortised cost of collecting messages from nodes in a distributed manner is greatly reduced, compared with centralised protocols. Hence, several distributed version of TDMA-based MAC protocols are proposed (Rhee et al., 2009; Ammar and Stevens, 1991). Munir et al. (2010) calculate an upper bound on the latency of all the streams so that all the packets of all the streams reach their destinations within their respective latency bounds. A novel framework from pTunes provides runtime parameter adaptation for low-power MAC protocols, automatically translating application-level requirements into MAC parameters that meet these requirements (Zimmerling et al., 2012). These works claim to effectively reduce the number of collisions and increase the network throughput. However, to both achieve robust and collision-free communication, TDMA broadcast scheduling and link scheduling problems (Arikan, 1984; Even et al., 1984) are proved to be NP-complete problems in wireless ad hoc networks (Ephremides and Truong, 1990).

To help appropriately reschedule and allocate time slots with fairness, it is essential to have a sufficient long enough period or length in any TDMA-based scheduling algorithms. This paper finds a least upper bound after node insertions and thus gives an insight for designer to devise a much more efficient rescheduling algorithm.

The proof of the added node's maximum degree includes the results of the kissing number problem in three-dimensional space, also known as the 13 spheres problem (Anstreicher, 2004; Hales, 2002; Conway and Sloane, 1999). Various proofs of the 13 spheres problem populated the literature (Schütte and van der Waerden, 1953; Leech, 1956).

## 3 Preliminaries

### *3.1 Definition*

This paper considers the network $G = (V, E)$, where $V$ is the set of nodes positioned in the $k$-dimensional Euclidean space $R^k$, and $E$ is a subset of bi-directional links defined by $\{(u, v) \mid (u, v) \in V^2\}$ (The notations $V$ and $V(G)$ are interchangeable). A link connects two nodes $v$ and $u$ if and only if the distance $d(v, u)$ is shorter than or equal to one. Two nodes are said to be *adjacent node* if they share at least one link. Two links are said to be *adjacent link* if they share the same endpoint of the links. The degree of a node is the number of its adjacent nodes. Let $\Delta$ be the maximum degree among all nodes in a network.

In link scheduling, define $T_{\overline{ab}}$ as the set of shared time slots between node $a$ and node $b$. The $Distance$ of two non-adjacent links are defined as size of the smallest set of edges $e \subset E$ that connects two links. $\text{Dist}_{\overline{ab},\overline{cd}}$ is the number of hop count from link $\overline{ab}$ to link $\overline{cd}$. The link scheduling criteria is stated as follows:

**Definition 1:** $\forall i, j, k, l \in V$ and $\overline{ij}, \overline{kl} \in E$, the link scheduling is interference-free if $T_{\overline{ij}} \cap T_{\overline{kl}} = \phi$, where $\overline{ij}, \overline{kl}$ are arbitrary two links with $Dist_{\overline{ij},\overline{kl}} < 2, i \neq j, k \neq l$, and $\overline{ij} \neq \overline{kl}$.

In broadcast scheduling, let $T_v$ be the set of time slots of Node $v$. The hop count of two nodes are defined as size of the smallest set of edges $e \subset E$ that connect two nodes. $\text{Hop}_{(a,b)}$ is the hop count from Node $a$ to Node $b$. The broadcast scheduling criteria is defined as:

**Definition 2:** $\forall i, j \in V$, the broadcast scheduling is interference-free if $T_i \cap T_j = \phi$, where $Hop_{(i,j)} < 2, i \neq j$.

## 3.2 *Relationship between $\Delta'$ and $\Delta$*

Intuitively, the probability of rescheduling successfully has negative correlation to node density. In other words, higher degree of the new adding node leads to more interferences and hence fails to find proper solutions. As derived in Appendix, the following theorems are the findings about theoretical upper bound of maximum degree after the new node insertion.

**Theorem 1:** *In any two-dimensional disk graph, after one vertex insertion, the degree of the new vertex is at most equal to* $5(\Delta + 1)$.

**Theorem 2:** *In any three-dimensional disk graph, after one vertex insertion, the degree of the new vertex is at most equal to* $12(\Delta + 1)$.

Theorems 1 and 2 give an upper bound of maximum degree after node insertion, which is denoted as $\Delta'$, in 2D and 3D networks. In a 2D real world with random node deployment, as shown in Figure 2, the increases of $\Delta$ after vertex insertion are not as many as the upper-bound case. The parameter $N$ is the number of nodes in the network. The results suggest that $\Delta'$ tends to be much less than $5(\Delta + 1)$(theoretic upper bound) in reality. To be much more representative, for each setting of the number of node $N = 10, 25, 50,$ and $100$, this simulation constructs 100,000 distinct topologies with 100 times vertex insertions in each topology. Distinct $\Delta$s are collected and the distribution are depicted in Figure 2. It can be seen that the increase of maximum degree is more obvious when $\Delta$ is rather smaller. Topologies with higher $\Delta$ only have less than 30% of chance that $\Delta' > \Delta$. In addition, the maximum degree of smaller number of nodes changes much more than that of higher density networks; in other words, large number of nodes does not change much in degree variance.

Moreover, the $\Delta'$ of the *connected* networks has been proved and can been summarised as in Theorem 3:

**Theorem 3:** *In a two-dimensional, connected disk graph, the degree of the new vertex after insertion is at most equal to* $5\Delta$*; the degree is at most equal to* $12\Delta$ *in a three-dimensional, connected disk graph.*

## 3.3 *Simple rescheduling algorithm*

It is assumed that the original network had been scheduled by a protocol that exploiting the *hidden square phenomenon* (e.g., DCLS (Wang and Ssu, 2010)). Thus, some links/nodes may have multiple time slots. The proposed novel local rescheduling schemes are based on transferring time slots to a new link/node. The performance of simple and intuitive solutions to local rescheduling is investigated, including 1000 topologies and 100 node injections. The network size is set to $1000 \times 1000$ and the cycle length is 100 slots. Without invoking any interferences during network constructions, each node strives to fulfil its own slot requests one by one, which is denoted as *occupied slot* throughout the thesis. Occupied slots of nodes are ranged from 2 to 15, 6 to 30, in link scheduling and broadcast scheduling, respectively.

*Simple link rescheduling (SLR)* and *simple broadcast rescheduling (SBR)* algorithms are proposed for evaluating the efficiency of existing rescheduling solutions. The main idea of both algorithms is to simply avoid interferences from neighbours after node addition. The remaining interference-free slots are chosen to be part of proper solutions. There are intrinsic differences between link scheduling and broadcast scheduling. Link scheduling allows two nodes communicate to each other, while broadcast scheduling is one direction to the other. Nevertheless, the nature of broadcast-based scheduling has the ability to send data from one node to all neighbours.

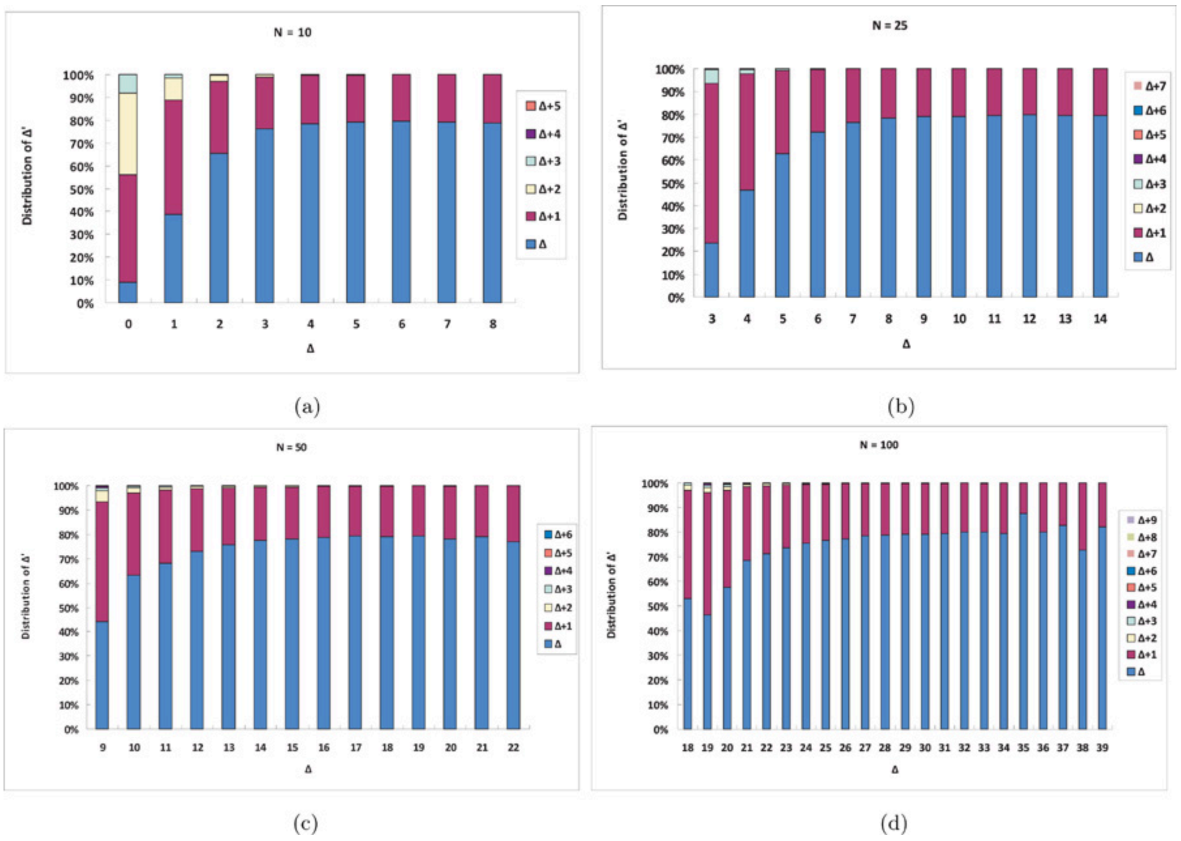
Figure 3 shows the relationship between successful rescheduling ratio with average neighbouring slots. The average neighbouring slots is the total number of time slots from the new coming node's one-hop neighbour. To avoid causing interferences, as revealed in Figure 3, average number of neighbouring slots approach to a certain limit. This phenomenon illustrates the bound of traffic loads which relates to the network's capacity. Note that larger number of nodes does not provide more neighbouring slots. In other words, to promote the performance of the slot usage, an appropriate design of system parameters, such as network average degree, occupied slot requirements, is needed.

Broadcast rescheduling has similar trends to link rescheduling. As shown in Figure 4, broadcast-based scheduling features more occupied slots of neighbours as number of nodes is small, which implies faster traffic flow in wireless networks. In the opposite, as node density is larger, fewer slots can be utilised in broadcast-scheduled networks. In both type of scheduling, higher node densities add more difficulties to satisfy its slot requirements and lead to break down of the scheduling.

## 4 Proposed local rescheduling algorithms

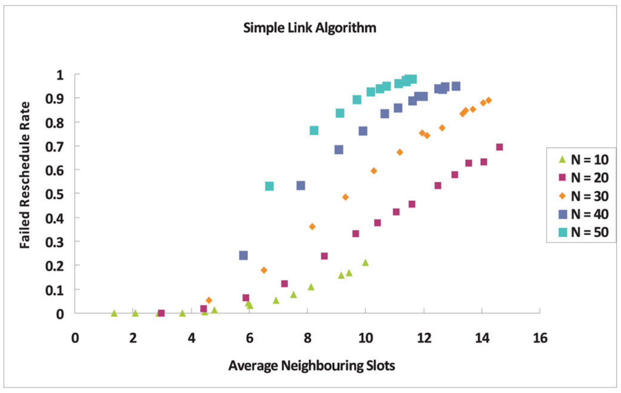
By exploiting the fact that each node may have claimed multiple time slots, the rescheduling is conducted by locally

**Figure 2** Distribution of $\Delta'$ with random deployment and arbitrary insertion of new node: (a) $N = 10$; $N = 25$ and (c) $N = 50$ and (d) $N = 100$



transferring time slots to the new node. In order to guarantee an interference-free schedule, nodes that are within two-hops from the new node must release the conflicted time slots after rescheduling, as stated in Definition 2; in link scheduling, links that are within 'two-hops' from the newly connected links must release the conflicted time slots after rescheduling, as stated in Definition 1.

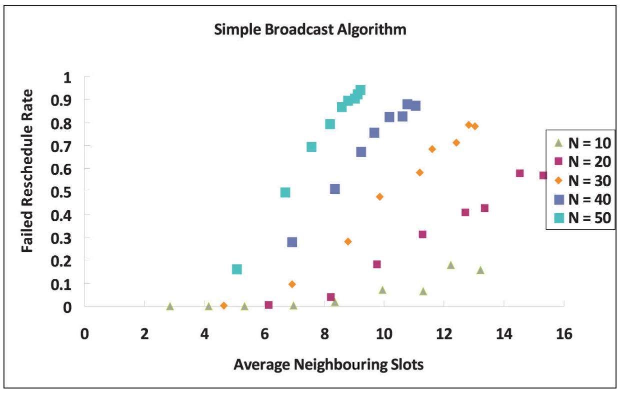
**Figure 3** Failed reschedule rate of link scheduling (see online version for colours)



## 4.1 *Local link scheduling*

Figure 5(a) illustrates the procedure of link rescheduling. The black vertices represent the scheduled nodes in the network; the grey vertex is the node new to the network. The solid lines show the links in the network, and the dotted lines indicate the links that connect the new node with its adjacent nodes. The goal of rescheduling is to assign time slots to all of the dotted lines.

**Figure 4** Failed reschedule rate of broadcast scheduling (see online version for colours)



The new node first obtains the time slot information from its two-hop neighbours. The information includes the maximum slot number (i.e., the schedule length) and the time slots at each link. Based on the information, the new node counts the appearing times of each slot (denoted by $t_i$). For each slot, the node calculates the minimum number of other slots which share same link (denoted by $m_i$). The node then sorts $t_i$, and for the slots with identical appearing times, the node reversely sorts $m_i$. Table 1 shows the result. The time slots with $m_i = 0$

are neglected. The node then sequentially selects four time slots listed in the table one after one.

**Figure 5** (a) link rescheduling and (b) broadcast rescheduling (see online version for colours)

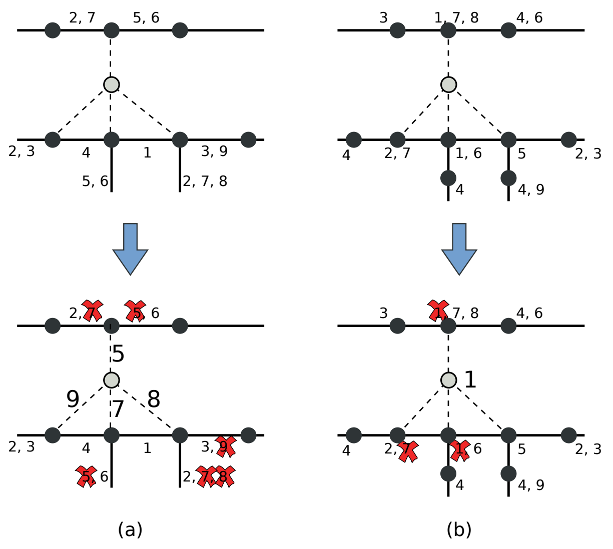


**Table 1** Local link rescheduling

| *Slot no.* | *8* | *9* | *3* | *5* | *6* | *7* | *2* |
|---|---|---|---|---|---|---|---|
| $t_i$ | 1 | 1 | 2 | 2 | 2 | 2 | 3 |
| $m_i$ | 2 | 1 | 1 | 1 | 1 | 1 | 1 |

The node will update the corresponding entry of $m_i$ after each of the selections. This procedure helps to maintain the network connectivity, for it prevents a link from releasing all its slots. In the example, the node first picks time slots #8 and #9. In the third selection the node will choose slot #5 instead of slot #3, and in the fourth selection the node will pick slot #7 instead of slot #6. Finally, the node randomly assigns the four slots to each of the dotted links and requests the previous slot holders to release the slots. The procedure is summarised in Algorithm 1. The first sort in LLR affects the least number of links, and the second sort looks forward to a balanced time slot distribution.

An alternative of local link rescheduling could reduce the required number of slot transfers, thus causing less slot conflicts. Consider the set which contains all links one-hop from a new node. The set can be partitioned in a way that any two of the partitions are either disconnected, or connected by links outside the set. Any node of a partition cannot deliver to nodes in other partitions. In other words, the nodes from different partitions are not connected. For example, the network in Figure 5(a) includes two partitions, including the upper partition and the lower partition. Before the new grey node inserts, nodes in upper partition cannot communicate with nodes in lower partition by any means. The alternative solution to link rescheduling is to share only one single slot for connecting each partition. The local rescheduling will only need to transfer slots #8 and #9, connecting each of the partitions to the new node, respectively. The procedure is summarised in Algorithm 2.

**Algorithm 1** Local Link Rescheduling Algorithm (**LLR**)

```
1:  get slot info. of each link within two hops;
2:  assign slots from the outcome of SLR;
3:  if no proper solution is found then
4:    sort slot no. based on $t_i$;
5:    for slots having identical $t_i$ do
6:      reversely sort slot no. based on $m_i$;
7:    end for
8:    while number of assigned slots $< \Delta'$ do
9:      if $m_i > 0$ then
10:       pick a slot greedily;
11:     end if
12:   end while
13: end if
14: broadcast the result;
```

**Algorithm 2** LLR Enhancement (**LLRE**)

```
1:  get slot info. of each link within two hops;
2:  assign slots from the outcome of SLR;
3:  if no proper solution is found then
4:    sort slot no. based on $t_i$;
5:    for slots having identical $t_i$ do
6:      reversely sort slot no. based on $m_i$;
7:    end for
8:    while number of assigned slots $<$ number of
      partitions do
9:      if $m_i > 0$ then
10:       pick a slot greedily;
11:     end if
12:   end while
13: end if
14: broadcast the result;
```

In this way, although the new node may not be able to directly communicate with some of its adjacent nodes, the node could ask those nodes of the same partition to relay the message and the corresponding delay would be small: any pair of nodes in a partition must be within two hops from each other. A simple mathematical proof is given as follows. From the viewpoint of the new node, any two of the partitions will be separated by at least $\frac{\pi}{3}$ degree. Suppose there are more than two partitions. Then each partition will dwell in a region limited by a central angle smaller than $2\pi - 2 \cdot \frac{\pi}{3} = \frac{4}{3}\pi$, and the diameter of the area must be shorter than or equal to $\sqrt{1^2 + 1^2 - 2\cos\frac{4}{3}\pi} = 1.732$. Therefore, any pair of nodes in a partition must be within two hops from each other.

## 4.2 *Local broadcast scheduling*

Compared with link rescheduling, broadcast rescheduling has an additional constraint. Suppose the new node is adjacent to several nodes which all had one identical time slot. In this case, all but one of the nodes must release the slot; otherwise, a primary interference will occur at the new node. The interference occurs because all nodes within the radio coverage of the broadcasting node are likely to be receivers.

The algorithm of local broadcast rescheduling is shown in Algorithm 3. Figure 5(b) provides an example. The new grey node first obtains the time slot information regarding the nodes within its two hops. The information includes the schedule length of each node and the slots assigned to each node. In this way, the new node will be able to identify the nodes which may cause primary interferences. For example, slots #1 and #7 may cause primary interferences at the new node, since they were assigned to multiple nodes which are all adjacent to the new vertex.

**Algorithm 3** Local Broadcast Rescheduling Algorithm (**LBR**)

```
1: get slot info. of each link within two hops;
2: resolve primary interference(s);
3: assign slots from the outcome of SBR;
4: if no proper solution is found then
5:     sort slot no. based on $t_i$;
6:     for slots having identical $t_i$ do
7:         reversely sort slot no. based on $m_i$;
8:     end for
9:     while number of assigned slots < number of slot
       request do
10:        if $m_i > 0$ then
11:            pick a slot greedily;
12:        end if
13:    end while
14: end if
15: broadcast the result;
```

The new node resolves the interferences by determining which node(s) must release the slot(s). In the example, the new node will issue two requests:

- the node in the middle of the upper subnetwork must release slot #1
- the second node from the left in the lower subnetwork must release slot #7.

The slot assignment ends when the slot request from the inserted node is satisfied. The remaining steps of rescheduling are similar to Algorithm 1. Table 2 lists the result of the second sort. Finally, the new node sequentially selects slot #1, the slot in the leftmost column, and then broadcast the result of new schedule.

**Table 2** Local broadcast rescheduling

| *Slot no.* | *1* | *7* | *8* | *9* | *5* | *6* | *2* | *3* | *4* |
|---|---|---|---|---|---|---|---|---|---|
| $t_i$ | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 2 | 4 |
| $m_i$ | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 |

## 5 Discussion

### 5.1 Local rescheduling superiority

In this section, assume plenty of time slots are available and are prepared right after the network initialisation. To indicate the improvement from the idea of local rescheduling, a problem is raised: *how many time slots should be prepared to guarantee a 100% collision-free scheduling after node insertion with/without local rescheduling?*

Without local rescheduling, Appendix of this thesis proves that $5(\Delta + 1)$ is the tight upper bound of the degree of the new node, and five is the tight upper bound of the number of partitions in 2D networks; $12(\Delta + 1)$ is the tight upper bound of the degree of the new node, and 12 is the tight upper bound of the number of partitions in 3D networks.

According to the given time schedule, local rescheduling can be divided into three cases. In first case, there are some time slots which were not taken by any node within two hops of the new node, and the new node can be scheduled solely based on these time slots. In this case, no node has to release any slots. SLR and SBR found these slots as solutions to a make a new schedule, while LLR, LLRE, and LBR no doubt produce proper solutions. In the second case of local rescheduling, a proper solution requires at least one slot to be transferred to the new node. All of the examples in the previous sections belong to this case. SLR and SBR fail to find available slots for new vertex without doing slot transfer. LLR, LLRE, and LBR discover these slots by transferring them to the new node without breaking the network's connectivity.

In the worst case of local rescheduling, no algorithms could achieve a proper solution without breaking network linkage. Consider Figure 5(a), for example. Suppose the schedule length is eight slots (thus the rightmost link in the subnetwork does not hold slot #9). Then any local rescheduling method will inevitably break some network links if the goal is to schedule all of the four dotted links. The worst case of local rescheduling occurs if and only if some nodes have to release all of their time slots in order to resolve slot conflicts. In other words, a proper solution of local rescheduling exists if and only if the total number of time slots at each node is more than the number of slots that a node might have to release.

With local link rescheduling, since each slot transfer only conflicts those slots of the same type, the maximum number of slots that a node might have to release is equal to the total number of slots transferred to a new node. In LLR, the number of slot transfers is equal to the degree of the new node; in LLRE, the number of transfers is reduced to the number of partitions, which is at most five partitions in 2D networks, as illustrated in Appendix. The argument can be summarised as follows:

**Lemma 4:** *In 2D (3D) networks, a* ***proper*** *solution of local link rescheduling can be obtained if each link had claimed more than* ***five (twelve)*** *time slots, respectively.*

In other words, in link scheduling, only (**five**, **twelve**) slots are needed using local rescheduling method compared with $(5(\Delta + 1), 12(\Delta + 1))$ slots in traditional scheduling method in 2D (3D) networks, respectively.

With local broadcast rescheduling, as long as a single time slot transferred to the new node, some other slots may need to be released in order to resolve primary interferences. By the definition of primary interference, only those nodes adjacent to the new node will possibly cause interferences at it. According to the broadcast scheduling criteria in Definition 2, once a node obtained a time slot, all of its one-hop and two-hop neighbouring nodes may not have the same slot. Therefore, the maximum number of nodes producing a primary interference at the new node will be five, which is equal to the size of a maximum independent node set within the transmission radius of the new node. If every node in the network has more than five slots, it will be feasible to resolve any number of primary interference occurring at the new node. A solution can be obtained by requesting each of the five nodes to retain a distinct slot and to release all the other slots.

**Lemma 5:** *In 2D (3D) networks, a **proper** solution of local broadcast rescheduling can be obtained if each node had claimed more than **five** (**twelve**) time slots, respectively.*

According to the discussions from Lemma 4 and Lemma 5, the induction can be concluded as:

**Theorem 6:** *A **proper** solution of local rescheduling can be obtained if each node/link had claimed more than **five** slots in two-dimension networks and **twelve** slots in three-dimension networks.*

To sum up, in both link and broadcast scheduling, only (**five**, **twelve**) slots are needed using local rescheduling method, where $(5(\Delta+1), 12(\Delta+1))$ slots are used in traditional scheduling method in 2D (3D) environments.

## 5.2 *Local rescheduling overhead*

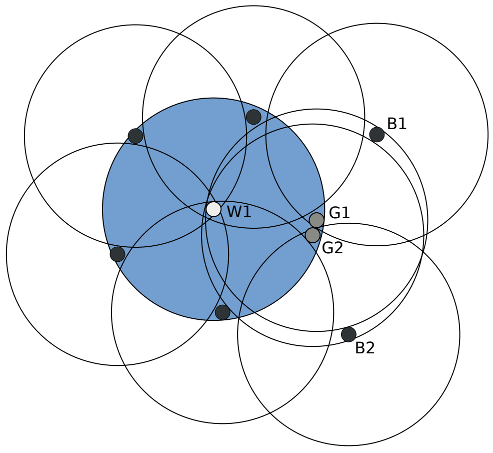
One way to look at the overhead of rescheduling is to count the total number of time slots that have to be released from the network. Since each time slot may be assigned to multiple nodes and each node may possess several time slots; and each slot transfer is likely to have conflicts at multiple nodes and each node possibly suffers from interferences of several slots. All of the above collisions contribute to the same kind of overhead. In LLR, each time slot of the new coming node results in at most 5 slot transfers after insertion. Consider the interference from 1-hop and 2-hop neighbours, a loose bound of maximum number of conflicts of LLR is $5 \cdot 5\Delta \cdot 5(\Delta - 1) \simeq 125\Delta^2$. Similarly, LLRE generates at most $5 \cdot 5 \cdot 5(\Delta - 1) \simeq 125\Delta$ conflicts.

For the case of broadcast rescheduling, Figure 6 points out that each one of the slot transfers might lead to more than five conflicts, which is quite different from link scheduling environments. The circles indicate the transmission radius of each node. The node $W1$ represents the newly added node. Node $B1$ is two-hops away from node $B2$ (connected via nodes $G1$ and $G2$), and each of them is two-hops away from the other black nodes. In this case, all of the six black nodes might have slots identical to what was transferred to $W1$.

## 5.3 *Alternative solution*

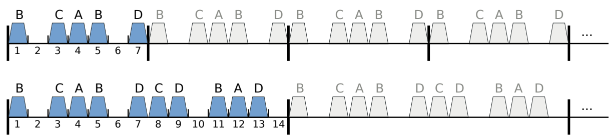
In 2D environments, a simple way to satisfy the condition is to rerun a scheduling algorithm five times before the network starts. In other words, the new cycle length is five times longer and the generated schedules are combined and concatenated. In this way, each node/link is guaranteed to possess more than five slots. As to the case of 3D networks, 12 repetitions of rerunning scheduling algorithm will be sufficient. The repetitions will produce enough time slots for rescheduling algorithms LLRE and LBR to produce proper solutions.

**Figure 6** Black nodes might have identical slots in broadcast rescheduling (see online version for colours)



The idea of repeating a schedule does not cause noticeable performance penalties. The repetitions prolong the length of schedule cycle, not necessarily the delay of transmission. Figure 7 is a schedule of a new node. Nodes $A$, $B$, $C$, and $D$ are the neighbours of the new vertex. The labels above are the assigned time slot which indicates the corresponding node. The label below is the slot number of the arranged schedule. The lower line shows the other schedule obtained by running a scheduling algorithm twice. After combining two different schedules, the average waiting time for adjacent nodes $B$ and $A$ grows longer, whilst the average waiting time for Nodes $C$ and $D$ are shorter.

**Figure 7** Example of two transmission schedules of the new node (see online version for colours)


B C A B D B C A B D B C A B D B C A B D ...
1 2 3 4 5 6 7
B C A B D C D B A D B C A B D C D B A D ...
1 2 3 4 5 6 7 8 9 10 11 12 13 14

The repetition of initial scheduling might require a new node have a longer delay in transmission. For example, in the upper schedule, suppose the time slot #7 in Figure 7 was deleted and transferred to the new node. The new node has to wait 13 time slots before it can transmit another message to node $D$. In most cases, to find a proper solution, it is adequate to repeat the scheduling scheme less than two times. In each of the repetitions the link/node may claim multiple time slots, thus reducing the required number of repetitions.

## 6 Simulations

### *6.1 Simulation setup*

Experiments have been evaluated with varying numbers of slot requests from the new node, cycle length, and the number of occupied slot in original schedule accordingly. All experiments are tested and evaluated by simulator C. Nodes are randomly deployed in a two-dimensional plane and the total number is ranged from 10 to 50. Each setting is tested with 1000 random network topologies. For each topology, 1000 times of random-insertions are performed, and the results are obtained in average. The sizes of the maps are fixed to 1000x1000 and communication range is set to 250. Due to space limitation, only the representative two-dimensional cases are shown in the literature.

### *6.2 Simulation results*

#### *6.2.1 Occupied slots*

The number of occupied slots is the bound that one node can possess. Nodes may have maximum number of slots or fewer slots consider the capacity and traffic loads of the networks. During network constructions, each node maximises its own slots without causing interferences. After the process of network building, a new node will be inserted and failed reschedule ratio is evaluated with different occupied slots limitation.

The slot requests from the new node are set to be the same as the number of occupied slots during the tests. In Figure 8, as the number of node increase, the existing rescheduling algorithm is getting harder finishing rescheduling. Meanwhile, SLR is difficult to find proper solutions when occupied slots are massive, which means the traffic load is heavier. The evaluation of SBR, which is similar to SLR, is not listed here due to limited space. Figure 9 shows the comparison between LLR and SLR. LLR outperforms SLR in spite of the number occupied slots (denoted as OS) equals 1. LLR utilises slot transfer as a method to improve the possibility of successful rescheduling. LLR algorithm reduces to SLR as OS = 1, for LLR cannot do any slot transfer if a link has only one available slot. Any transfer results in link fracture and no proper solutions can be found. Different with SLR, LLR reduces its failed rate as the traffic is more congest, which is benefit from the slot transfer scheme that balanced the traffic load.

Figure 10 plots the comparison between LBR and SBR. In broadcast scheduling, LBR is nearly 100% ‘reschedulable’ when OS $\geq$ 2. It implies an empirical observation that if a pre-scheduled network with random deployment of node and each node has at least two available slots, it can be fairly expected to find a proper solution without restarting schedule process globally.

To be representative and to distinguish the performance of the proposed algorithms from SLR and SBR, the slot requests and the occupied slots are set to be 2 without especially mentioned in the following tests.

**Figure 8** Failed rescheduling ratio with distinct occupied slots (see online version for colours)

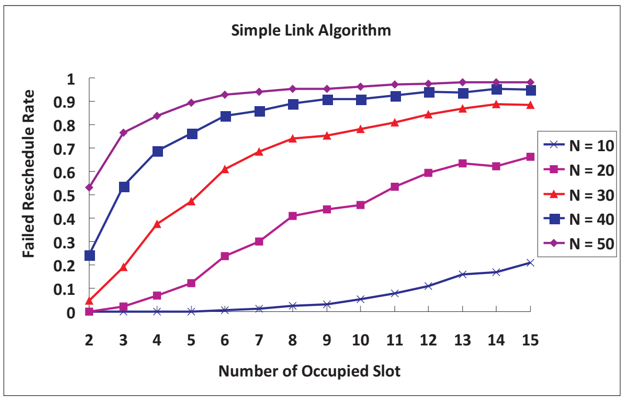
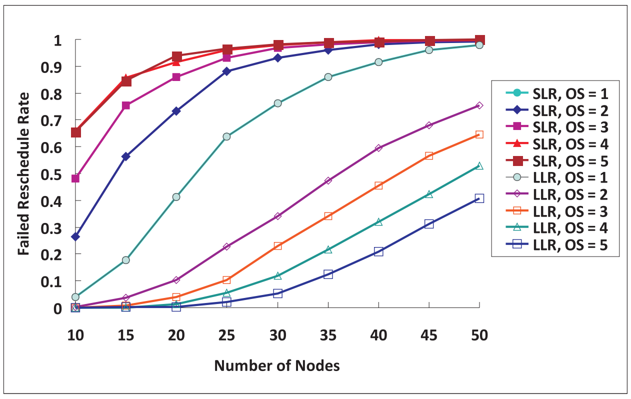


**Figure 9** LLR and SLR with distinct occupied slots (see online version for colours)



#### *6.2.2 Cycle Length*

Figure 11 examines the failed reschedule rate of SLR after node insertion. Failed reschedule rate decreases as the cycle length increases, since longer cycles provide rooms for more slot transfers. A higher number of deployed node yields more interferences and lead to high level of failed rate. Figures 12 and 13 plot the influences of variance of cycle lengths between LLR and SLR. SLR fails to reschedule for no vacancy of time slots as the cycle length is short. LLR mitigates the effects of having shorter cycle in existing scheduling. Figures 14 and 15 give an observation to the broadcast scheduling. It can be seen that LBR has above 99% of chances finding proper solutions, while SBR fails to reschedule much more often as network density is congest.

**Figure 10** LBR and SBR with distinct occupied slots (see online version for colours)

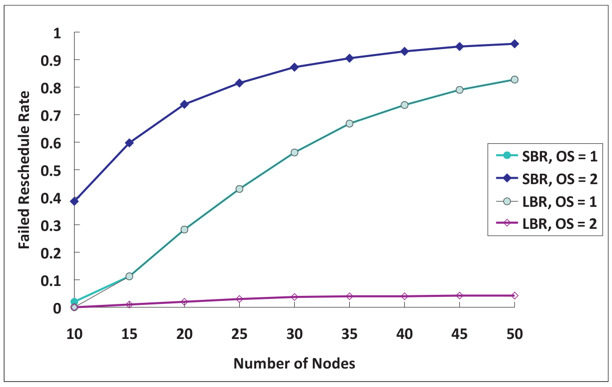


**Figure 11** Failed rate w.r.t. cycle lengths (see online version for colours)

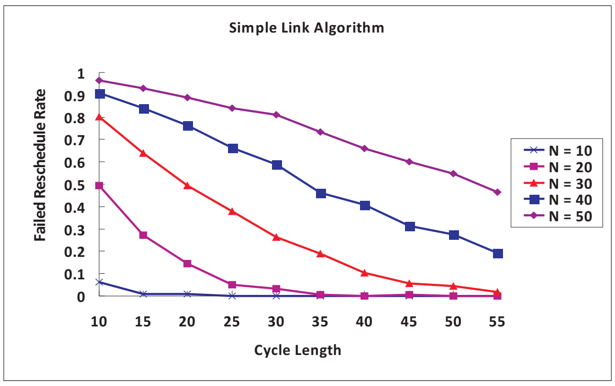
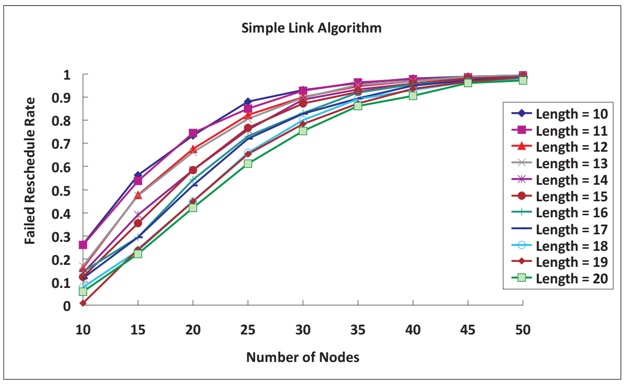


**Figure 12** SLR with distinct cycle lengths (see online version for colours)



### 6.2.3 *Slot request*

To study the effect of slot request from new node, experiments were tested with distinct number of slot requests from new vertex. Figure 16 depicts that the failed reschedule rate has positive proportion to the increasing number of slot requests. Besides, the more occupied slots of each node possess, the higher rate of reschedule fails. Larger number of nodes in the networks stands for more neighbours and hence more collisions that may potentially happen, as shown in Figure 17. Apparently, along with network density, the number of slot requests has deep impact on the performance of SLR. Nevertheless, LLR overcomes the difficulties and offers outstanding improvements over SLR. By transferring the slots and avoiding interferences, as illustrated in Figure 18, the successfully reschedule rate is up to 95% in link scheduling.

**Figure 13** LLR with distinct cycle lengths (see online version for colours)

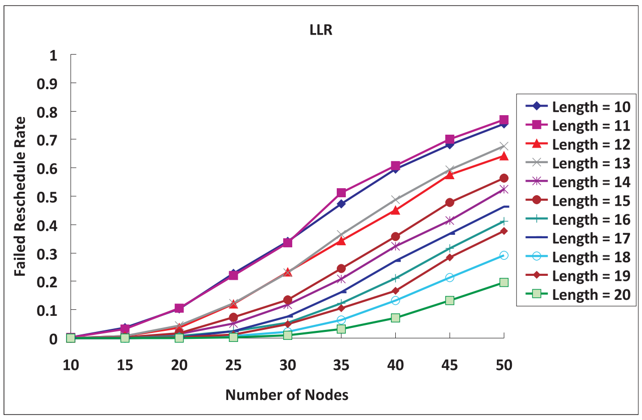
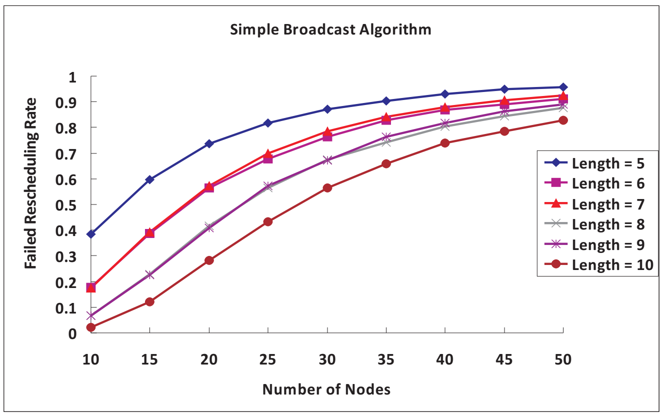


**Figure 14** SBR with distinct cycle lengths (see online version for colours)



In broadcast scheduling, the growing number of slot requests from new vertex degrades the performance of SBR sharply as in link scheduling. Figures 19 and 20 represent the comparison between LBR and SBR. LBR has better reschedule rate compared with SBR. LBR shows superior performance when number of slot request is low. Even with more demands, the proposed scheme decreases the failed rate effectively.

**Figure 15** LBR with distinct cycle lengths (see online version for colours)

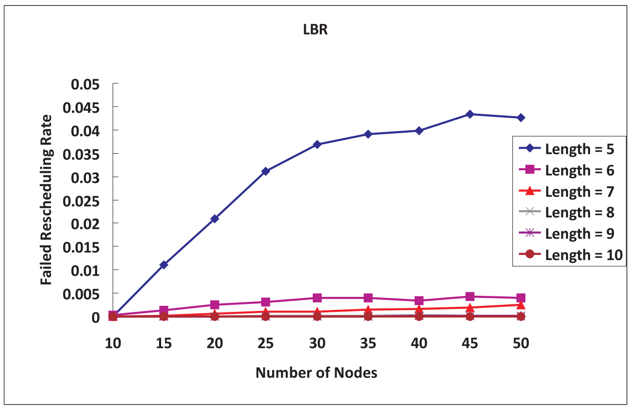


**Figure 16** Failed rescheduling ratio with distinct slot requests (see online version for colours)

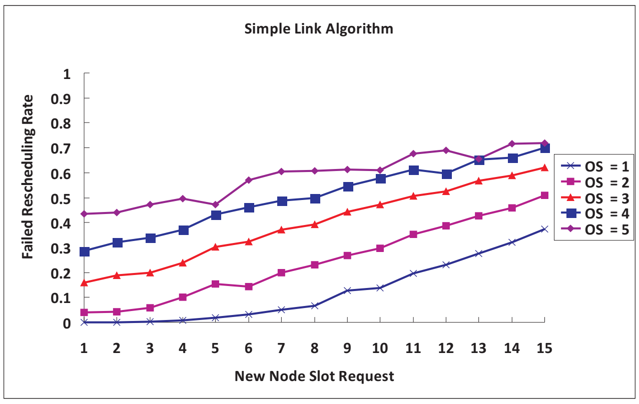
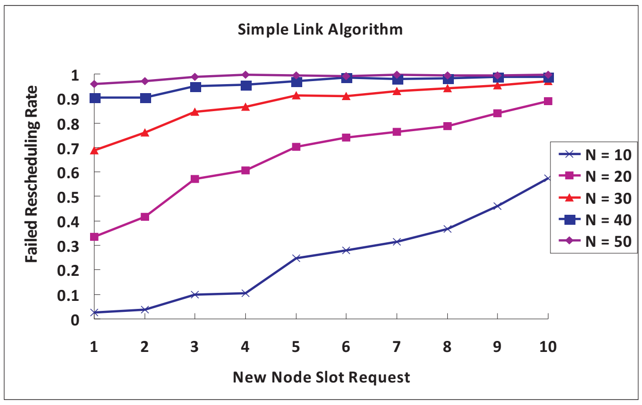


**Figure 17** SLR with distinct slot requests and number of nodes (see online version for colours)



## *6.3 Link rescheduling algorithm enhancement*

Algorithm 2 is the enhancement of the link rescheduling algorithm LLR. In order to further decrease the number of interference from neighbours, slot transfers are done only in different partitions. It is proved in this thesis that only 5 partitions among neighbours for a single node. In other words, at most 5 slots are needed to be transferred if collisions happen to the new vertex. Figure 21 shows the improvement from LLRE. With the current setting, LLRE has almost 100% reschedule rate as $OS \geq 2$. LLRE reduces the number of slot transfer and hence diminishes the impact of interference. Figures 22 and 23 tell the refinement from LLRE in distinct cycle lengths. In addition, due to fewer collisions, LLRE has better reschedule rate compared with LLR, as shown in Figures 24 and 25.

**Figure 18** LLR with distinct slot requests and number of nodes (see online version for colours)

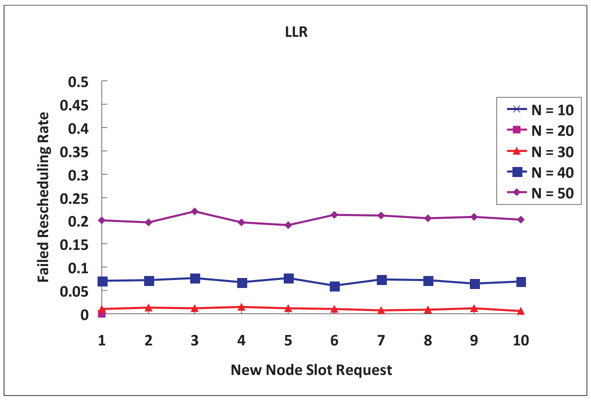


**Figure 19** SBR with distinct slot requests (see online version for colours)

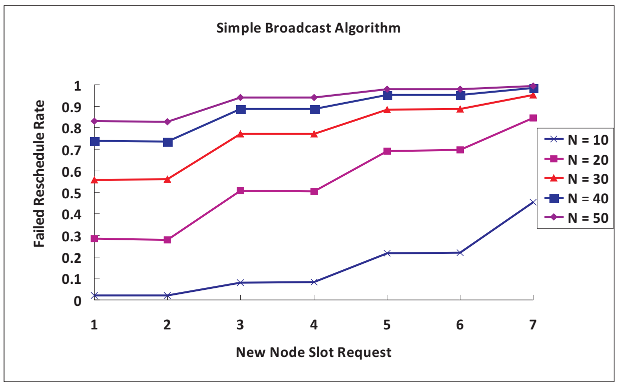


**Figure 20** LBR with distinct slot requests (see online version for colours)

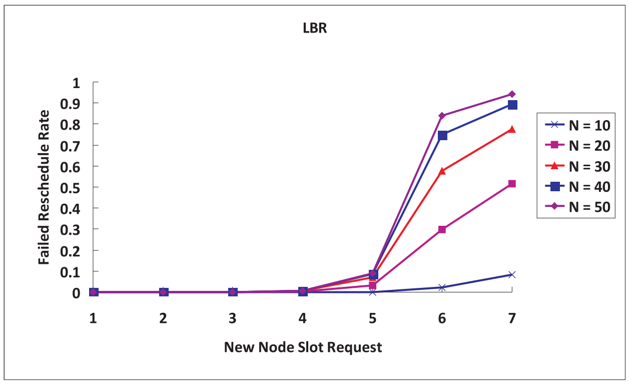


**Figure 21** LLR and LLRE with distinct occupied slots (see online version for colours)

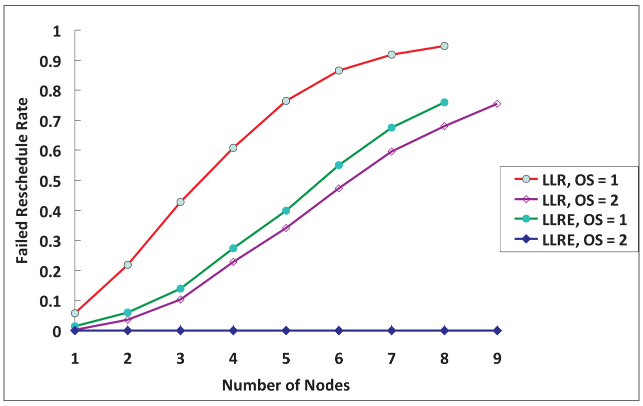


**Figure 22** LLR with distinct cycle lengths (see online version for colours)

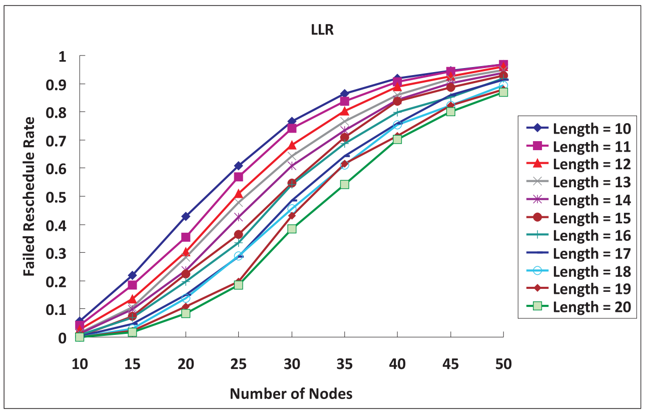


**Figure 23** LLRE with distinct cycle lengths (see online version for colours)

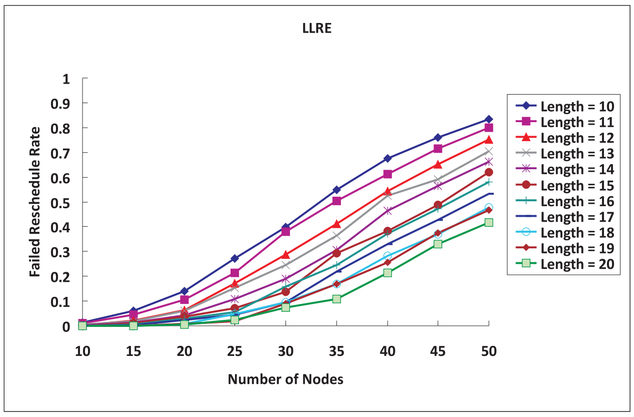


**Figure 24** LLR with distinct slot requests (see online version for colours)

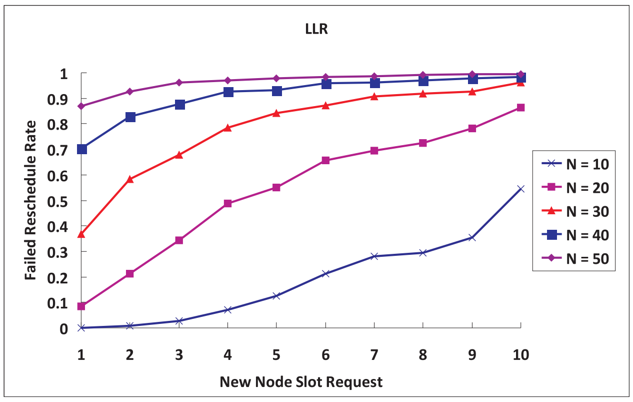
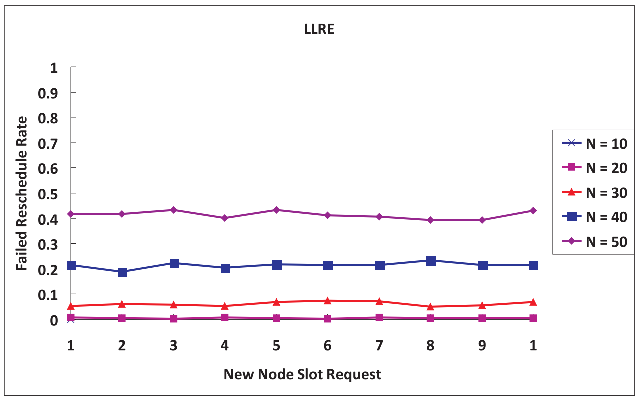


**Figure 25** LLRE with distinct slot requests (see online version for colours)



## 7 Conclusion

The collisions of packet transmissions in mobile sensor networks have been one of the main topics in MAC/Protocol layers. To diminish the degradation of packet transmission successful rate, the problems of local rescheduling in link rescheduling and broadcast scheduling in WSNs are investigated in this paper. The rescheduling is conducted by locally transferring time slots to the new node. Several efficient schemes have been proposed, LLR and LLRE are link rescheduling solutions where LBR is the solution to broadcast rescheduling. These proposed schemes manage to overcome the shortcomings of existing simple rescheduling algorithms, SLR and SBR. A theoretical bound of maximum network degree after node insertion is given.

This paper examines the bound of maximum degree under node introduction. The bound is given as the function of the current maximum node degree. Given the condition that each node have the identical radius, the results are conducted in both two-dimensional and three-dimensional spaces. On the other hand, the maximum degree is inherently bounded by the total number of nodes. Analysis shows that its tightness depends on both the number of nodes and the ratio of the radius of node and space. Simulations validate the bounds in random network topologies.

The theoretical bound happens rarely after numerous simulations. Hence, the real-world experiments with random injection of nodes are surveyed to study the effects on local rescheduling. Both LLR and LBR outperform existing schemes SLR and SBR with respect to varied numbers of nodes, occupied slots, cycle lengths and slot requests. LLR and LBR achieve higher successful rate of finding *proper* solutions, despite its simplicity. With small overhead, LLRE further enhances the performance of LLR by transferring slots to partitions among neighbours of the new node.

## References

Ahn, G.S., Hong, S.G., Miluzzo, E., Campbell, A.T. and Cuomo, F. (2006) ‘Funneling-MAC: a localized, sink-oriented MAC for boosting fidelity in sensor networks’, *International Conference on Embedded Networked Sensor Systems*, Boulder, Colorado, USA, pp.293–306.

Ammar, M.H. and Stevens, D.S. (1991) ‘A distributed TDMA rescheduling procedure for mobile packet radio networks’, *IEEE International Conference on Communications*, Vol. 3, pp.1609–1613.

Anstreicher, K.M. (2004) ‘The thirteen spheres: a new proof’, *Discrete and Computational Geometry*, Vol. 31, pp.613–625.

Arikan, E. (1984) ‘Some complexity results about packet radio networks’, *IEEE Transactions on Information Theory*, Vol. 30, pp.681–685.

Chen, A., Kumar, S. and Lai, T.H. (2007) ‘Designing localized algorithms for barrier coverage’, *ACM International Conference on Mobile Computing and Networking*, Montreal, QC, Canada, pp.63–74.

Chen, Y.S., Lin, Y.W. and Lee, S.L. (2011) ‘A mobicast routing protocol in underwater sensor networks’, *IEEE Wireless Communications and Networking Conference*, Cancun, Quintana Roo, Mexico, pp.510–515.

Conway, J.H. and Sloane, N. J.A. (1999) *Sphere Packings, Lattices and Groups*, 3rd ed., Springer.

Dhurandher, S.K., Obaidat, M.S. and Gupta, M. (2010) ‘A novel geocast technique with hole detection in underwater sensor networks’, *IEEE/ACS International Conference on Computer Systems and Applications*, Hammamet, Tunisia, pp.1–8.

Doudou, M., Alaei, M., Djenouri, D., Barcelo-Ordinas, J. and Badache, N. (2013a) *Duo-MAC: Energy and Time Constrained Data Delivery MAC protocol in Wireless Sensor Networks*, pp.424–430.

Doudou, M., Djenouri, D. and Badache, N. (2013b) ‘Survey on latency issues of asynchronous MAC protocols in delay-sensitive wireless sensor networks’, *Communications Surveys Tutorials, IEEE*, Vol. 15, No. 2, pp.528–550.

Ephremides, A. and Truong, T.V. (1990) ‘Scheduling broadcasts in multihop radio networks’, *IEEE Transactions on Communications*, Vol. 38, No. 4, pp.456–460.

Even, S., Goldreich, O., Moran, S. and Tong, P. (1984) ‘On the NP-completeness of certain network testing problems’, *Networks*, Vol. 14, No. 1, pp.1–24.

Hales, T.C. (2002) ‘An overview of the Kepler conjecture’, *IEEE Communications Magazine*, Vol. 44, pp.115–121.

Hossain, E. and Bhargava, V.K. (2001) ‘A centralized TDMA-based scheme for fair bandwidth allocation in wireless IP networks’, *IEEE Journal on Selected Areas in Communications*, Vol. 19, No. 11, pp.2201–2214.

Leech, J. (1956) ‘The problem of thirteen spheres’, *The Mathematical Gazette*, Vol. 40, No. 331, pp.22–23.

Munir, S., Lin, S., Hoque, E., Nirjon, S.M.S., Stankovic, J.A. and Whitehouse, K. (2010) *Addressing Burstiness for Reliable Communication and Latency Bound Generation in Wireless Sensor Networks*, pp.303–314.

Ou, C.H. and Ssu, K.F. (2008) ‘Sensor position determination with flying anchors in three-dimensional wireless sensor networks’, *IEEE Transactions on Mobile Computing*, Vol. 7, No. 9, pp.1084–1097.

Pantelidou, A. and Ephremides, A. (2011) ‘Scheduling in wireless networks’, *Foundations and Trends in Networking*, Vol. 4, No. 4, pp.421–511.

Park, P., Fischione, C., Bonivento, A., Johansson, K. and Sangiovanni-Vincent, A. (2011) ‘Breath: an adaptive protocol for industrial control applications using wireless sensor networks’, *IEEE Transactions on Mobile Computing*, Vol. 10, No. 6, pp.821–838.

Peng, Y., Li, Z., Qiao, D. and Zhang, W. (2011) *Delay-Bounded MAC with Minimal Idle Listening for Sensor Networks*, pp.1314–1322.

Ramanathan, S. (1997) ‘A unified framework and algorithm for (T/F/C)DMA channel assignment in wireless networks’, *Joint Conference of the IEEE Computer and Communications Societies*, Kobe, Japan, Vol. 2, pp.900–907.

Ramanathan, S. and Lloyd, E.L. (1993) ‘Scheduling algorithms for multihop radio networks’, *Transactions on Networking*, Vol. 1, No. 2, pp.166–177.

Ravindran, S. and Narayanasamy, P. (2011) ‘Energy-aware face geocast for wireless adhoc and sensor networks’, *International Conference on Electronics Computer Technology*, Kanyakumari, India, Vol. 1, pp.6–10.

Rhee, I., Warrier, A., Min, J. and Xu, L. (2009) ‘DRAND: distributed randomized TDMA scheduling for wireless ad hoc networks’, *IEEE Transactions on Mobile Computing*, Vol. 8, No. 10, October, pp.1384–1396.

Schütte, K. and van der Waerden, B.L. (1953) ‘Das Problem der dreizehn Kugeln’, *Mathematische Annalen*, Vol. 125, pp.325–334.

Wang, C. and Ssu, K.F. (2010) ‘A distributed collision-free low-latency link scheduling scheme in wireless sensor networks’, *Wireless Communications and Networking Conference*, Sydney, Australia, pp.1–6.

Wang, W.Z., Wang, Y., Li, X.Y., Song, W.Z. and Frieder, O. (2006) ‘Efficient interference-aware TDMA link scheduling for static wireless networks’, *International Conference on Mobile Computing and Networking*, Los Angles, CA, USA, pp.262–273.

Yackoski, J. and Shen, C.C. (2008) 'UW-FLASHR: achieving high channel utilization in a time-based acoustic MAC protocol', *International Workshop on Underwater Networks*, San Francisco, CA, USA, pp.59–66.

Yang, C.H. and Ssu, K.F. (2008) 'An energy-efficient routing protocol in underwater sensor networks', *International Conference on Sensing Technology*, Tainan, Taiwan, pp.114–118.

Ye, W., Heidemann, J. and Estrin, D. (2002) 'An energy-efficient MAC protocol for wireless sensor networks', *International Conference on Computer Communications*, New York City, NY, USA, pp.1567–1576.

Yigitel, M.A., Incel, O.D. and Ersoy, C. (2011) 'Design and implementation of a QoS-aware {MAC} protocol for wireless multimedia sensor networks', *Computer Communications*, Vol. 34, No. 16, pp.1991–2001.

Yu, Q., Tan, C. and Zhou, H. (2007) *A Low-Latency MAC Protocol for Wireless Sensor Networks*, pp.2816–2820.

Zimmerling, M., Ferrari, F., Mottola, L., Voigt, T. and Thiele, L. (2012) *pTunes: Runtime Parameter Adaptation for Low-power MAC Protocols*, pp.173–184.

## Appendix

This section proves the maximum degree of the new node. Some definitions are given as follows. The *r-neighbourhood* of $p$ is the set $N'_r(p) = \{x \in R^k \mid 0 < d(p,x) < r\}$, where $r$ is called the *radius* of $N'_r(p)$. In $R^2$, neighbourhoods are interiors of circles; in $R^3$, neighbourhoods are interiors of spheres. The set of points $x$ satisfying $d(p,x) = r$ is called the *boundary* of the $r$-neighbourhood of $p$, or the *r-boundary* of $p$. The *closure* of $A \subset R^k$ is the smallest subset of $R^k$ which contains $A$. The closure of $N'_r(p)$, denoted by $N_r(p)$, is the union of the $r$-neighbourhood of $p$ and its $r$-boundary. Consequently, in the disk graph, the degree of the vertex $v$ at $p$ can be denoted by $|N_1(p)|$, which is the number of vertices in the closure $N_1(p)$. For simplicity, $N_1(p)$ is denoted by $N(p)$ in the following text. Let $\Delta$ be the maximum degree in the graph before vertex insertion.

In the following text, two graphs are said to be *reducible* to each other if and only if both of them have the same $\Delta$ and $|V|$. To prove the maximum of $|N(p)|$, it is sufficient to consider the graph which is reducible to all other graphs of $N(p)$. The two lemmas below suggest the existence of such graphs.

**Lemma A1:** *Let $p$ be a point in $R^k$. Any disk graph $G$ in $N(p)$ is reducible to the graph $G'$ such that $V(G') = \{v \mid d((v,u) \leq \gamma$ for all $u \in N(v)$, and $0 < \gamma < 1\}$.*

*Proof*: The proof is given by construction. Let $Q = \{q_1, q_2, ..., q_m\}$ be the independent dominating set of $G$ in $N(p)$, and choose $g1$ so that $|N(q_1)| = \Delta$. Set $0 < \gamma < 1$. For $k = 1, ..., m$, we sequentially move each vertex in $N(q_k)$ into $N_{\gamma/2}(q_k)$; denote the induced graph by $G'$. This procedure both preserves the maximum degree of $G$ and maintains the number of vertices. Hence, $G$ is reducible to $G'$. □

**Lemma A2:** *Let $p$ be a point in $R^k$. Let $G$ be any collection of independent vertices in $N(p)$. Let $G'$ be an disk graph in which all vertices are on the 1-boundary of $N(p)$. Then $G$ is reducible to $G'$.*

*Proof*: From $G$, we construct $G'$ by pushing each $v \in V$ toward the direction $\overrightarrow{pv}$ till it reaches the 1-boundary of $N(p)$. Clearly, each pair of vertices are connected in $G'$ only if they were connected in $G$. Therefore, $V(G')$ is also an independent set, and the proof follows. □

Lemma 7 implies that every disk graph is reducible to a collection of independent, complete subgraphs. For example, let $Q$ be the independent dominating set of any graph $G$ in $N(p)$. Suppose $d(q_1, q_2) > \alpha \geq 1$ for any $q_1, q_2 \in Q$. Let $\gamma = r(\alpha) < (\alpha - 1)/2$. Due to that $r(\alpha) < \alpha - 1$, $G$ is reducible to a graph where each vertex is belonged to exactly one of $N(q)$ for $q \in Q$; due to that $r(\alpha) < 1$, each subgraph is a complete graph. Finally, since $\gamma = r(\alpha) < (\alpha - 1)/2$, any two subgraphs are independent.

**Proposition A1:** *Every disk graph is reducible to a collection of independent, complete subgraphs.*

Therefore,

$$\max |N(p)| = M(\Delta + 1), \tag{A1}$$

where $M$ is the maximum number of independent, complete subgraphs in $N(p)$. Let $K$ be the maximum number of independent vertices in $N(p)$. It is easy to see that $M \leq K$. And $M = K$ due to Proposition 9. According to Lemma 8, $K$ can be determined by the case that all vertices are on the 1-boundary of $p$. In this case, $K = 5$ in $R^2$, clearly; the kissing number problem implies that $K \leq 12$ in $R^3$. As a result,

$$\max |N(p)| = 5(\Delta + 1) \text{ in } R^2, \tag{A2}$$

and

$$\max |N(p)| \leq 12(\Delta + 1) \text{ in } R^3. \tag{A3}$$

Thus the following theorems.

**Theorem A1:** *In any two-dimensional disk graph, after one vertex insertion, the degree of the new vertex is at most equal to* $5(\Delta + 1)$.

**Theorem A2:** *In any three-dimensional disk graph, after one vertex insertion, the degree of the new vertex is at most equal to* $12(\Delta + 1)$.

*Proof*: In the kissing number problem, at most 12 balls can simultaneously touch a sphere of the same radius. Suppose the 12 balls touch the sphere in a way that the contact points together form a regular icosahedron circumscribing the sphere. The edge length of the icosahedron, denoted by $l_p$, will be $l_p = 4(\sqrt{10 + 2\sqrt{5}})^{-1} \approx 1.0514 > 1$. That is, the 12 contact points form an independent set. Thus, the equality of equation (A3) holds. □

As a result, in two-dimensional space, the inserted vertex has at most $5(\Delta + 1)$ adjacent vertices. Let $C$ be a vertex configuration where all the adjacent vertices are divided into five independent groups, and the size of each group is $\Delta + 1$. We prove that $C$ is not reducible to any other vertex configuration: On one hand, each complete subgraph can have at most $\Delta + 1$ vertices, and therefore $C$ cannot be reduced to five or less independent groups. On the other hand, it is impossible to create six or more independent groups in $N(p)$. Therefore, $C$ is the only possible vertex configuration with $5(\Delta + 1)$ vertices.

The similar reasoning concludes that, in three-dimensional space, if there are $12(\Delta + 1)$ vertices adjacent to the inserted vertex, what follows is the only possible vertex configuration: all the adjacent vertices are divided into 12 independent groups, and the size of each group is $\Delta + 1$.

**Theorem A3:** *In a two-dimensional, connected disk graph, the degree of the new vertex after insertion is at most equal to* $5\Delta$*; the degree is at most equal to* $12\Delta$ *in a three-dimensional, connected disk graph.*

*Proof*: Starting from the maximum vertex configuration in $N(p)$, we prove the theorem by connecting each of the independent, complete subgraph in a way that removes the least number of vertices from $N(p)$. There are three approaches to connect the subgraphs: 1. Connect the subgraphs to a connected subgraph external to $N(p)$; 2. Connect the subgraphs by placing vertices in $N(p)$; 3. Connect some of the subgraphs by the second approach, and then connect the rest by the first approach.

In the first approach, it is convenient to assume a connected subgraph, named $E$, which surrounds $N(p)$ but does not have connection to any of the vertices in $N(p)$. We can connect $E$ to a subgraph $S$ in $N(p)$ by placing a set of vertices $v_1, v_2, ..., v_i$ between $N(p)$ and $E$ such that $d(S, v_1) \leq 1$ and $d(v_i, E) \leq 1$. Recall that $S$ is complete and has $\Delta + 1$ vertices; therefore, after connecting to $E$, the subgraph $S$ must remove at least one vertex to make the degrees of vertices in $S$ not exceed $\Delta$. As a result, each connection causes $|N(p)|$ to decrease by one.

In the second approach, first we show that placing vertex $v$ anywhere in $N(p)$ will make $N(v)$ include all vertices of at least one complete subgraph. Let $v_1$ be a vertex in subgraph $S_1$ and $v_2$ be a vertex in subgraph $S_2$, respectively, such that $d(v_1, v_2) \geq d(v_1', v_2')$ for any $v_1' \in S_1$ and $v_2' \in S_2$. By definition, $d(v_1, v_2) > 1$ and $\angle v_1 p v_2 > \pi/3$. Due to that there are five independent subgraphs, we can choose $S_1$ and $S_2$ such that $\angle v_1 p v_2 \leq 2\pi - 4(\pi/3) = 2\pi/3$. Therefore, $d(v_1, v_2) < 2$. Since the diameter of $N(v)$ is two, $N(v)$ will completely include either $S_1$ or $S_2$.

Let $S_1$ be the subgraph which is completely included in $N(v)$, and let $S_2$ be an arbitrary subgraph which is partially included in $N(v)$. Let $S_2a$ be the subgraph of $S_2$ inside $N(v)$ and $S_2b$ be the subgraph of $S_2$ outside $N(v)$. Suppose $|S_2a| = m$. Then $N(v)$ includes at least $(\Delta + 1) + m$ vertices.

For the case of $m = 1$, we have to remove at least two vertices in $N(p)$ in order not to make the degree of $v$ exceed $\Delta$. The two removals must come from $S_1$. To keep the degrees of vertices in $S_2$ not exceed $\Delta$, we have to remove at least one vertex from $S_2b$. As a result, placing one vertex in $N(p)$ to connect two subgraphs causes at least three vertex removals, and $|N(p)|$ decreases by at least two. For the case of $m \geq 2$, we have to remove at least $m + 1$ vertices in $N(p)$, and thus $|N(p)|$ decreases by at least $m$. Compared with the first connecting approach, the second approach does not give a larger $|N(p)|$.

Note the degree of vertices in $S_2b$, after connection, must be smaller than $\Delta$. Therefore, we can connect $S_2b$ to a path external to $N(p)$, and the procedure does not cause any vertex removal in $N(p)$.

Suppose that placing a vertex in $N(p)$ connects three subgraphs. The third subgraph has to remove at least one vertex. Therefore, in the sense of the total number of vertex removals in $N(p)$, the above situation is equivalent to the following method: Connect two subgraphs by placing a vertex in $N(p)$, and then connect the result to the third subgraph with a path external to $N(p)$. In general, to connect each additional subgraph, at least one vertex must be removed from the additional subgraph. By induction, the second approach does not give a larger $|N(p)|$. $\square$