國立交通大學

網路工程研究所

碩士論文

在專用控制通道技術架構下使用時間基準之多重通道無線網狀網路傳輸媒介存取控制協定

TBM: Time-Based Multi-Channel MAC Protocol for Wireless Mesh Networks under Dedicated Control Channel Approach

研究生：鄭宇翔

指導教授：簡榮宏　教授

中華民國　九十七　年七月

在專用控制通道技術架構下使用時間基準之多重通道無線網狀網路傳輸媒介存取控制協定

TBM: Time-Based Multi-Channel MAC Protocol for Wireless Mesh Networks under Dedicated Control Channel Approach

研 究 生：鄭宇翔　　Student：Yu-Hsiang Cheng

指導教授：簡榮宏　　Advisor：Rong-Hong Jan

國 立 交 通 大 學

網 路 工 程 研 究 所

碩 士 論 文

A Thesis

Submitted to Institute of Network Engineering

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science

July 2008

Hsinchu, Taiwan, Republic of China

中華民國九十七年七月

在專用控制通道技術架構下使用時間基準之多重通道無線網狀網路傳輸媒介存取控制協定

研究生：鄭宇翔　　指導教授：簡榮宏 博士

國立交通大學資訊科學與工程研究所

摘　要

在使用多重通道的無線網狀網路中，以專用控制通道技術為基礎的網路架構可解決目前大部份多重通道上的問題，如多重通道隱藏主機、失聰與廣播支援等問題。然而在專用控制通道架構下有兩個挑戰：一個是控制通道瓶頸問題，另一個則是通道動態選擇問題。本篇論文即是在專用控制通道技術的架構下，提出以時間為基準的多重通道媒介控制存取層協定，簡稱 TBM。此協定包含三大部分，第一部份透過正確預測控制程序的發起時間，以降低多餘的控制成本；第二部份依各節點狀態動態決定每次傳輸的資料封包數量；第三部份則在所有可用通道中，選出對週遭其他節點影響最小的通道來傳輸資料，以提升通道的同時使用率。前兩者能有效的解決控制通道瓶頸問題，第三部份則能提高資料通道的利用率。模擬結果顯示出本論文所提出的 TBM 較其他方法能提高相當可觀的效能。

TBM: Time-Based Multi-Channel MAC Protocol for Wireless Mesh Networks under Dedicated Control Channel Approach

Student: Yu-Hsiang Cheng Advisor: Dr. Rong-Hong Jan

INSTITUTE OF NETWORK ENGININERRING

NATIONAL CHIAO TUNG UNIVERSITY

Abstract

In wireless mesh networks (WMNs), the *dedicated control channel* (DCC) approach can support broadcasting and avoid multi-channel hidden terminal problem and deafness problem in the multi-channel environment. However, there are two serious problems in the DCC approach: the *control channel bottleneck* and *dynamic channel selection problem*. In this thesis, a protocol, named the *time-based Multi-channel MAC* (*TBM*), is proposed to resolve these two challenges. The TBM protocol consists of three components. The first one aims at reducing the control overhead by properly predicting the control initiation time. The second one can dynamically aggregate multiple packets to transmit with a single control process. The final component selects the channel that has the least influence to nearby nodes to improve channel reusability. Simulation results show that TBM can achieve significant improvement in the throughput in comparison with the existing work.

誌謝

在這兩年的碩士生涯中，最需要感謝的人是我的指導教授簡榮宏博士。老師平日用心指導，在我提出問題時總能指引出最正確的方向，令我在這兩年的研究生活中，不管是在資訊領域的知識亦或研究能力的培養均成長許多。論文的完成更得感謝老師的大力協助，使我的論文能夠完整而嚴謹。

我非常感謝我的口試委員——蔣村杰博士、王國禎老師與陳健老師，他們在我口試的時候給予我的指導讓我有相當大的收穫。再來，必須好好感謝安凱學長。除了教會我許多研究上所需的能力外，對研究的態度嚴謹而專一的學長是我在研究這條路上學習的好對象。接著要提的是兩年日子裡陪伴我努力的實驗室夥伴。與敬之、佑笙、小黑與 NCK 在這兩年一起奮鬥走過的點點滴滴是我無法抹去的回憶。而總是陪我哈啦打屁的世昌學長，總是呵呵笑的蕙如學姊，比較晚認識的家瑋與文彬學長也都在研究上與平日生活上給予我很大的幫助。較晚來的志賢、淑盈與子興也常帶給我許多的歡笑。其他還有很多在這兩年對我有許多幫助的人，怕自己有所遺漏造成殊憾，故僅謝謝所有給我幫助的朋友們。

最後要感謝的是我的家人，由於他們在背後默默的支持以及鼓勵，在這兩年我才能無後顧之憂並專心一致地在研究領域上有所進展。這些都是讓我擁有前進的動力，沒有你們的體諒、包容，相信我無法一個人走到目前的地方。

Content

List of Figures

List of Tables

Chapter 1 Introduction

In recent years, the wireless mesh networks (WMNs) are the most common networks used to extend the reach of the last-mile access to the Internet [1]. WMNs consist of two types of mesh nodes: mesh routers and mesh clients equipped with IEEE 802.11 radio interfaces. A mesh client can access to the network by connecting to one or more mesh routers which supply the functionality of access points (APs), and the traffic in the network is relayed hop-by-hop to the destinations by some mesh routers through wireless links. In this way, a wireless network backhaul is easily established without any wired connection. However, two adjacent wireless links using the same channel cannot transmit concurrently. Therefore, it is expected to exploit multiple channels among mesh nodes to increase capacity of the networks. For example, the IEEE 802.11 a/b/g provides 12, 3, and 3 orthogonal channels, respectively. These channels are spread in non-overlapped spectrums and can be simultaneously used for transmission.

In order to utilize multiple channels in WMNs, designing a multi-channel medium access control (MMAC) protocol is the most important [2]. However, a mesh node using only one interface cannot sense carriers and receive control packets (such as RTS/CTS) from different channel at the same time. This limitation results in three major problems in the design of MMAC [2]. These problems are the multi-channel hidden terminal problem, deafness problem, and broadcasting support.

1) *Multi-channel Hidden Terminal Problem*: The IEEE 802.11 DCF can avoid the hidden terminal problem, but it is further complicated in the multiple channel environment. Consider an example in Fig. 1.1a. Node A has data to transfer to B by sending a RTS packet on channel 2, and then B replies a CTS packet to silent other transmission on

channel 2. During the negotiation between A and B, C and D are in communication using channel 3 so that C cannot detect the CTS from B. Consequently, at the end of the communication between C and D, C may illegally initiate a RTS to B using channel 2 and incur a collision. This problem is called multi-channel hidden terminal problem.

2) *Deafness Problem*: In Fig. 1.1b, node C wants to communicate with B by sending RTS using channel 2, but B cannot hear the RTS of C because the interface of B has tuned to channel 1. Therefore, B will not reply CTS to C, and C will continually retry RTS until exceeding the number of the maximum retry.

3) *Broadcasting Support*: In the multi-channel environment, the interface of each node may not stay on the same channel. Hence, if the higher layer protocol requires broadcasting support at the link layer, it is hard to reach all nodes by broadcasting on any channel.

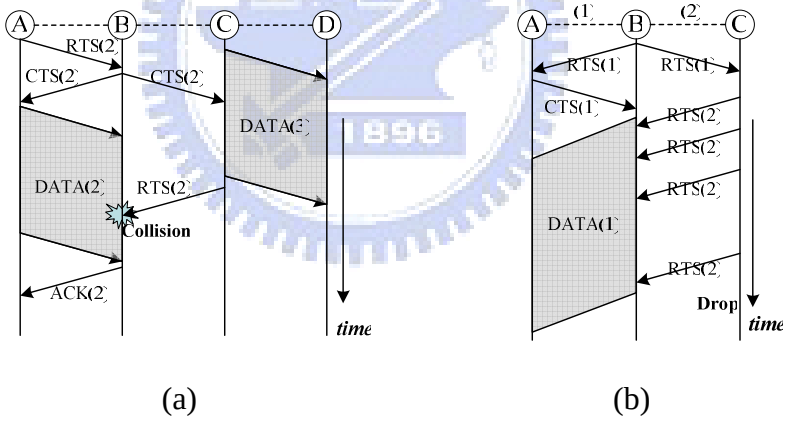


Fig. 1.1 Multi-channel hidden terminal and deafness problems.

To overcome these problems, the *dedicated control channel* (DCC) *approach* was proposed in the literatures [3-9]. In DCC approach, each node has two interfaces: one is the *control interface* and the other is the *data interface*. The control interface is fixed on a common channel, named the *control channel*, for sensing and exchanging control packets. The data interface can dynamically switch among the remaining channels, named *data*

channels, for data transmission. The sender negotiates with the receiver by exchanging some control packets on the control channel to find a data channel which will be free at both sides, and then the sender transmits a data packet to the receiver using the selected data channel[1]. Because all control packets are exchanged in a common channel, the above-mentioned problems are inherently solved.

However, the design of a DCC MMAC protocol confronts two major challenges:

1) *Control channel bottleneck problem*: In DCC approach, only one common channel is used for exchanging control packets. As the example in Fig. 1.2 shows, if the time to transmit a data packet is about 3 times of the length of one control process, the control channel has been fully utilized under 3 data channels. Therefore, the throughput of the networks cannot be further upgraded even if more data channels are added, and the control channel has become a bottleneck of the overall performance. Analytic evidence in [3] has shown that the bottleneck problem will become more serious if the number of data channels, data rate, or node's density increases.
2) *Dynamic channel selection problem*: In DCC approach, the channel's usage of each data channel is flexible, which can be varied in an on-demand matter. Nevertheless, it would be difficult to select a proper channel in a dynamic way, because each node has no enough information about the channel statuses of its neighbor nodes. For example, in Fig. 1.3 node A, B, C and D have free channel lists {1, 2}, {2, 3}, {1, 2, 3}, and {1, 2, 3}, respectively. But if C cannot be aware of the channel statuses of both A and B, it may choose channel 2 to communicate with D. As a result, the link between A and B cannot be active at the same time.

[1] The negotiation is called the *control process* and includes the backoff time, inter-frame spaces, propagation delay, and control packet exchanging dialogue.

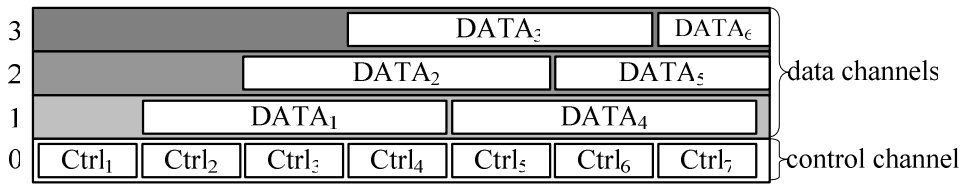


Fig. 1.2 Dedicated control channel MMAC.

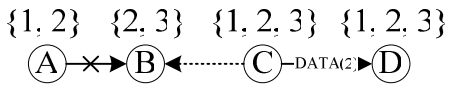


Fig. 1.3 An example of improper selection of a data channel.

To resolve these two challenges, we propose a new MMAC protocol using the DCC approach. This protocol consists of the following three components:

1. *Control initiation time prediction* (*CIP*): In DCC, the sender has to initiate a control process with the receiver for coordinating a communicable data channel. Somewhile, a control process may fail due to no mutual free channel at both sides. If the process fails, the bandwidth of exchanging control packets is wasted and the control channel bottleneck problem is aggravated. To reduce such overhead, this component aims at avoiding the unsuccessful control processes by properly predicting the control process initiation time.
2. *Dynamic data aggregation* (*DDA*): Although CIP component can reduce the number of the unsuccessful control processes, the bottleneck problem is still serious if the control process has to be initialized for each data transmission. To completely overcome this limitation, one effective way is to transmit multiple data packets with a single control process. However, determining the number of data packets to be aggregated becomes a complicated issue. If the number is too small, the effect will not be significant; on the contrary, if it is too large, the retransmission cost would be high. Our DDA component can dynamically make this decision for each transmission according to the utilization of the interfaces (control/data interface idle time).

3. *Enhanced channel selection* (*ECS*): As illustrated in Fig. 1.3, channel 2 is the only data channel free for both A and B. But, it is now under used by C and D. Therefore, A cannot transmit to B until the channel is released by the other two nodes. In other words, the *release time* of channel 2 at node C and D is critical to the transmission between A and B. In our channel selection strategy, any sender-receiver pair will select the channel that has the least influence to the other neighbor nodes. It means that ECS will choose the channel which has the least total increment to the release time of the critical channels of all neighbor nodes. Therefore, the data channels can gain better reusability.

Overall, the main idea of these components is to utilize the information about the time related to the link-layer operations (i.e. control process initiation time, interface idle time, and channel release time). Hence, we called this protocol the *Time-Based MMAC* (TBM).

The remainder of the thesis is organized as follows: In chapter 2, we first compare different types of MMAC approaches to explain why the DCC approach is preferred in our study. Then, the protocols related to this approach are reviewed. In section 3-1 of chapter 3, the basic operation of our protocol is presented. In 3-2, we describe the detail designs of the three components. In chapter 4, we conduct simulation results for our performance evaluation. Finally, conclusion remarks are given in chapter 5.

Chapter 2 Related Works

In this chapter, we first compare different types of MMAC approaches to explain why the DCC approach is preferred in our study. Then, the existing protocols related to the DCC approach are reviewed.

2.1 Comparison of MMAC Approaches

Several MMAC protocols have been proposed in the literature. In general, these protocols can be classified into 4 types: the channel-fixed, receiver-based, split control phase, and dedicated control channel approaches.

1) *The Channel-fixed Approach* [11, 12]: Each interface is fixed on one channel. Any two adjacent nodes can communicate with each other only if both of them have an interface assigned the same channel. However, there are two main problems in this approach. First, hardware cost is too high. For example, if a node wants to use 3 different channels, it should be equipped 3 interfaces. Second, the inflexible channels' usage may confine the network connectivity. For instance, if two consecutive nodes are on the unique path of some source-destination pairs but having no interface on the same channel, the network would be partitioned.
2) *The Receiver-based Approach* [13]: This approach has no connectivity problem. In this approach, each interface of each node has been assigned a dedicated channel, and the node will stay on this channel to listen for request if it has no data to send. Otherwise, the node can connect with any neighbor by turn its interface to the channel that is pre-specified on the receiver side. However, since a node's interface is not always fixed, the multi-channel hidden terminal and deafness problem may occur. In addition, since a node's neighbors are

not necessarily on the same channel, it also has the broadcasting support problem.

3) *Spilt Control Phase Approach* [14, 15]: This approach suggests splitting each beacon interval into two phases, named control phase and data phase, respectively. In the control phase, all sender-receiver pairs exchange control packets using a common channel and select a data channel for data transmission later. Then, at the data phase, each pair can switch their interfaces to the selected data channel and communicate with each other. Because all nodes shares the same channel in the control phase, the multi-channel hidden problem, deafness problem, and broadcasting support problem, are automatically solved. However, this approach has two drawbacks. First, the channels except the common channel used for exchanging control packets are idle in the control phase. Second, this approach relies on strict synchronization mechanism to align the two phases among nodes, and it is difficult to implement in practice.
4) *The Dedicated Control Channel* (DCC) *Approach*: The major advantage of the DCC approach is that it requires no time synchronization. Each node can perform sensing and sending control packets using its control interface at any time. Moreover, since control packets are also exchanged in a common channel, named the control channel, all advantages in the split control phase approach are preserved except the control channel cannot be used for data transmission in DCC.

Owing to these merits of DCC approach, it is preferred to design a MMAC protocol based on this approach. However, as introduced in chapter 1, the performance might be restrained by the bottleneck of the control channel or the selection of data channels. Hence, the related works on this approach were primarily focused on solving these two problems.

2.2 Existing Protocols on DCC Approach

The first DCC-based protocol for the multi-channel environment, named the *dynamic*

channel assignment (DCA), was presented in [3]. In DCA, each node maintains a list of unused data channels, called the *free channel list* (*FCL*). Before sending a data packet, the sender transmits a RTS carrying its FCL to the intended receiver. Based on the FCL carried by RTS, the receiver chooses a data channel that is free at both sides, and replies the selected channel to the sender by sending a CTS. After receiving the response, the sender broadcasts a RES packet to inhibit its neighbor nodes from using the same data channel. Then, the two nodes exchange data using the selected data channel.

Compared with the IEEE 802.11 DCF, the DCA protocol needs additional control bandwidth for the RES packets. To reduce such overhead, Wu and Lee [4] suggested that the sender shall propose a channel in the RTS based on previously behaviors of the receiver. Using this way, if the proposed channel is accepted by receiver, the RES can be omitted. A similar idea is in [5], where the RTS is followed by a reply-to-RTS (RRTS) which indicate whether or not the channel is acceptable. If not, the process will continue until the both sides agree on an acceptable channel. In general, the control overhead can be reduced in [4] and [5] if a proper channel is suggested by the sender. Our protocol can also avoid redundant control packets by predicting proper control initiation time (by the CIP component).

Another line of researches [6, 7] suggested using multiple control channels instead of a single one. The authors in [6] suggested that the sender initializes the control process using the default control channel at the first time. If the default control channel is occupied by another link, the alternative control channel would be chosen to perform the control process. Then, the authors showed that the optimal number of control channels is a function of the total number of available channels and the packet size. For example, with 12 channels, and data packet size of 1028 bytes, 3 control channels are required. The protocol in [7] also employs an extra control channel for replying ACK. But its purpose is to avoid collision between ACK for a receiver and the data packets from other senders, which allows hidden and exposed nodes to conduct their reception and transmission at the same time. Although using

more control channels is beneficial, the available data channels will be sacrificed. In contrast, the dynamic data aggregation technique in our protocol can resolve the bottleneck problem using only one control channel (by the DDA component).

About the channel selection strategy, in the DCA protocol, the sender and receiver have to negotiate a data channel that will be free at both sides. If there are multiple choices, the channel will be selected at random by the receiver. Thus, the requirement is only to find a communicable channel. There are two enhancements in [8] and [9]. The strategy in [8] requires that the received power in the selected channel is the least in order to avoid potential interference. Similarly, in [9], the most robust free data channel will be chosen, according to the carrier-to-interference-ratio. In other words, the both strategies concern not only the connectivity, but also the *quality* of the selected channel. However, their strategies do not concern the influence to nearby nodes, which may lead to a lower channel's reusability. Our strategy (the ECS component) will take this factor into consideration.

Chapter 3 The Proposed TBM Protocol

3.1 Protocol Descriptions

In this section, we describe the overview of the TBM protocol. First of all, we define the network model and symbols that will be used in our protocol in subsection 3.1.1. Then, the basic operation and the major functionalities of the three components are described in subsection 3.1.2. The explanation and detail design of each component will be presented in next section.

3.1.1 Network Model and Symbols

The concerned network consists of (H + 1) non-overlapped channels. Each node has a control interface and a data interface. The first channel (h = 0) is the control channel fixed on the control interface for exchanging control packets. The other channels (h = 1, 2, …, H) are the data channels used by the data interface for data transmission and reception. For each node u, it maintains its own statuses as following:

- *ch_rel_time*(u, h): the *release time* of the h^{th} channel, where h = 0, 1, …, H;
- *if_rel_time*(u): the *release time* of the data interface.

Any change to *ch_rel_time*(u, h), h = 1, …, H, and *if_rel_time*(u) will be sent to nearby nodes carried by some control packets, e.g. CTS or RES. Otherwise, each node u maintains two data structures to record the latest statuses of its neighbor nodes. One is a *channel release time table* (CRT_u) and the other is an *interface release time vector* (IRV_u). When the node u receives the control packets from some other node v, it will update $CRT_u(v, h)$ and $IRV_u(v)$ to keep records of the *ch_rel_time*(v, h) and *if_rel_time*(v) of node v, respectively. In addition, let N_u

denote the set of nodes adjacent to u. For each 1-hop destination $v \in N_u$, a separated queue Q_v is created to buffer any packet that will be sent or forwarded to v. Table 3.1 lists the other symbols used in our TBM protocol.

Table 3.1 Meanings of symbols used in the TBM protocol

Symbol	Meaning
τ	The propagation delay.
$T_{curr}(u)$	Current time of a node u.
T_{SIFS}	Length of the short inter-frame spaces.
T_{DIFS}	Length of the distributed inter-frame spaces.
T_{BF}	Length of the random backoff period.
T_{RTS}	Time to transmit a RTS packet.
T_{CTS}	Time to transmit a CTS packet.
T_{RES}	Time to transmit a RES packet.
T_{ACK}	Time to transmit a ACK packet.
T_{DATA}	Time to transmit a DATA packet.
$\mathcal{H}$	$\mathcal{H} = \{1, 2, \ldots, H\}$; The set of data channels.

3.1.2 Basic Operation

The basic operation of the TBM protocol is illustrated in Fig. 3.1, and it is primarily extended from the DCA protocol proposed in [3]. In this subsection, we will present the overview of the TBM protocol and emphasize on the association with the newly introduced statuses and the three components. The steps of the basic operation are as following:

1. Once the node u having data packets to transmit to a node x, these packets are buffered in Q_x. The node u chooses one of these nodes (here we take node v) which has the oldest packet (has the smallest sequence number) in queues as the targeted receiver.
2. Then, u calculates a *control initiation time* of v in this transmission, denoted as *ctrl_ini_time*(u, v) (presented in subsection 3.2.1). The node u can not start to initial the RTS transmission until this time is equal to $T_{curr}(u)$. However, this initiation time is not

fixed. It might be continuously postponed as long as node *u* observes that it has no chance to initialize a successful control process, e.g. receiving other node's control packets with network allocation vector of the control channel or data channels.

3. At the time $T_{curr}(u)$ that is equal to *ctrl_ini_time*(*u*, *v*), node *u* can start to do the following processes:

   a) First, node *u* decides the number of data packets to be aggregated for node *v*, denoted as $K_{u,v}$ (presented in subsection 3.2.2). Accordingly, the network allocation vector of the possible data transmission (NAV_{DATA}) can be set as

   $$NAV_{DATA} = K_{u,v} \times (T_{DATA} + T_{SIFS}) + T_{ACK} + 2\tau.$$

   b) Next, if there is no carrier on the control channel in a T_{DIFS} plus the remaining T_{BF} period, node *u* will send a RTS to node *v* with NAV_{DATA} and other essential information for the succeeding channel selection at the receiver side (presented in subsection 3.2.3).

   c) Otherwise, if there is carrier on the control channel, it has to go back to step 2 to recalculate a new *control initiation time*.

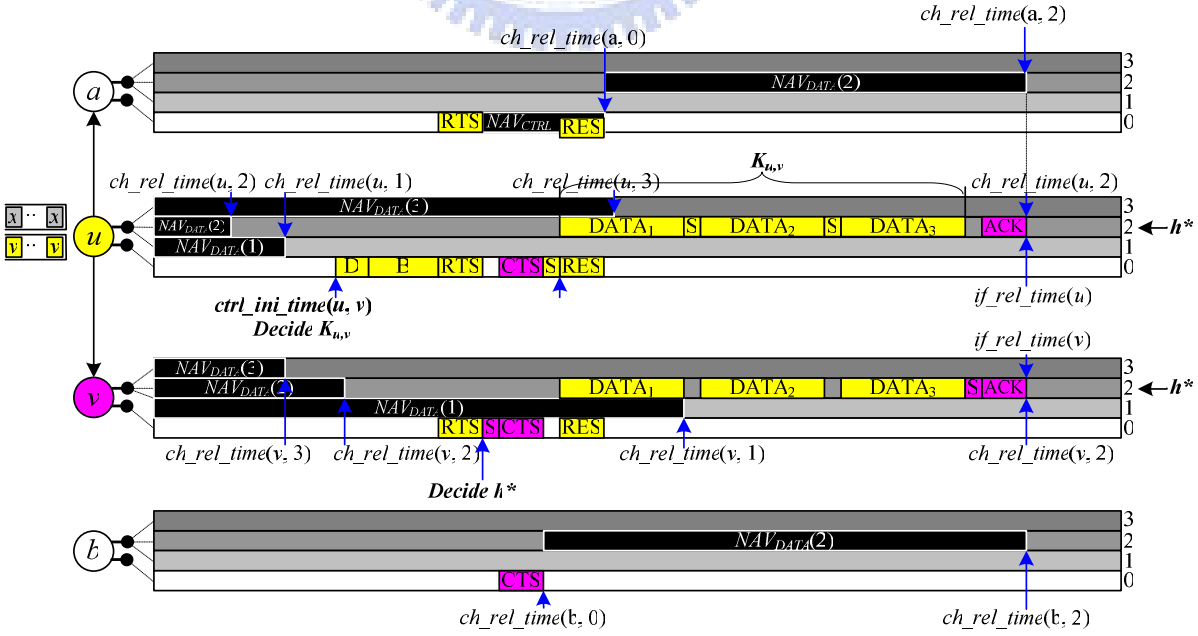


Fig. 3.1 Basic operation of the TBM protocol

4. On the node v receiving the RTS from u at a time $T_{curr}(v)$, it has to find a data channel h^* based on the information in RTS and its own statuses using an enhanced channel selection strategy (presented in subsection 3.2.3). If node v selects a data channel h^* for the forthcoming data transmission, node v would do the following two steps:

   a) Node v update its $if\_rel\_time(v)$ and $ch\_rel\_time(v, h^*)$ such that:

   $$if\_rel\_time(v) = ch\_rel\_time(v, h^*) = T_{curr}(v) + T_{rctrl} + T_{SIFS} + NAV_{DATA} - \tau,$$

   where $T_{rctrl} = T_{SIFS} + T_{CTS} + \tau$. Note that the period of $T_{rctrl} + T_{SIFS}$ is the remaining controlling time before u transmits data packets.

   b) Then, after a T_{SIFS}, node v replies a CTS to u containing the data channel h^* and the *remaining time* of its $if\_rel\_time(v)$ and $ch\_rel\_time(v, h)$ whenever received by other nodes, denoted as $if\_rel\_time^+(v)$ and $ch\_rel\_time^+(v, h)$, for any $h \in \mathcal{H}$, respectively. Their values are as following:

   $$if\_rel\_time^+(v) = \max\{0, if\_rel\_time(v) - T_{curr}(v) - T_{rcrtl}\};$$

   $$ch\_rel\_time^+(v, h) = \max\{0, ch\_rel\_time(v, h) - T_{curr}(v) - T_{rcrtl}\},$$

   Otherwise, if node v has not selected any data channel, after a T_{SIFS}, it directly replies a CTS with $if\_rel\_time^+(v)$ and $ch\_rel\_time^+(v, h)$ which values are the same as the above-mentioned. (The additional fields carried by CTS are h^*(if any), $if\_rel\_time^+(v)$ and $ch\_rel\_time^+(v, h)$.)

5. When node u received the CTS at a time $T_{curr}(u)$, it performs the following steps:

   a) If there is a channel h^* in CTS, it updates its data interface and channel's statuses as following:

   $$if\_rel\_time(u) = ch\_rel\_time(u, h^*) = T_{curr}(u) + ch\_rel\_time^+(v, h^*) + \tau.$$

   b) Then, after a T_{SIFS}, node u broadcasts the h^* and its own $ch\_rel\_time^+(u, h)$, for any $h \in \mathcal{H}$, to nearby nodes using a RES packet, where:

   $$ch\_rel\_time^+(u, h) = \max\{0, ch\_rel\_time(u, h) - T_{curr}(u) - T_{SIFS} - T_{RES} - \tau\}.$$

   Note that the NAV_{DATA} has been implicated in both $ch\_rel\_time^+(v, h^*)$ and

$ch\_rel\_time^{+}(u, h^{*})$ in CTS and RES, respectively. (The additional fields carried by RES are h^{*} and $ch\_rel\_time^{+}(u, h)$.)

c) At the same time with broadcasting the RES, node *u* start to transmit data packets to *v* using the selected channel h^{*}.

On the contrary, if there is no h^{*} in CTS, it has to go back to step 2 to recalculate a new *control initiation time*, and restart its control process again.

6. On node *v* completely receiving data packets from *u*, *v* replies an ACK packet on h^{*}.

In order to prevent the other nodes from using the control channel during the RTS-CTS-RES handshaking, as an irrelevant node *x* (here it is *a* in Fig. 3.1) received the RTS from *u* at a time $T_{curr}(x)$, it updates its control channel's status such that

$$ch\_rel\_time(x, 0) = T_{curr}(x) + NAV_{CTRL},$$

where $NAV_{CTRL} = T_{rctrl} + T_{SIFS} + T_{RES} + \tau$, the time when *x* may receive a RES from *u* (as shown in Fig. 3.1). On the other hand, for any node *x* surrounding to this transmission (here it is either *a* or *b*), whenever it received the RES or CTS from a node *y* (here it is either *u* or *v*) at a time $T_{curr}(x)$, if a data channel h^{*} is going to be used, it will update its data channel's status such that

$$ch\_rel\_time(x, h^{*}) = \max\{ch\_rel\_time(x, h^{*}), T_{curr}(x) + ch\_rel\_time^{+}(y, h^{*})\}.$$

Additionally, a node *x* (including *u*, *v*, *a*, and *b*) will refresh its recorded statues about IRV_x and CRT_x for another node *y*, according to the received CTS or RES from *y* such that

$$IRV_x(y) = T_{curr}(x) + ch\_rel\_time^{+}(y, h^{*});$$

$$CRT_x(y, h) = T_{curr}(x) + ch\_rel\_time^{+}(y, h), \text{ for any } h \in \mathcal{H}.$$

3.2 Components Designs

In this section, we present the detail designs for the three components in the TBM protocol.

3.2.1 Control Initiation Time Prediction

First, we describe the design of CIP component. Before each data transmission, the sender has to initiate a control process with its receiver to coordinate a free data channel. If the coordination fails frequently, it would spend considerable time and bandwidth for exchanging control packets, and it would aggravate the control channel bottleneck problem. Even worse, the NAV_{CTRL} carrying by an ineffective and unnecessary RTS may prevent the other nodes from initializing or completing their control processes. In order to reduce such overhead, one important is to properly predict the control initiation time. The idea is based on utilizing the release time of node's resources (the data interface and data channels) at both sides.

Let us consider two nodes u and v. At any time $T_{curr}(u)$, node u cannot initialize the control process until three conditions has been satisfied. These situations are described as following:

1) First, node u can communicate with v only if there is at least one channel $h \in \mathcal{H}$ that has been released at both sides, i.e., $T_{curr}(u) \geq ch\_rel\_time(u, h)$ and $T_{curr}(u) \geq ch\_rel\_time(v, h)$. In other words, the *earliest channel release time of u and v*, denoted as $ear\_ch\_rel\_time(u, v)$, can be calculated as

   $$ear\_ch\_rel\_time(u, v) = \min\{\max\{ch\_rel\_time(u, h), ch\_rel\_time(v, h)\} \mid h \in \mathcal{H}\}.$$

   Clearly, there is no free data channel between u and v before $ear\_ch\_rel\_time(u, v)$.

2) For a transmission between u and v, neither the data interface of u nor v can be under used, i.e. $T_{curr}(u) \geq if\_rel\_time(u)$ and $T_{curr}(u) \geq if\_rel\_time(v)$. Therefore, the time, when the link of u and v can be released for data transmission, can be defined as,

   $$link\_rel\_time(u, v) = \max\{ear\_ch\_rel\_time(u, v), if\_rel\_time(u), if\_rel\_time(v)\},$$

   which be named the *link release time of u and v*.

3) Node u cannot start its control process until there is neither physical nor protocol interference on the control channel, i.e. $T_{curr}(u) \geq ch\_rel\_time(u, 0)$.

Combining these three facts, we define the *control initiation time* as

$$ctrl\_ini\_time(u, v) = \max\{ch\_rel\_time(u, 0), link\_rel\_time(u, v)\}.$$

Clearly, any control process initiated from u before $ctrl\_ini\_time(u, v)$ cannot be successful.

In practice, the release time of nearby nodes can be acquired from the CTS and RES sent by them, and the statuses about $ch\_rel\_time(v, h)$ and $if\_rel\_time(v)$ in the above equations can be replaced by $CRT_u(v, h)$ or $IRV_u(v)$, respectively. The new equations are as following:

$$ear\_ch\_rel\_time(u, v) = \min\{\max\{ch\_rel\_time(u, h), CRT_u(v, h)\} \mid h \in \mathcal{H}\};$$

$$link\_rel\_time(u, v) = \max\{ear\_ch\_rel\_time(u, v), if\_rel\_time(u), IRV_u(v)\};$$

$$ctrl\_ini\_time(u, v) = \max\{ch\_rel\_time(u, 0), link\_rel\_time(u, v)\}.$$

However, $ctrl\_ini\_time(u, v)$ may be postponed. As shown in Fig. 3.2, before initiating the control process, the value of $ctrl\_ini\_time(u, v)$ has to be continuously postponed, whenever $ch\_rel\_time(u, 0)$ or $link\_rel\_time(u, v)$ is changed due to receiving some control packets. In Fig. 3.2, the $ctrl\_ini\_time(u, v)$ is postponed twice, caused respectively by the changes of $ch\_rel\_time(u, 0)$ and $link\_rel\_time(u, v)$, and it is marked as (a) and (b) in Fig 3.2.

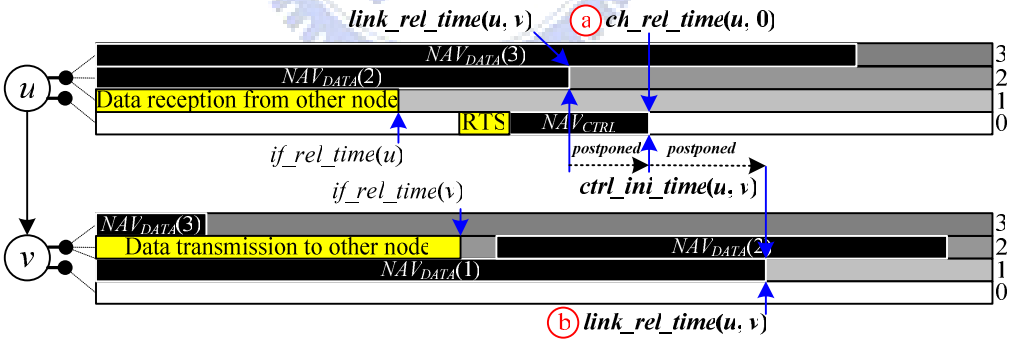


Fig. 3.2 Link release time and control initiation time between two nodes u and v.

3.2.2 Dynamic Data Aggregation

Although CIP can reduce unnecessary RTS transmissions, the bottleneck problem is still subsistent if the control process has to be done before each data transmission. The problem becomes more serious as the number of data channels, data rate, or node's density increases

[3]. It is because that the requests for contention and coordination in the control channel become more intensive under these circumstances. In order to completely break through this limitation, one effective way is to transmit multiple data packets using only one control process [10]. It has three advantages in this way. First, both the physical collisions and the logical blocking by NAV_{CTRL} can be mitigated, because multiple data transmission could be sent after one control process. Second, the time of inter-frame spaces and the backoff will become relatively small due to a long packet's train. Furthermore, the reception of multiple data packets can be replied using only one ACK packet.

However, a new problem arises: *How many packets should be aggregated in one transmission*? If the number is too small, the effect by the above advantages will not be significant. Because a node uses its data interface for a short packet's train, it may still spend the most of time to contend or wait for the access to the control channel. Thus, lead to a lower utilization of its data interface. On the contrary, if the number is too large, it may incur large idle time on its control interface due to too long packet's train. Consequently the performance could be deteriorated by the increased retransmission cost and the unfairness problem from the longer packet's train. On the other hand, for a given node, the loading of the control channel in its surrounding area is not always uniform, which would be varied due to some factors such as flow loading, node's density, and routing path. For example, in Fig. 3-3, it is easier for *u* to coordinate with *v*, since *v* has only two neighbors, but it is much harder if *u* wants to coordinate with *w*, where other four nodes are surrounded and all of them compete to create a link with *w*.

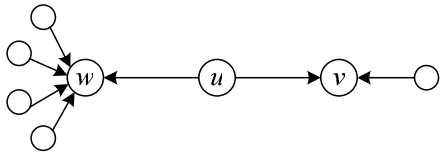


Fig. 3.3 Non-uniform loading in the control channel of node *u*.

Our main idea is to dynamically adjust the number of data packets to be aggregated for each link based on the idle time of both the control and data interfaces. The primary goal is to balance the utilization of the two interfaces such the average throughput can be maximized.

Consider a node u and a destination $v \in N_u$. The number of packets, that will be aggregated from u to v, is denoted as $K_{u,v}$ and initiated as 1 at the beginning. Note that each transmission consists of least one data packet and $K_{u,v}$ can not be over the queue size. Hence, the range of $K_{u,v}$ is bound within 1 and $|Q_v|$.

After node u chooses node v as the targeted destination, u would start to initialize a control process. Recall that the *control initiation time* may be postponed by some reasons (described in 3.2.1). Assuming that node u has been waiting to transmit data packets to v for a period, and it is preparing to restart the control process with v, as shown in Fig. 3.4. At this time point, if node u observes that its data interface has experienced a certain amount of time in *idle* status, denoted as *data_idle_time*(u, v) since the start of the current transmission, it will enlarge its $K_{u,v}$ such that

$$K_{u,v} = \min\{|Q_v|, K_{u,v} + \lceil data\_idle\_time(u, v) / (T_{DATA} + T_{SIFS}) \rceil\}.$$

The reason is that the data interface's idle time during this period is resulted from the recent contention in the control channel of either u or v. It means that the longer *data_idle_time*(u, v) is, the more serious the control bottleneck problem is. In order to avoid overly sending control messages and to arise the utilization of u's data interface, the length of the packet's train has to be expended in proportion to the experienced idle time.

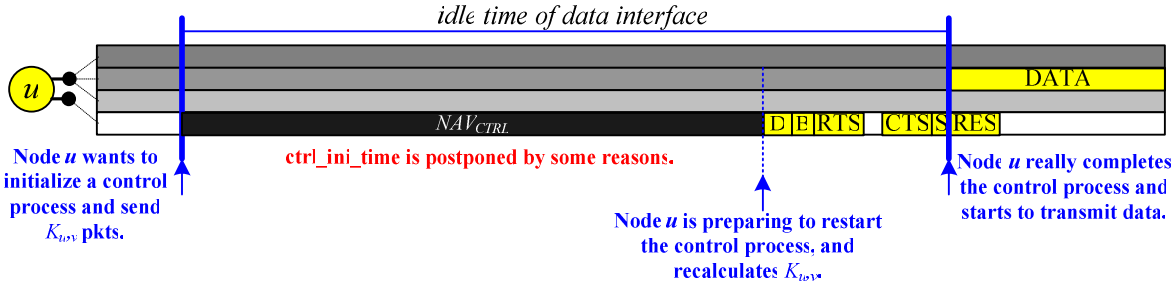


Fig. 3.4 Data interface idle time (simplified).

On the contrary, as shown in Fig. 3.5, if node *u* observes that its control interface was in idle status for a period of time during the current data transmission, denoted as *ctrl_idle_time*(*u*, *v*), in order to increase the control channel's utilization and avoid possible cost from retransmission and unfairness problem, it has to shrink back its $K_{u,v}$ in proportion to the *ctrl_idle_time*(*u*, *v*). That is, node *u* adjusts

$$K_{u,v} = \max\{1, K_{u,v} - \lfloor ctrl\_idle\_time(u, v) / (T_{DATA} + T_{SIFS}) \rfloor\},$$

at the end of the current transmission to *v*.

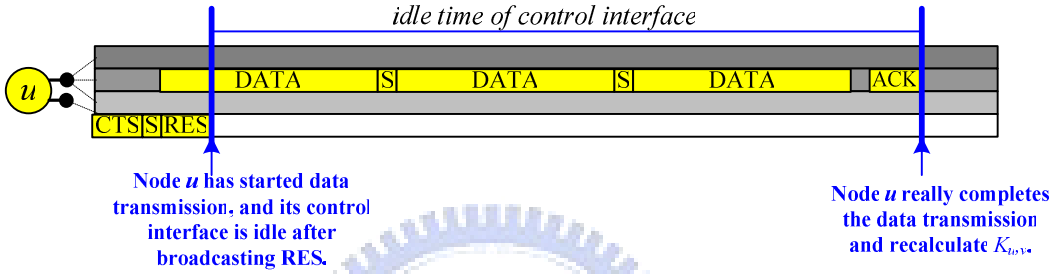


Fig. 3.5 Control interface idle time (simplified).

Now, we formally define the two variables of *data_idle_time*(*u*, *v*) and *ctrl_idle_time*(*u*, *v*). For a transmission from *u* to *v* (refer to the illustration in Fig. 3.6), we denote *data_tx_time**(*u*, *v*) and *data_tx_time*(*u*, *v*) as respectively the *earliest* and the *actual* time when node *u* can start transmitting data packets to *v*. That is,

$$data\_tx\_time^*(u, v) = ctrl\_ini\_time^*(u, v) + T_{sctrl} + T_{SIFS};$$

where $T_{sctrl} = T_{DIFS} + T_{BF} + T_{RTS} + \tau + T_{SIFS} + T_{CTS} + \tau$, and *ctrl_ini_time**(*u*, *v*) is the *first* control initiation time predicted for the current transmission by CIP component without any postponement. By definitions, *u*'s data interface need not be active before *data_tx_time**(*u*, *v*) and it must be under used after *data_tx_time*(*u*, *v*). In other words, at any time $T_{curr}(u)$, the data interface of *u* is in idle status only if $T_{curr}(u) \in [data\_tx\_time^*(u, v), data\_tx\_time(u, v))$. However, the gap between *data_tx_time**(*u*, *v*) and *data_tx_time*(*u*, *v*) is not only caused by the contention in the control channel, but also results from the change of *link_rel_time*(*u*, *v*), i.e. $T_{curr(u)} < link\_rel\_time(u, v)$. It means that the statuses of data channels of u or v is not available. When the later case happens (see the example of the period between t_1 to t_2 in Fig.

3.6a), there is no free data channel to be used for data transmission even if the control channel is free. For the reason, the data interface's idle time should ignore this case. Accordingly, we have the following definition:

At any time $T_{curr}(u)$, the data interface of a node u is *idle* if and only if

i) $data\_tx\_time^*(u, v) \le T_{curr}(u) < data\_tx\_time(u, v)$;

ii) $link\_rel\_time(u, v) < T_{curr}(u)$.

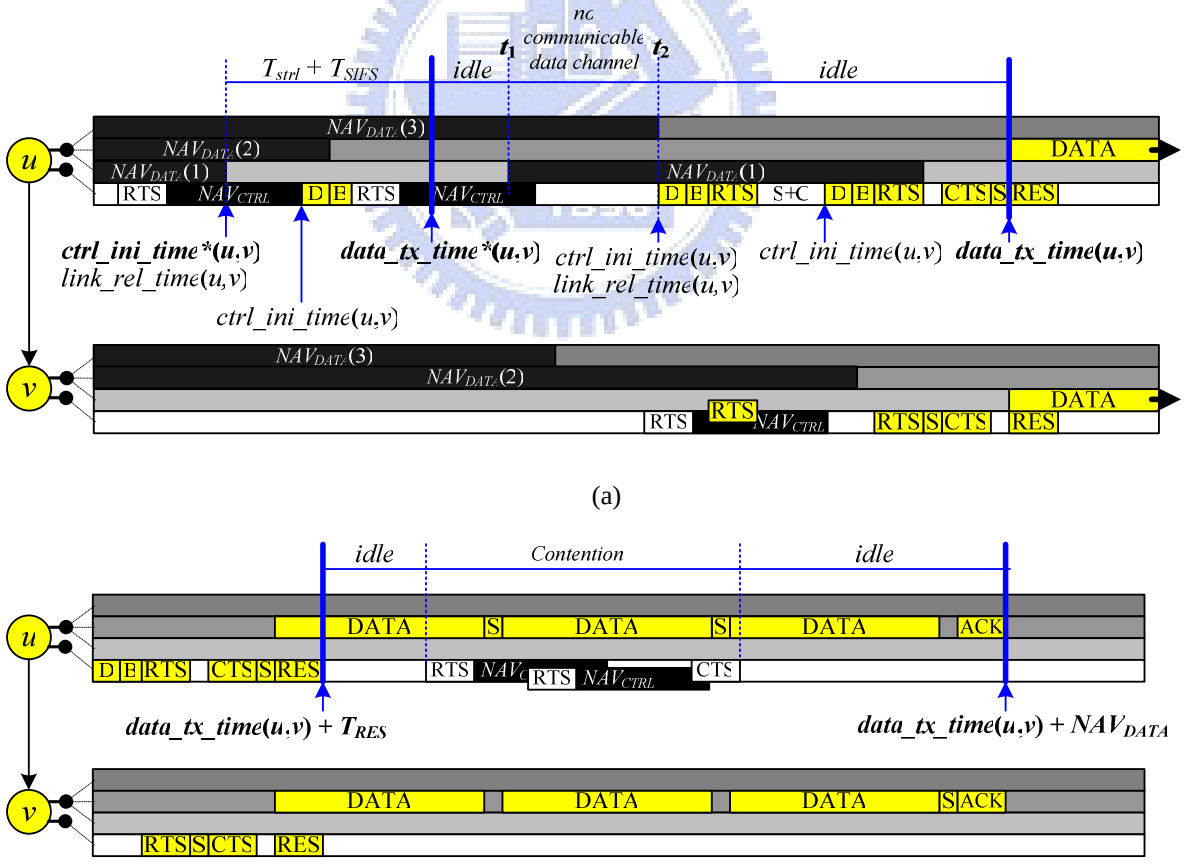
Similarly, referring to Fig. 3.6b, the control interface's idle time is defined as follows:

At any time $T_{curr}(u)$, the control interface of a node u is *idle* if and only if

i) $data\_tx\_time(u, v) + T_{RES} < T_{curr(u)} < data\_tx\_time(u, v) + NAV_{DATA}$;

ii) $ch\_rel\_time(u, 0) < T_{curr(u)}$

iii) There is no physical carrier on the control channel of u.



(a)

(b)

Fig. 3.6 (a) Data interface idle time; (b) Control interface idle time.

3.2.3 Enhanced Channel Selection Strategy

In the primitive DCC-based protocol [3], there is no well-designed channel selection strategy, and it is simply to find an available data channel that is free at both sides. If there are multiple choices, the channel will be randomly selected. Therefore, it results in the *dynamic channel selection* problem mentioned in chapter 1. Our ECS component aims at improving the channels' reusability such that more data transmissions among different nodes can be active simultaneously. The main idea is to select the data channel that will cause the least total increment to the release time of the critical recourses (either data channels or data interface) essential for the possible transmissions at nearby nodes.

Let us formally describe our strategy using symbol terms. Consider a sender u and a receiver v. The time $T_{curr}(u)$ is now at $ctrl\_ini\_time(u, v)$. At this time, node u performs three calculations for each neighbor $w \in N_u - \{v\}$.

1. First, sender u calculates the current status of neighbor w.
2. Node u forms a free channel list as follows, according to the channel statuses of v in CRT_u

$$FCL(u \mid v) = \{h \in FCL(u) \mid CRT_u(v, h) \le T_{curr}(u) + T_{sctrl}\},$$

where

$$FCL(u) = \{h \in \mathcal{H} \mid ch\_rel\_time(u, h) \le T_{curr}(u) + T_{sctrl}\},$$

Then, node u calculates the future status of neighbor w if node u and v select a data channel $h \in FCL(u \mid v)$ for data transmission.

3. Final, u calculates the total increment to the release time of a data channel $h \in FCL(u \mid v)$ by the current and future statuses of each neighbor w, and sends the results to v by RTS.

After receiving RTS, receiver v performs the same three calculations as sender u, and chooses the data channel for data transmission according to these calculating results.

Now, we formally define the detailed calculations. First, u calculates the earliest time when at least one data channel will be released at node $w \in N_u - \{v\}$, denoted as $CR_u(w)$, based

on its CRT_u, i.e.,

$$CR_u(w) = \min\{CRT_u(w, h) | h \in \mathcal{H}\}.$$

Combining with w's data interface release time in IRV_u, we define

$$NR_u(w) = \max\{CR_u(w), IRV_u(w)\}.$$

The two variables of $CR_u(w)$ and $NR_u(w)$ are called the *critical channel release time* and *node release time* of w, respectively. The $NR_u(w)$ implicates the earliest time when node w can start a transmission, since for any destination $s \in N_w$, its control initiation time is confined as

$$NR_u(w) \le link\_rel\_time(w, s) \le ctrl\_ini\_time(w, s).$$

In other words, node w can not start a transmission before $NR_u(w)$, and it can represent the status of w before u and v select a data channel for data transmission.

Then, if node u has selected a data channel $h \in FCL(u \mid v)$, the new critical channel release time of $w \in N_u - \{v\}$, denoted as $CR_u^+(w \mid h)$, may be enlarged by the NAV_{DATA} on channel h. It can be formulated as follows

$$CR_u^+(w \mid h) = \min\left\{\begin{array}{l}\max\left\{\begin{array}{l}CRT_u(w, h), \\ T_{curr}(u) + T_{sctrl} + T_{SIFS} + NAV_{DATA}\end{array}\right\}, \\ \min\{CRT_u(w, h') \mid h' \in \mathcal{H} - \{h\}\}\end{array}\right\}.$$

This equation indicates the original value may be replaced by the release time of another channel $h' \in H - \{h\}$ if the release time of the original critical channel is enlarged so that it is no longer critical. Similarly, the corresponding node release time of w can be rewrote as

$$NR_u^+(w \mid h) = \max\{CR_u^+(w \mid h), IRV_u(w)\},$$

and it can represent the new status of neighbor w after node u and v select a communicable channel $h \in FCL(u \mid v)$ for data transmission. Using these terms, the increment to the node release time of w, resulted from the data transmission between u and v on data channel h can be characterized as

$$\Delta_u(w,h)=\begin{pmatrix}\max\{NR_u^+(w\mid h),T_{curr}(u)+T_{sctrl}+T_{SIFS}+T_{RES}+\tau\}-\\ \max\{NR_u(w),T_{curr}(u)+T_{sctrl}+T_{SIFS}+T_{RES}+\tau\}\end{pmatrix}.$$

Notice that this equation neglects the part of increment before $T_{curr}(u)+T_{sctrl}+T_{SIFS}+T_{RES}+\tau$, since the neighbor *w* cannot use the control channel due to receiving the RTS packet and *w* can initial the control process after it has received the RES packet from *u*. Accordingly, the total increment to the node release time of *u*'s neighbors can be defined by

$$\Delta_u(h)=\sum_{w\in N_u-\{v\}}\Delta_u(w,h).$$

Node *u* will calculate the $\Delta_u(h)$ for each $h \in FCL(u \mid v)$ and sends this information and N_u to *v* accompanying with the RTS packet. (There are four additional types of information carried by RTS: NAV_{DATA}, $FCL(u \mid v)$, $\Delta_u(h)$, and N_u.)

When node *v* received the RTS at a time $T_{curr}(v)$, it performs the same calculation for each free data channel $h \in FCL(v) \cap FCL(u \mid v)$ and neighbor $w \in N_v - \{u\} - N_u$, where

$$FCL(v) = \{h \in \mathcal{H} \mid ch\_rel\_time(v, h) \le T_{curr}(v) + T_{rctrl}\},$$

except

$$CR_v^+(w\mid h)=\min\left\{\begin{matrix}\max\left\{\begin{matrix}CRT_v(w,h),\\ T_{curr}(v)+T_{rctrl}+T_{SIFS}+NAV_{DATA}-\tau\end{matrix}\right\},\\ \min\{CRT_v(w,h')\mid h'\in H-\{h\}\}\end{matrix}\right\},$$

and

$$\Delta_v(h)=\sum_{w\in N_v-\{u\}-N_u}\begin{pmatrix}\max\{NR_v^+(w\mid h),T_{curr}(u)+T_{rctrl}\}-\\ \max\{NR_v(w),T_{curr}(u)+T_{rctrl}\}\end{pmatrix}.$$

Then, for each $h \in FCL(v) \cap FCL(u \mid v)$, the total increment to the node release time of all neighbors of either *u* or *v* are merged into the following cost function

$$\Delta_{u,v}(h)=\Delta_u(h)+\Delta_v(h).$$

The channel *h* that has the least $\Delta_{u,v}(h)$ will be chosen as the communication channel, and denotes as h^*. The selected channel *h** will be sent back to *u* using a CTS and then broadcasted

to *u*'s neighbor using a RES packet.

An example of this channel selection strategy is illustrated in Fig. 3.7, where node *u* transmits to *v* using channel 1. The node release time of *b* is increased by $\Delta_v(b,1) > 0$, since channel 1 is no longer the critical channel of *b*. It also influences node *c*, but the range is started by the time when *c* received the RES. Likewise, the critical channel of *d* is altered from 1 to 3, but the $NR_u(d)$ is not changed, because the interface release time of *d* dominates the value. Lastly, the transmission has no influence to node *a*, since channel 1 is not critical to *a*.

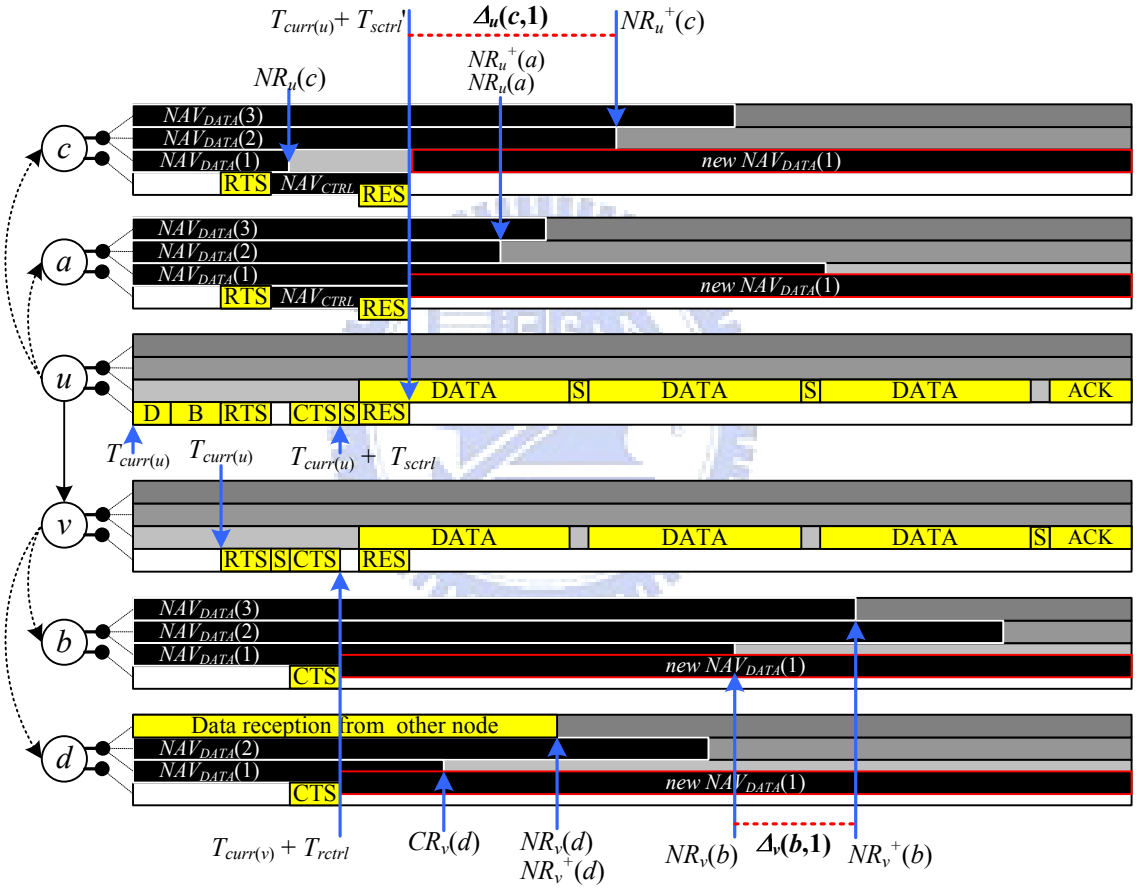


Fig. 3.7 Increments to node release time if selecting channel 1.

Chapter 4 Experiments

In this chapter, we conduct simulation results to evaluate the performance of our TBM protocol, and compare with IEEE 802.11 MAC protocol and DCA protocol proposed in [3]. Our simulations were conducted using the network simulator 2 (ns-2) [16] with CMU wireless extensions. The network topologies under test are classified into two categories:

1) *Random Single-hop Networks*: In each run, there are 100 nodes randomly placed on a region. We generate single hop flows for 50 distinct node pairs. The performance here measures the link layer throughput.
2) *Multi-hop Networks*: We focus on the end-to-end throughput. There are two subcases:
    b) *Grid Multi-hop Networks*: As shown in Fig. 4.1, a 10×10 grid topology with 10 horizontal flows and the other 10 vertical flows was tested. The distance between adjacent nodes is 100m. In each run, the source node and the destination node of each flow are randomly chosen from the margin of the grid.
    c) *Random Multi-hop Networks*: For each random multi-hop network, we also randomly place 100 nodes on a region, while 20 multi-hop flows are established 20 distinct source and destination pairs. Each flow will go through the shortest routing path, fixed at the beginning.

Under these network topologies, we study the impact from three factors:

a) Number of data channels is varied from 2 to 11.
b) Data rate of data channels are varied by 0.5Mbps, 1Mbps, 2Mbps, 5.5Mbps, and 11Mbps.
c) Data frame size is varied by 256 bytes, 512 bytes, 1024 bytes, and 2048 bytes.

The detail parameters setting are listed in Table 4.1.

Table 4.1 Simulation Values

Parameters	Values
Deployment Region	1000m×1000m
Number of Nodes	100 nodes
Transmission Range	250 meters
Carrier Sensing Range	250 meters
Number of Channels (*H*)	12 non-overlapped channels
Data Rate of the Control Channel	11 Mbps
Data Rate of each Data Channel	11 Mbps
Data Frame Size	1024 Bytes
Queue Size	50 packets
Traffic Type	UDP flow with Constant Bit Rate (CBR)
Flow Rate	11 Mbps in the single-hop topology
	2 Mbps in the multi-hop topology
Simulation Duration	100 seconds
Number of runs	20 runs

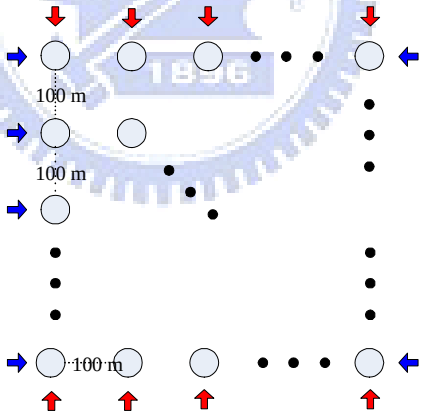


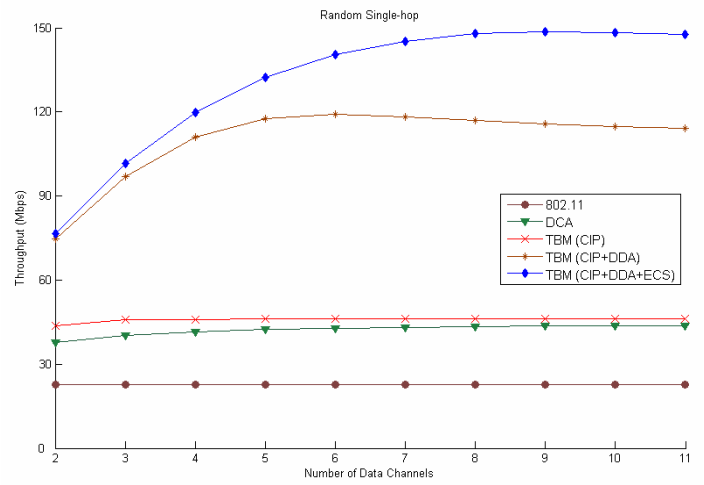
Fig. 4.1 10×10 Grid Topology

4.1 Comparison in varied number of data channels

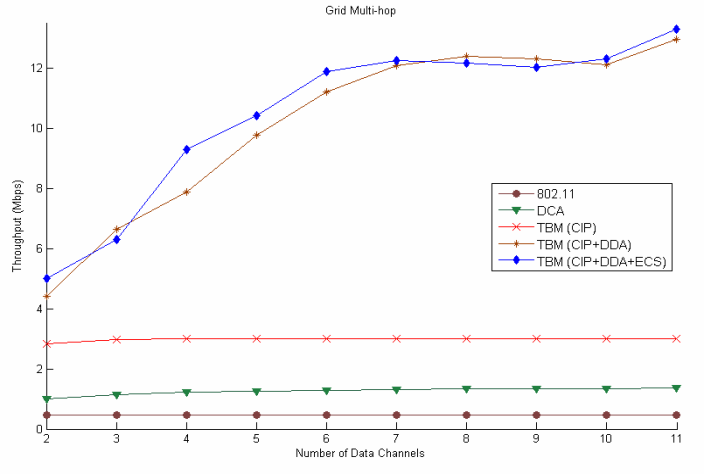
In this section, we study the impact of the number of data channels to observe two things: the effect on the aggregate throughput of the networks and the utilization of data channels of each scheme. Fig. 4.2a and 4.2b indicate that our CIP scheme can elevate the aggregate throughput at least 5% and 7% in random single-hop networks and random multi-hop networks,

respectively. The upgrade comes from that CIP can effectively prevent from unnecessary RTS transmission. Fig. 4.2c shows the improvement from CIP can be up to 120% in the grid topology. The significant enhancement in the grid topology has two reasons. First, the transmission target changes frequently. Compared to DCA, the node using CIP scheme has more detailed information of its neighborhood's channel release time after the transmission target switches to another one. Therefore, the node using CIP could transmit RTS when the destination has the same free channel. Second, the node density in the grid networks is more uniform than the density in the other two networks. Therefore, each node in the grid topology confronts much more contention than the other two topologies. Because of the more contention, the unsuccessful control processes which CIP focuses are so much that the performance DCA achieves is only a little higher than 802.11. Hence, CIP can gain good improvement by reducing the unsuccessful control processes.

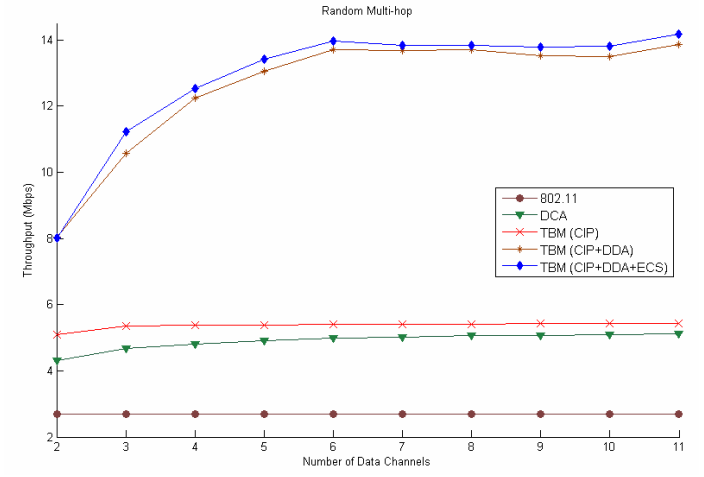
In addition, the curves in Fig. 4.2 depict that the throughput of DCA is saturated as the number of data channels is more than three. In contrast, our protocol with DDA component is not saturated until the number of data channels is more than six. Fig. 4.2 also indicates that the aggregate throughput synergically achieved by DDA and CIP is higher than DCA over 97% in single-hop topology (Fig. 4.2a), 340% in grid topology (Fig. 4.2b), and 86% in random multi-hop topology (Fig. 4.2c). The results have verified that DDA have higher throughput and achieves higher utilization of data channels. Finally, Fig. 4.2a presents that ECS can effectively improve the throughput of the link layer and upgrade the utilization of data channels further. In Fig. 4.2a, ECS improves 18% compared to DDA scheme as there are six data channels, and ECS has 30% enhancement more than DDA as there are 11 data channels. However, Fig. 4.2b and Fig. 4.2c display that compared to DDA, ECS only upgrades the throughput at most 18% and 6% in the grid topology and in the random multi-hop topology, respectively.



(a)



(b)



(c)

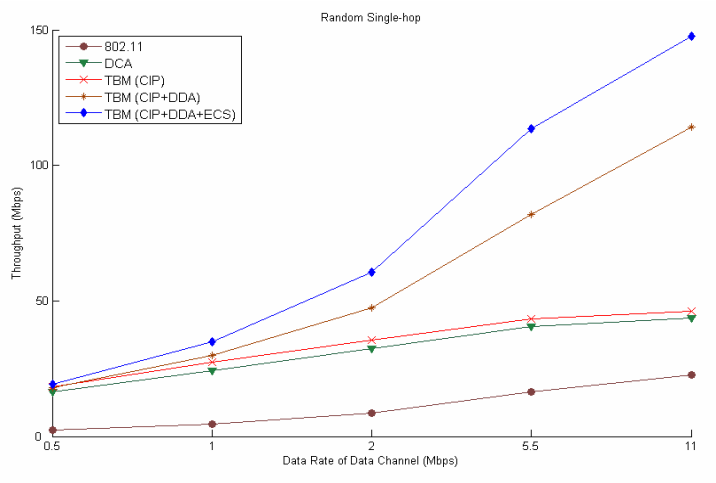
Fig. 4.2 Variation in the number of data channels (a) Random single-hop; (b) Grid multi-hop; (c) Random multi-hop

4.2 Comparison in varied data rate of data channels

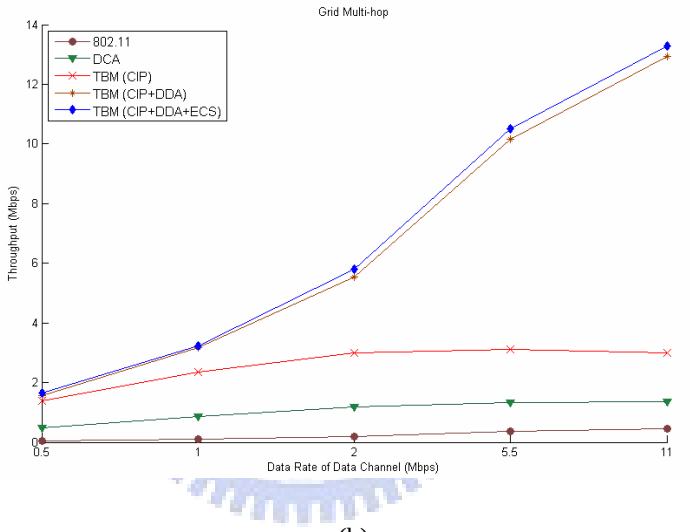
Analytic evidence in [3] has shown that the bottleneck problem will become more serious if the data rate increases. In this subsection, we vary the data rate of data channels to observe its effect. In Fig. 4.3a, as the data rate of data channels is 5.5 Mbps, DCA has 145% improvement more than 802.11, but it is down to 93% when the data rate of data channels is 11 Mbps. The results verify that the faster the data rate is, the more serious the bottleneck problem is. Compared to DCA, CIP scheme gets significant improvement of the throughput in the grid topology even if the data rate is 0.5 Mbps. Besides, in Fig. 4.3a and 4.3c, the throughput achieved by our TBM protocol is not saturate even the data rate is up to 11 Mbps. The simulation results in Fig. 4.3 demonstrate that our TBM protocol can achieve higher throughput as the data rate of data channels increases because the control channel bottleneck problem can be resolved by TBM protocol efficiently.

4.3 Comparison in varied data frame size

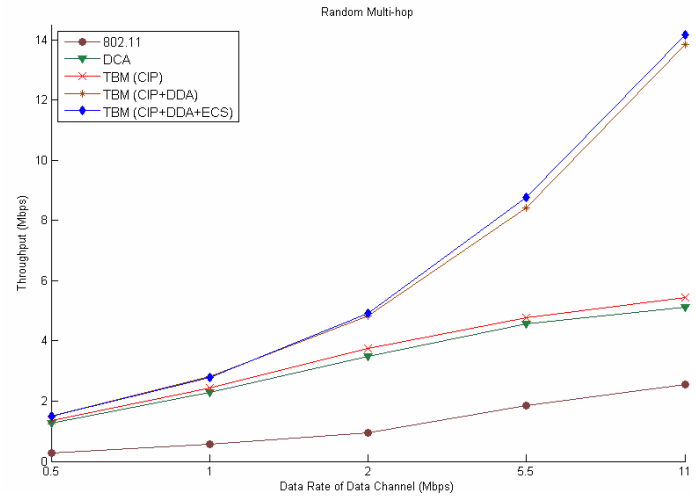
In this section, we vary the frame size of the data packet. In Fig. 4.4a, DCA has 90% and 147% improvement more than 802.11 when frame size is 1024 bytes and 2048 bytes, respectively. It indicates that the level of the control channel bottleneck problem decreases as the data frame size increases. In the contrary, DDA upgrades the throughput 192% and 63% more than DCA as frame size is 1024 bytes and 2048 bytes, respectively. The results in Fig. 4.4 demonstrate that our DDA scheme can achieve higher throughput than DCA even if the data frame size is as small as 256 bytes.



(a)

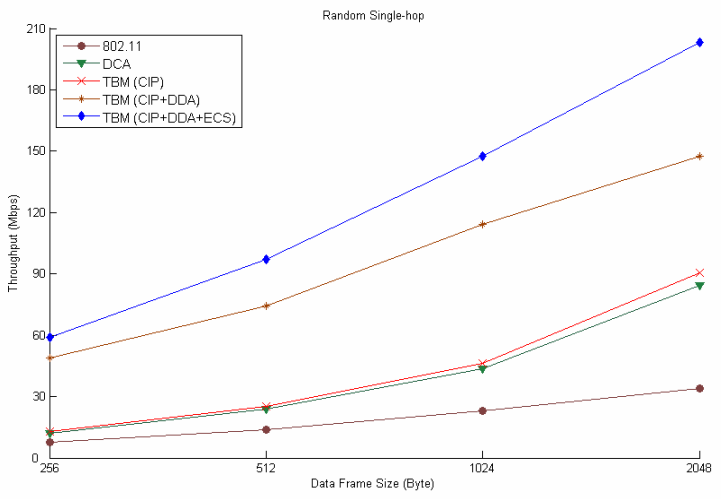


(b)

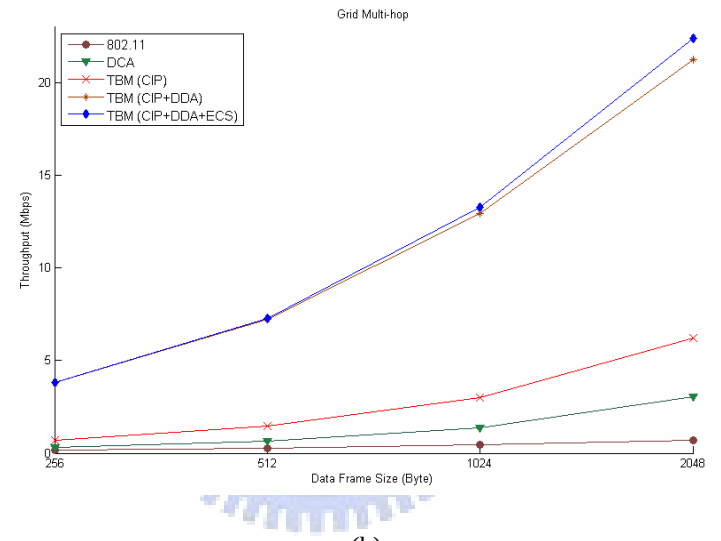


(c)

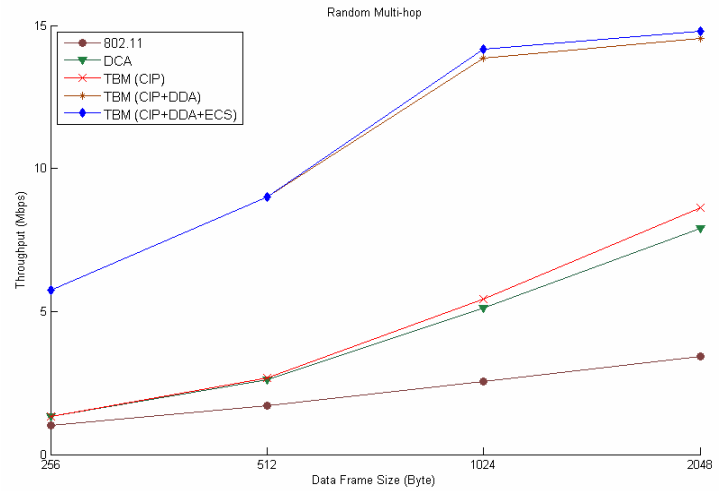
Fig. 4.3 Variation in data rate of data channels (a) Random single-hop; (b) Grid multi-hop; (c) Random multi-hop.



(a)



(b)



(c)

Fig. 4.4 Variation in data frame size (a) Random single-hop; (b) Grid multi-hop; (c) Random multi-hop.

Chapter 5 Conclusion

In this thesis, we have proposed a Time-based MMAC (TBM) protocol to overcome the control channel bottleneck and the dynamic channel selection problem for the DCC approach. There are three components in the TBM protocol. The *control initiation time prediction* (CIP) can reduce the unnecessary control process by properly predicting the control initiation time to increase the chance for a successful coordination. The *dynamic data aggregation* (DDA) can dynamically adjust the number of aggregated packets according to the real-time condition of both the control and data interfaces. The *enhanced channel selection strategy* (ECS) can gain better channel reusability by selecting the channel that has the least influence to nearby nodes. Simulation results have shown that our protocol with these components achieves significant improvement in comparison with previous works, especially when the number of channels, data frame size, and data rate of data channels are large.

Bibliography

[1]. I. F. Akyildiz and X. Wang, Weilin, “Wireless mesh networks: a suvery,” *Computer Network and ISDN System*, vol. 47, pp. 445-487, 2005.

[2]. J. Mo. H.-S. So, and J. Walrand, "Comparison of multichannel MAC protocols," *IEEE Transaction on Mobile Computing*, vol. 7, no. 1, 2008.

[3]. S.-L. Wu, Y. Lin, Y.-C. Tseng, and J.-P. Sheu, "A new multi-channel MAC protocol with on-demand channel assignment for multi-hop mobile ad hoc networks," *In Proc. of Int'l Symp. on Parallel Architectures, Algorithms and Networks (ISPAN'00)*, pp. 232-237, 2000.

[4]. P.-J. Wu and C.-N. Lee, "On-demand commection-oriented multi-channel MAC protocol for ad-hoc network," *In Proc. of 3rd IEEE Conf. Society Conf. on Sensor and Ad Hoc Networks*, pp. 621-625, 2006.

[5]. W.-C. Hung, K.-L.-E. Law, and A. Leon-Garcia, "A dynamic multi-channel mac for ad-hoc LAN," *In Proc. of 21st Biennial Symp. On Communications*, pp. 31-35, 2002.

[6]. H. Koubaa, "Fairness-enhance multiple control channels MAC for ad hoc networks," 2005.

[7]. M.-B. Benveniste and Z. Tao, "Performance evaluation of a medium access control protocol for IEEE 802.11s mesh networks," *In Proc. of IEEE Sarnoff Symp.*, 2006.

[8]. J. Zhang. Y. Wang, and J. Wang, "DCC-MAC: A new MAC protocol for ad-hoc networks based on dual control channel," *In Proc. of 25th IEEE Int'l Symp. on Personal Indoor and Mobile Radio Communication*, pp. 1341-1345, 2003.

[9]. N. Jain, S-R. Das, A. Nasipuri, "A multichannel CSMA MAC protocol with receiver-based channel selection for multihop wireless network," *In Proc. of 10th Int'l Conf. on Computer Communications and Networks (ICCCN'01)*, 2001.

[10]. P. Kyasanur, J. Padhye, and P. Bahl, "On the efficacy of separating control and data into

different frequency bands," *In Proc. of Broadband Networks*, pp. 646-655, 2005.

[11]. A. Raniwala, K. Gopalan, and T. C. Chiueh, “Centralized channel assignment and routing algorithm for multi-channel wireless mesh networks,” *ACM Mobile Computing and Communications Review*, vol. 8, no. 2, pp. 50-55, 2004.

[12]. A.K. Das, H.M.K Alazemi, R. Vijayakumar and S. Roy, “Optimization models for fixed channel assignment in wireless mesh networks with multiple radios,” *IEEE SECON*, 2005.

[13]. J. Crichigno, M.-Y. Wu, W. Shu, "Protocols and architectures for channel assignment in wireless mesh networks," *Ad Hoc Networks*, to appear, 2007.

[14]. J. So, and N. Vaidy, "*Multi-channel MAC for ad hoc networks: handling multi-channel hidden terminals using a single transceiver*," *In Prof. of MobiHoc’04*, 2004.

[15]. J. Chen, S. Sheu, C. Yang, "A new multichannel access protocol for IEEE 802.11 ad hoc wireless LANs," In proc. of IEEE Int'l Symp. on Personal, Indoor and Mobile Radio Communications, 2003.

[16]. ns-2. [Online]. Available: http://www.isi.edu/nsnam/ns/