Computer Networks 55 (2011) 2176–2195

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Release-time-based multi-channel MAC protocol for wireless mesh networks ☆

Andy An-Kai Jeng, Rong-Hong Jan *, Chi-Yu Li, Chien Chen

Department of Computer Science, National Chiao Tung University, 1001 Ta Hsueh Road, Hsinchu 30005, Taiwan

ARTICLE INFO

Article history:
Received 21 August 2009
Received in revised form 12 July 2010
Accepted 28 February 2011
Available online 6 March 2011
Responsible Editor: L. Lenzini

Keywords:
Wireless mesh network
Multi-channel multi-interface
Medium access control
Channel selection

ABSTRACT

The wireless mesh network (WMN) has been considered one of the most promising techniques for extending broadband access to the last mile. In order to utilize multiple channels to increase the throughput in WMNs, a variety of multi-channel MAC (MMAC) protocols have been proposed in the literature. In particular, the *dedicated control channel* (*DCC*) *approach* can greatly simply many design issues in multi-channel environments by using a common control channel to exchange control signals. On the other hand, it allows each sender–receiver pair to dynamically select a data channel for their data transmission in an on-demand matter. However, the common control channel would become a bottleneck of the overall performance. Besides, the selection of data channels would be highly related to the final throughput. In this paper, we propose a new MMAC protocol, named the *release-time-based MMAC* (*RTBM*) to overcome the control channel bottleneck and data channel selection problems in the *DCC* approach. The *RTBM* consists of three major components: (1) *Control initiation-time predication* (*CIP*); (2) *Dynamic data-flow control* (*DDC*); (3) *Enhanced channel selection* (*ECS*). The *CIP* can predict a proper initiation time for each control process to reduce control overhead. The *DDC* can dynamically adjust the flow of each data transmission to fully exploit the channel bandwidth. The *ECS* can achieve a higher reusability of data channels to further enhance the throughput. Simulation results show that the *RTBM* can substantially improve the throughput in both single-hop and multi-hop networks.

1. Introduction

The wireless mesh network (WMN) has been considered one of the most promising techniques for extending broadband access to the last mile [1]. The WMN consists of a set of mesh access points (MAPs). A mobile station can access the network by connecting to a nearby MAP. Each MAP acts as a wireless router to forward traffic hop-by-hop to destinations. Thus, by deploying in such a fashion, a backhaul network can easily be built up without wired connection. Moreover, the network capacity can be substantially improved by using multiple channels. The IEEE 802.11b/g and 802.11a standards provide up to 3 and 12 orthogonal channels, respectively, in 2.4 GHz and 5 GHz spectrums. Nodes within the interference range of each other can transmit on different channels simultaneously to increase the throughput.

In order to utilize multiple channels in WMNs, a key issue is to design a *multi-channel medium access control* (*MMAC*) *protocol* [2] to handle operations at the data-link layer. However, due to the limitation that a wireless card (for most commercial

☆ This research was supported in part by the National Science Council, Taiwan, ROC, under grants NSC97-2219-009-006, and 97-2221-E-009-049-MY3.
* Corresponding author. Tel.: +886 3 5712121x31539.
E-mail address: rhjan@cis.nctu.du.tw (R.-H. Jan).

1389-1286/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.comnet.2011.02.017

devices) cannot perform on two different channels at a time, the design of many essential mechanisms, such as contention resolution, hidden terminal avoidance, and broadcasting support, could be much more difficult in comparison with a single-channel MAC (e.g. IEEE 802.11 DCF).

To overcome the above limitation, a prominent approach, called the *dedicated control channel* (*DCC*), was proposed [3–10]. In this approach, each node is equipped with a *control interface* and a *data interface*. The control interface is permanently fixed on a common channel (called *control channel*) for sensing or exchanging control signals. On the other hand, the data interface can switch among the remaining channels (called *data channels*) for data transmission. As shown in Fig. 1, if node u intends to transmit to node v, it first initiates a control process with v on the control channel to coordinate a data channel. Then, nodes u and v can communicate on that channel. Since the control channel is shared by all nodes, other competitors in the interference range of u or v can be aware of the coordination. Besides, a link-layer broadcast can be easily achieved by emitting on the control channel. Most importantly, since each node has an interface dedicated to listen to the control channel, nodes can exchange control signals without any time synchronization mechanism.

However, designing a *DCC*-based MMAC protocol confronts two major challenges:

(1) *Control channel bottleneck problem*: As described above, using a common control channel can greatly simplify the design of a MMAC. Nonetheless, if too many nodes contend on it, the control channel would become a bottleneck of the overall performance. As shown in Fig. 2(a), three sender–receiver pairs $\{u, v\}$, $\{x, y\}$, and $\{w, z\}$ are coordinating on the control channel ch_0. Besides, there are three data channels with bandwidth B. Ideally, the throughput can achieve $3B$ if each pair transmits on a different data channel. However, since the time required for a control process is about a half of a data transmission in this example, at most two data channels can be utilized at the same time, resulting in a lower throughput of $2B$. In other words, the throughput is saturated by the control channel's bandwidth. The bottleneck problem will become more serious as the number of data channels, data rates, or node's density increases [3].

(2) *Data channel selection problem*: The *DCC* approach allows each sender–receiver pair to select a data channel in an on-demand matter. But, if the selection strategy were not carefully designed in concern with the reusability of channels, the throughput would be lower. As shown in Fig. 2, assume that the sets of channels which are free to nodes y, x, u, and v are as specified in Fig. 2(b). Clearly, the transmissions from y to x and from u to v can be active simultaneously on different channels. But, if nodes u and v do not consider channel statuses at nearby nodes, they may select ch_2 in prior to nodes y and x, further degrading the throughput from $2B$ to B.

In this paper, we propose a new MMAC protocol to resolve the two challenges in the *DCC* approach, composing of the following three components:(1) *Control initiation-time prediction* (*CIP*) reduces the control overhead by properly predicting the initiation time of each control process to avoid unsuccessful channel coordination; (2) *Dynamic data-flow control* (*DDC*) dynamically adjusts the amount of flow in each data transmission to balance the congestion in the control and data

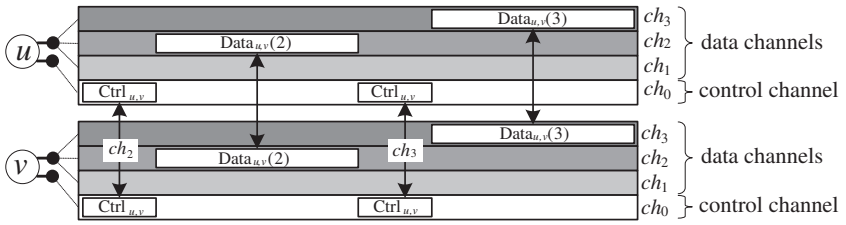


Fig. 1. Dedicated control channel approach.

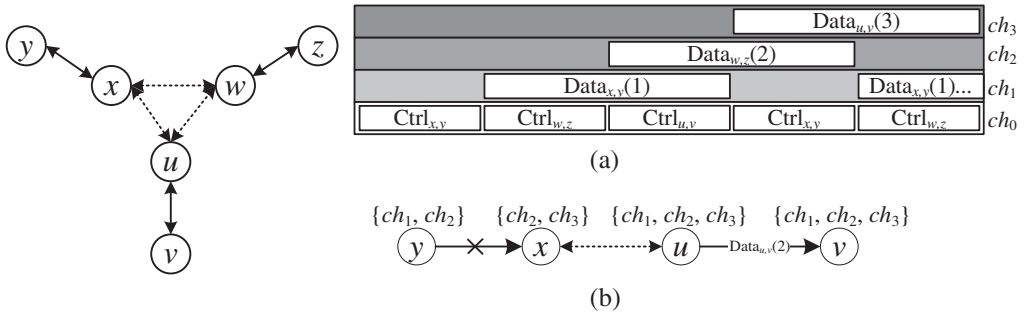


Fig. 2. (a) Control channel bottleneck problem and (b) data channel selection problem.

channels. (3) *Enhanced channel selection* (*ECS*) improves the reusability of data channels by selecting a channel that adds the least total delay to the starting times of the transmissions at nearby nodes. The design of these components is primarily based on exploiting the release times of channels and interfaces. Hence, we named this protocol the *Release-Time-Based MMAC* (*RTBM*).

The rest of this article is organized as follows: In Section 2, we compare different types of MMAC approaches and review existing protocols related to the *DCC* approach. Section 3 describes the basic operation of the *RTBM* protocol. The detailed design of each component is present in Section 4. In Section 5, we conduct a series of simulations to evaluate the performance. Concluding remarks are given in the last section.

2. MMAC approaches and related works

This section consists of two parts. First, we compare the pros and cons of different MMAC approaches. Next, existing protocols related to the *DCC* approach are reviewed.

2.1. MMAC approaches

A variety of MMAC protocols has been proposed in the literature. According to the way of coordinating data channels [1,2,23], existing protocols can be classified into the *channel fixed*, *receiver based*, *hybrid*, *spite control phase*, *dedicated control channel, and single/parallel rendezvous* approaches.

In the *channel fixed approach* [11,12], each interface is fixed on a channel permanently or for a long period of time. If a node *A* wants to communicate with a neighboring node *B*, they must have some interfaces fixed on the same channel. Then, A can send packet (e.g. RTS/CTS/DATA) directly to *B*, without additional control overheads to find a common channel. Besides, since the channels are fixed, contention within each group of interfaces on the same channel can be resolved by a standard MAC (e.g. 802.11 DCF). However, the fixed structure also limits the ability of using diverse channels. The number of channels that can be used by a node is limited by the number of its interfaces. In addition, if there is no common channel shared by two adjacent nodes, their traffic has to be relay through a longer path.

In contrast, the *receiver based approach* allows each node to utilize diverse channels with a single interface [3,13]. Each node is specified a channel in advance. For communication, a node u can connect with any nearby node v by just turning its interface the channel specified to v. However, since nodes are always fixed on the specified channels, some control signals may lose. As shown in Fig. 3(a), nodes *A*, *B*, *C*, and *D* are specified ch_1, ch_2, ch_2, and ch_3, respectively. At the beginning, node *A* intents to transmit to *B*. So, it exchanges the RTS-CTS with *B* using *B*'s channel. However, at the moment, node *C* has turned its interface to ch_3 for transmitting to *D*. Thus, node *C* cannot be aware of the CTS from *B*. As a result, after the current transmission, node *C* may content for ch_2 and incur a collision at *B*. It is the so called *multi-channel hidden terminal problem* [2]. Even worse, the lost signals may cause link failure. In Fig. 3(b), node *C* is sending requests to *B* using *B*'s channel. But, node *B* has turned its interface to ch_1 so that it cannot be aware of the request from *C*. Thus, node C may keep sending the RTSs until it falsely concludes that its link to *B* has broken. It is the so-called *deafness problem* [2].

The *hybrid approach* employs two interfaces to overcome the problems in both channel fixed and receiver based approaches [23,24]. One interface is fixed on a specified channel for receiving tasks, and the other interface can be dynamically switched on the channel of the fixed interface of the intended receiver. In this way a node can utilized diverse channel via its switchable interface and keep trace of control packets via its fixed interface. Moreover, singe channels of fixed interfaces are rarely changes, a channel switching can be made immediately without any coordination. However, this approach has two drawbacks. First, although a node can utilize more diverse channels, not limited by the number of its interface, the commu-

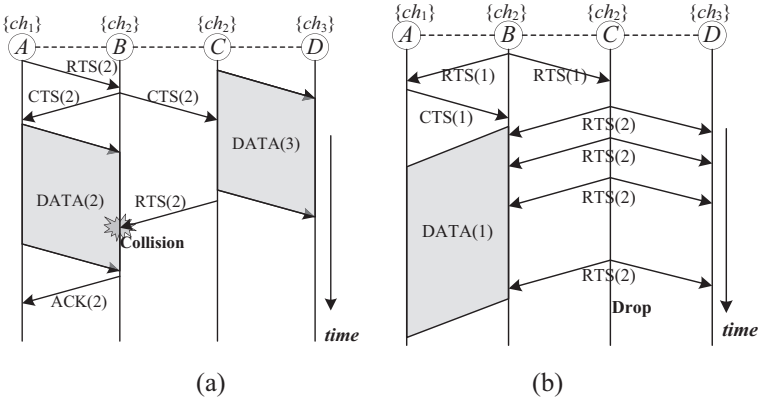


Fig. 3. Receiver based approach (a) multi-channel hidden terminal problem and (b) deafness problem.

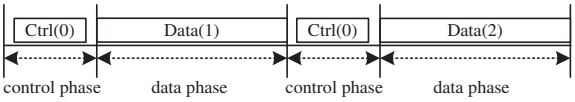


Fig. 4. Split control phase approach.

nication channel between two nodes is still fixed, which is not flexible when the channel of the intended receiver is highly interfered. Second, a link-layer broadcast is hard to be implemented. A node should broadcast the same message on all channels to ensure the delivery.

The *split control phase approach* divides each beacon interval into a *control phase* and a *data phase* [14,15]. As shown in Fig. 4, all nodes periodically rendezvous to a common channel in the control phase to sense carriers and coordinate channels for the subsequent transmissions in the data phase. In this approach, nodes have the most flexibility to use diverse channels during the data phase. Besides, the exchange of control packets and link-layer broadcast can be easily achieved during the control phase. Nonetheless, the phase alignment relies on strict time synchronization so that it is hard to implement in practice.

The *dedicated control channel approach* does not require any time synchronization mechanism. With the control interface, each node can access the control channel at anytime. However, as mentioned in Section 1, the throughput could be limited by the bandwidth of control channel and influenced by the selection of data channels. Besides, the dynamic channel selection means that more control overheads are required for the channel coordination. Hence, in this paper we focus on solving these challenges.

Finally, in the *common rendezvous approach* [16], nodes not exchanging data hop through all channels synchronously. A pair of nodes stops hopping as soon as they make an agreement for transmission and rejoin the common hopping pattern subsequently after transmission ends. In comparison with the *DCC*, this approach can make use of all the channels for data exchange and requires only one interface per node. However, it does not resolve the control bottleneck problem since only one pair of nodes can make an agreement at the same time. The *parallel rendezvous approach* [17,18] overcomes this problem by assigning different hopping sequences to nodes. In SSCH [17], each node picks multiple sequences and follows them in a time-multiplexed manner. When node *A* wants to talk to node *B*, *A* waits until it is on the same channel as *B*. The McMAC [18] improves SSCH by allowing a sender to temporally hop to the sequence of the receiver to avoid waiting. The major problem in these rendezvous approaches is that they may incur large switching delay for channel hopping, and each node requires synchronization mechanisms to track the hopping sequence of the others. Besides, a link-layer broadcast is hard to be implemented in a hopping fashion.

2.2. Existing protocols for DCC approach

The concept of using separated channels or special devices (e.g. busy tones) to improve medium access control has been extensively studied in works such as [19–22], but these researches consider only one data channel, i.e. designed for single-channel MAC. The first MMAC protocol under the *DCC* approach was presented in [3] and named the *dynamic channel assignment* (*DCA*). In this protocol, each node maintains a *free channel list* (*FCL*) to record unused data channels. As a node u intends to transmit to a node v, it first sends a RTS to v carrying its *FCL*. Then, node v compares the received *FCL* with its own *FCL* to select a common free channel. If there is any, the selected channel will be replied to u using a CTS. Once received the CTS, node u emits a reserve-to-send (RES) to inhibit other nodes from using the same channel. Meanwhile, nodes u and v can start to exchange the DATA and ACK frames on the selected channel. Integration with the power control technique was proposed in [4].

Compared with the IEEE 802.11 DCF, the *DCA* needs an additional control frame (e.g. RES) to reserve the channel selected at the receiver's side. To avoid such overhead, the protocol in [5] suggests that each sender can firstly propose a free channel in the RTS. If the channel is free to the receiver, the data transmission can be started immediately upon received the CTS. Otherwise, it follows the same way in [3]. A similar protocol appears in [6], where the proposed channel in the RTS will be replied by a reply-to-RTS (RRTS) that indicates whether the channel is free or not. The negotiation will continue until a common free channel is found. Nonetheless, these protocols may spend more control bandwidth if the proposed channel is not accepted by the receiver.

Another way to relieve the bottleneck problem is by using multiple control channels. Koubaa [7] showed that the number of control channels required to achieve the maximal throughput is a function of the available channels and packet size. For instance, with 12 channels and packet size of 1024 bytes, providing three control channels is optimal. Likewise, the protocol in [8] employs an extra channel for replaying ACKs to improve the channel reusability. By replying the ACK in a separated channel, a sender can be active simultaneously with its hidden terminals. Although using multiple control channels is beneficial, coordinating on different channels could be more complicated.

The channel selection strategy in the *DCA* [3] is simply to find a communicable channel. Each sender–receiver pair coordinates a channel that is free at the two sides. If there are multiple choices, one channel will be chose at random. The strategy was slightly improved in [9,10]. In [9], the channel with the least received power will be chosen to avoid potential interference. Similarly, in [10], the most robust channel will be selected according to the carrier-to-interference ratio. In other words, [9,10] are concerned about not only the communicability, but also the *quality* of the selected channel.

3. Protocol description

The *RTBM* is primarily based on the *DCA* protocol [3]. Similarly to the *DCA*, each sender–receiver pair has to exchange the RTS-CTS-RES sequence on the control channel to coordinate and reserve a data channel for the subsequent transmission of the DATA and ACK messages. The *RTBM* further incorporates with the following components to resolve the control channel bottleneck and data channel selection problems in the *DCC* approach:

(1) *Control initial-time prediction* (*CIP*): In the *DCA* [3], each sender has to initiate a control process (e.g. the RES-CTS-RES) with the intended receiver to coordinate a data channel that is free at two sides. If the coordination succeeds, the data transmission (e.g. the DATA-ACK) can be started on the selected channel. However, if the coordination fails, the expensed control bandwidth and time (including backoff timer, transmission time, inter-frame spaces, propagation delay, and processing time) are wasted. The *CIP* can avoid unsuccessful channel coordination by properly predicting the initiation time of each control process. The prediction will jointly consider the channels and interfaces statuses at the sender and receiver.
(2) *Dynamic data-flow control*: (*DDC*): As shown in Fig. 2(b), the bottleneck problem from the common control channel would lower down the utilization of data channels. To breakthrough this limitation, an intuitive way is to expend the *data flow* (i.e. the number of packets to be sent) for each control process to increase the bandwidth usage [20]. However, if too many packets were transmitted with only a few control processes, the data channels could be occupied by some node pairs for a long period of time, which may instead incur unfairness problem to other competitors who are intended to access the data channels. Ideally, the above problems can be optimally solved by adjusting the data flow such that both the control and data channels are fully utilized. For example, in Fig. 2(b), if node u sent 1.5 times of packets to node v by each control process, all channels can be fully exploited. However, due to the dynamicity in wireless environments, such as the variation in network traffic or topology, the optimal setting would be varied from time to time. The *DDC* component can dynamically adjust the amount of flow in each data transmission to balance the congestion in the control and data channels.
(3) *Enhanced channel selection*: (*ECS*): The selection strategy in the *DCA* [3] is simply to find a communicable data channel that is free to sender and receiver. If there are multiple choices, one channel will be chose at random. Succeeding works in [9,10] also consider the quality of the selected channel, but none of them concerns the influence to nearby transmissions. The *ECS* can improve the reusability of data channels. Each pair will cooperatively coordinate a free data channel that will add the least total delay to the starting times of potential transmissions at nearby nodes. In other words, the selected channel is not only communicable, but also has the least influence to the transmission opportunities of nearby nodes.

In the following, we first define the statuses and symbols used in this protocol. Next, the basic operation is described. We will focus here on how the *RTBM* interacts with the three components and defined statuses. The detail design of each component will be presented in the next section.

3.1. Statuses and symbols

We assume that the network has a set of orthogonal channels $\mathscr{H} = \{h | h = 0, 1, 2, \ldots, H\}$. Each node has a control interface and a data interface. The first channel ($h = 0$) is for control purpose, and the remaining channels ($h = 1, 2, \ldots, H$) can be used for data transmission. A channel (or interface) is *released* when it can be used for a transmission or reception. Each node u has the following statues:

- $ch\_rel\_time(u, h)$: the *release time* of the hth channel at u, $h = 0, 1, \ldots, H$;
- $if\_rel\_time(u)$: the *release time* of the data interface at u.

In addition, each node u maintains a *channel release time table* (CRT_u) and an *interface release time vector* (IRV_u) to record the channels' and interfaces' statuses at nearby nodes. The fields $CRT_u(v, h)$ and $IRV_u(v)$, respectively, keep track of the latest statuses about the $ch\_rel\_time(v, h)$ and $if\_rel\_time(v)$ for each neighboring node v. Moreover, a separate queue is created for each 1-hop destination. The purpose is to avoid head-of-line blocking if two or more packets need to be sent for the same node. Other time symbols that will be used in our protocol are listed in Table 1.

3.2. Basic operation

The basic operation of the *RTBM* is illustrated in Fig. 5. Consider node u. Let us see how node u operates with other nodes in this protocol. In the beginning, node u creates a separate queue for nodes a and v. To avoid starvation, when some packets arrived in queues the destination with the oldest packet will be chosen as the target. Assume that node u has decided to transmit to node v. First of all, it applies the *CIP* to predict a *control initiation time* for v, denoted as $ctrl\_ini\_time(u, v)$. The

Table 1
Time symbols used in the *RTBM* protocol.

Symbols	Meanings
T_{curr}	Current time of a node
T_{DATA_i}	Time to transmit the i-th data packet in a queue
T_{RTS}	Time to transmit a RTS frame
T_{CTS}	Time to transmit a CTS frame
T_{RES}	Time to transmit a RES frame
T_{ACK}	Time to transmit an ACK frame
BF	Remaining backoff time
τ	Maximal propagation delay

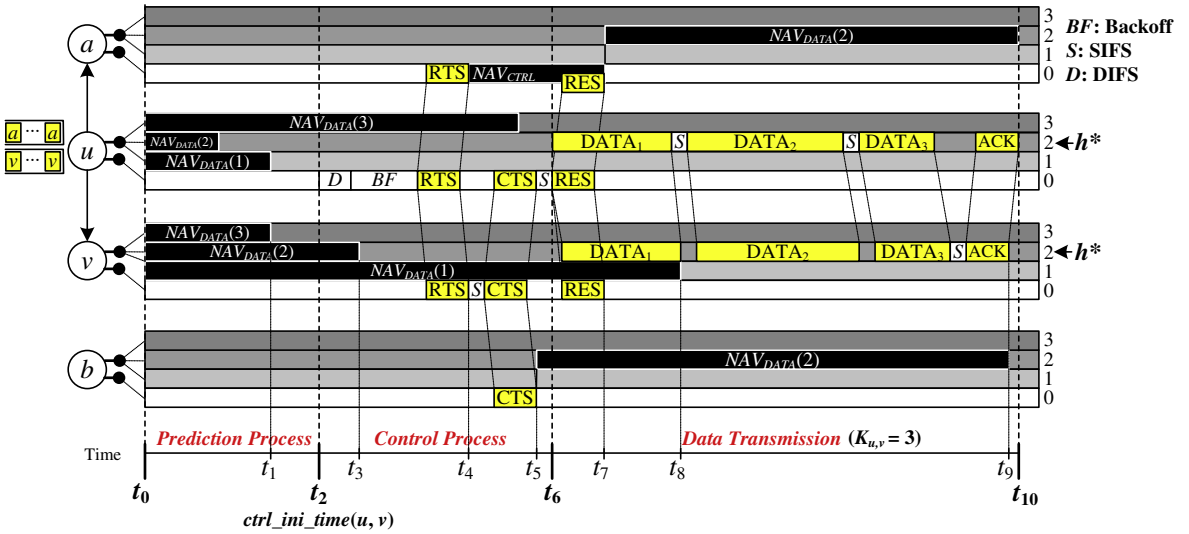


Fig. 5. Basic operation of the *RTBM* protocol.

time indicates when node u can successfully initiate a control process with v. As shown in Fig. 5, $ctrl\_ini\_time(u, v) = t_2$. The prediction process will continue before the control process is actually started.

At $ctrl\_ini\_time(u, v)$, node u starts the following control process. It first applies the *DDC* to determine the number of data frames to be sent for v, denoted as $K_{u,v}$. According to the $K_{u,v}$, the network allocation vector required to exchange the DATA and ACK messages can be set as

$$NAV_{DATA} = \sum_{i=1}^{K_{u,v}} (T_{DATA_i} + SIFS) + T_{ACK} + 2\tau,$$

where T_{DATA_i} is the time to transmit the ith data packet in the queue for v. As shown in Fig. 5, $K_{u,v} = 3$ and $NAV_{DATA} = t_{10} - t_6$. So, node u will transmit the first three packets to v during t_6 to t_{10}. Next, node u applies the *ECS* to evaluate the cost for transmitting on each data channel h. The cost, denoted as $\Delta_u(h)$, is the total increment to the starting times of possible transmissions at u's nearby nodes as if node u transmitted on channel h for a period of NAV_{DATA}. Then, following the IEEE 802.11 backoff mechanism, if there was no carrier on the control channel in a *DIFS* plus the *BF*, node u sends a RTS to v, containing the NAV_{DATA} and the $\Delta_u(h)$, for any $h = 1, 2, \ldots, H$.

Once received the RTS, node v also applies the *ECS* to evaluate the cost $\Delta_v(h)$, for each data channel h, i.e. the total increment to the starting times of possible transmissions at v's nearby nodes as if node v transmitted on channel h for a period of $NAV_{DATA} - \tau$. Combining the costs evaluated from the two sides, the total cost for communicating between u and v on a channel h, denoted as $\Delta_{u,v}(h)$, is the sum of the $\Delta_u(h)$ and $\Delta_v(h)$. The data channel that will be released to u and v and has the least $\Delta_{u,v}(h)$ will be selected. If such a channel can be found, denoted as h^*, node v has to update the following statues for the forthcoming data transmission on h^* such that

$$if\_rel\_time(v) = ch\_rel\_time(v, h^*) = T_{curr} + 2SIFS + T_{CTS} + NAV_{DATA}.$$

For example, at $T_{curr} = t_4$, node v sets $if\_rel\_time(v) = t_9$, and $ch\_rel\_time(v, h^*) = ch\_rel\_time(v, 2) = t_9$. Besides, node v sets two timers as follows in its CTS.

$$if\_duration(v) = \max\{0, if\_rel\_time(v) - T_{curr} - (SIFS + T_{CTS} + \tau)\},$$

$$ch\_duration(v,h) = \max\{0, ch\_rel\_time(v,h) - T_{curr} - (SIFS + T_{CTS} + \tau)\}, \quad h = 1,2,\ldots,H.$$

The $if\_duration(v)(ch\_duration(v,h))$ indicates the amount of time the data interface (data channel h) of node v will be reserved. For example, at $T_{curr} = t_4$, node v sets

$$if\_duration(v) = \max\{0, t_9 - t_4 - (SIFS + T_{CTS} + \tau)\} = t_9 - t_5,$$

$$ch\_duration(v,1) = \max\{0, t_8 - t_4 - (SIFS + T_{CTS} + \tau)\} = t_8 - t_5$$

$$ch\_duration(v,2) = \max\{0, t_9 - t_4 - (SIF + T_{CTS} + \tau)\} = t_9 - t_5,$$

$$ch\_duration(v,3) = \max\{0, t_1 - t_4 - (SIFS + T_{CTS} + \tau)\} = 0.$$

After waiting a *SIFS*, node v replies a CTS to u, containing the h^* (if there is any), $if\_duration(v)$, and $ch\_duration(v,h)$, for any $h = 1, 2, \ldots, H$.

Once received the CTS, if a channel h^* was indicated, node u also updates

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

for the forthcoming data transmission on h^* and sets two timers as follows in its RES.

$$if\_duration(u) = \max\{0, if\_rel\_time(u) - T_{curr} - (SIFS + T_{RES} + \tau)\},$$

$$ch\_duration(u,h) = max\{0, ch\_rel\_time(u,h) - T_{curr} - (SIFS + T_{RES} + \tau)\}, \quad h = 1,2,\ldots,H.$$

After waiting a *SIFS*, the h^* is rebroadcasted to nearby nodes along with a RES to reserve the channel. Meanwhile, node u can start to send $K_{u,v}$ packets to node v via channel h^*. Finally, node v replies an ACK to node u at the end of the data transmission.

On the other hand, once an irrelevant node x (e.g. nodes a and b) received the CTS or RES from a node y (e.g. nodes u and v), if a channel, i.e. h^*, was indicated inside, node x has to inhibit itself from using the same data channel by setting

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

Note that the NAV_{DATA} has been implicated in both the $ch\_duration(v,h^*)$ and $ch\_duration(u,h^*)$. Hence, it is neither in the CTS nor in the RES.

Moreover, to prevent nodes from disrupting the control process between u and v, an extra *NAV* is employed on the control channel. More precisely, once an irrelevant node x (e.g. node a) received the RTS from u, it has to block its control channel for a period of time by setting

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

where $NAV_{CTRL} = 2SIFS + T_{CTS} + T_{RES} + 2\tau$, specifying the amount of time the control channel will be for the RTS-CTS-RES dialogue.

Lastly, to maintain the statuses of nearby nodes, once a node x (e.g. nodes a, b, u, and v) receives a CTS or RES from a node y (e.g. nodes u and v), it has to update its *channel release time table* (CRT_x) and its *interface release time vector* (IRV_x) as follows.

$$IRV_x(V) = T_{curr} + if\_duration(y),$$

$$CRT_x(y,h) = T_{curr} + ch\_duration(y,h), \quad \text{for any} \quad h = 1,2,\ldots,H.$$

The above processes are in normal situation which could be interrupted in some circumstances. First, node u detected some control signals before the *BF* expires. In this case, the RTS will not be sent out by u. Second, the RTS or CTS was collided or the control channel of v was blocked by a NAV_{CTRL}. In this case, node u will not receive the CTS from v within the timeout period of $SIFS + T_{CTS} + 2\tau$. Third, there was no data channel h^* in indicated in the CTS from v. In this case, the data transmission cannot be started by u. When some of these cases happened, the following process should be done: If the retry limited is not reached, node u has to terminate the current control process, apply again the *CIP*, and restart the next attempt at the new $ctrl\_ini\_time(u, v)$; otherwise, node u has to abort any process with v an select a new target from its queues.

4. Components design

In this section, we present the detail design of the three components in our protocol.

4.1. Control initiation-time prediction

The *CIP* is designed to reduce unsuccessful channel coordination that would incur redundant control overhead. It is achieved by properly predicting the initiation time of each control process, based on the information about the release times

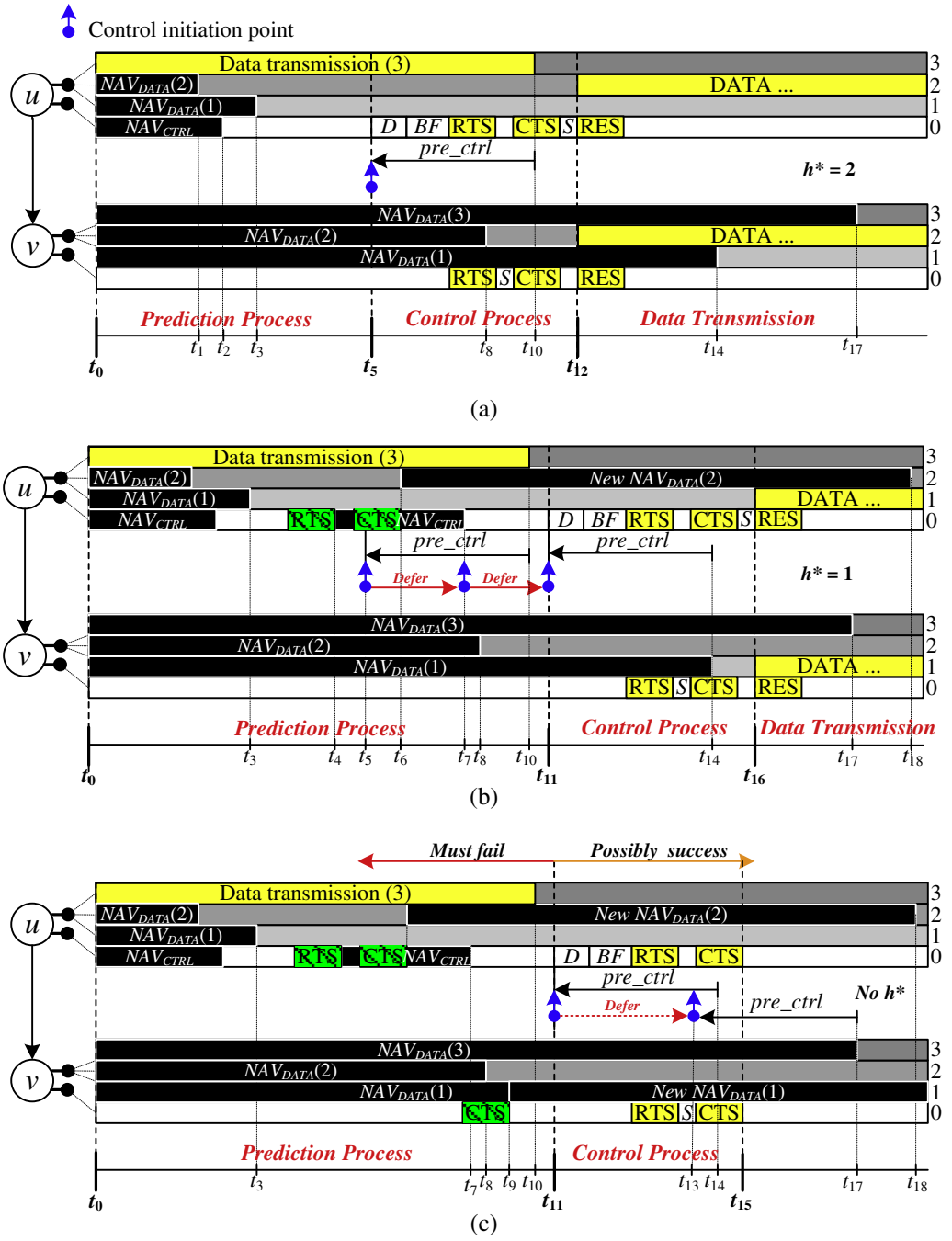


Fig. 6. Control initiation prediction process (a) without deferral, (b) with deferral and (c) invalid scenario.

of channels and interfaces at sender and receiver. To explain the idea, let us see the example in Fig. 6(a).[1] In the beginning, node u has some packets for v at t_0. At the moment, the release times of ch_1, ch_2, and ch_3 at u and v are (t_3, t_1, t_{10}) and (t_{14}, t_8, t_{17}), respectively. We know that two nodes can communicate with each other only if at least one data channel has been released to both of them. Therefore, node u cannot send any data frame to v before $t_8 = \min\{\max\{t_3, t_{14}\}, \max\{t_1, t_8\}, \max\{t_{10}, t_{17}\}\}$, i.e., the release time of ch_2. Moreover, no data frame can be transmitted until the data interfaces of both u and v have been released. With the two considerations, we define the *link release time* of u and v as

[1] To simplify our presentation, the propagation τ will not be drawn in the hereafter figures.

$$link\_rel\_time(u,v) = \max\left\{\begin{array}{l}\min\limits_{h=1,\ldots,H}\left\{\max\left\{\begin{array}{l}ch\_rel\_time(u,h),\\ CRT_u(v,h)\end{array}\right\}\right\},\\ if\_rel\_time(u),\\ IRV_u(v)\end{array}\right\},$$

which means that all resources required for communicating on the virtual link between u and v will be released at *link_rel_time*(u, v). In Fig. 6(a), the link release time of u and v at t_0 is $t_{10} = \max\{t_8, t_{10}, t_0\}$. Clearly, it is the earliest possible time that u can start a data transmission with v.

With the observation, now we discuss when node u can initiate the control process with v. In the *DCC* approach, since control frames (e.g. RTS/CTS/RES) and data frames (e.g. DATA/ACK) are exchanged using different interfaces and channels, a control process can be started earlier before the virtual link (data channels and data interfaces) is released. Besides, the data frames can be sent out once node u has received the CTS from v for a *SIFS*. Therefore, if node u intends to transmit at the earliest time t_{10}, it should initiate the control process in advance at $t_{10} - (DIFS + BF + T_{RTS} + T_{CTS} + 2SIFS + 2\tau)$. However, the backoff timer *BF* should be removed from the composition of $(DIFS + BF + T_{RTS} + T_{CTS} + 2SIFS + 2\tau)$. Otherwise, other senders waiting for the same resource (u's data interface) may send their RTSs at the same time with u's RTS, and thus incur collision. In addition, node u cannot perform any control process until the control channel is released. Combining these facts, the *control initiation time of u and v* is defined as

$$ctrl\_ini\_time(u,v) = \max\left\{\begin{array}{l}link\_rel\_time(u,v) - pre\_ctrl,\\ ch\_rel\_time(u,0)\end{array}\right\},$$

where $pre\_ctrl = DIFS + T_{RTS} + T_{CTS} + 2SIFS + 2\tau$, denoting the preprocessing time. In this example, node u can apply this function at t_0 and predict that the initiation time is $t_5 = \max\{t_{10} - pre\_ctrl, t_2\}$.

Note that, before the control process is actually started, the predicted time could be updated if the *ch_rel_time*$(u, 0)$ or *link_rel_time*(u, v) is changed. When such event occurs, the control process is deferred to the updated initiation time. For example, in Fig. 6(b), node u received a RTS at t_4 that changes the *ch_rel_time*$(u, 0)$ to t_7. So, the *ctrl_ini_time*(u, v) is deferred from t_5 to t_7. Furthermore, node u received a CTS at t_6, indicating that ch_2 will be released at t_{18}. Hence, the *link_rel_time*(u, v) is changed from t_{10} to t_{14}. Accordingly, the *ctrl_ini_time*(u, v) is further deferred from t_7 to t_{11}.

In addition, since the records in the CRT_u and IRV_u are not always the newest, the predicted time could be invalided. As shown in Fig. 6(c), node v received a CTS with an updated *ch_duration*$(v, 1)$ at t_9 and the *ch_rel_time*$(v, 1)$ is changed accordingly. Normally, the control initiation time should be deferred from t_{11} to t_{13} to response to the change. But, since node v does not send any control frame during t_9 to t_{11}, the change is invisible to u. As a result, node u will still initiate the control process at t_{11} and the control process may fail. However, the *CIP* does ensure that any control process initiated before t_{11} must fail, because for any channel h (or interface), if it is not released at $CRT_u(v, h)$ (or $IRV_u(v)$) to u, it is also not released at *ch_rel_time*(v, h) (or *if_rel_time*(v)) to v. Thus, the *CIP* can help nodes to avoid unsuccessful channel coordination.

4.2. Dynamic data-flow control

The *DDC* can dynamically adjust the number of data packets being sent in each data transmission to balance the congestion in the control and data channels. The *DCC* maintains a variable $K_{u,v}$ for each 1-hop node v specifying the number of data packets that will be sent in the next transmission from nodes u to v. $K_{u,v}$ is initiated as 1 and will be dynamically adjusted according to the idle statuses on the control and data interfaces.

Take a look at Fig. 7 to explain this idea. There are three sender–receiver pairs of $\{u, v\}$, $\{x, y\}$, and $\{w, z\}$ with data flows sustained during $[t_0, t_3]$, $[t_1, t_3]$, and $[t_1, t_2]$, respectively. Besides, nodes u and v are in the interference range of nodes x and w,

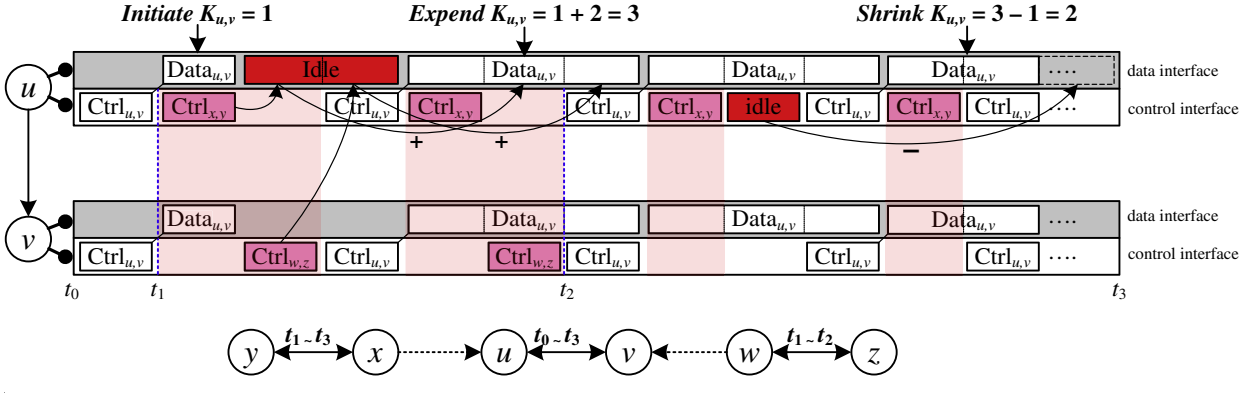


Fig. 7. Basic concept of the *DDC* component.

receptively. Assuming that the time required for a control process is equal to the time required for sending one data packet, we show how *DDC* adjusts $K_{u,v}$. In the beginning, $K_{u,v} = 1$. Thus, node u sends one data packet to v. Then, node u intends to send the next packet to v. However, at this moment, the control processes for other two flows have started, so node u cannot initiate a control process immediately with v. Consequently, u's data interface experiences a period of idle status before the next data transmission starts. In order to mitigate such influence from control channel, the *DDC* expends $K_{u,v}$ in proportion to the idle time experienced from u's data interface (i.e. $K_{u,v} = 1 + 2 = 3$) to increase the utilization of data channels for the subsequent data transmission to v. As shown in this example, by the end of the second data transmission, the next data transmission can be started immediately without any idling. Nonetheless, during the third data transmission, since the data flow between w and z has finished, the control channel becomes less congested at v, resulting in a period of idle status in u's control interface before the next control process starts. In order to reflect such fact, the *DCC* shrinks $K_{u,v}$ in proportion to the idle time experimented from u's control interface (i.e. $K_{u,v} = 3 - 1 = 2$). Finally, $K_{u,v}$ converges to 2 and the congestion in control and data channels are balanced. The process will continue if any further change occurred.

In summary, $K_{u,v}$ is adjusted according to two idle timers: the *data interface idle time*, denoted as $dif\_idle\_time(u, v)$, and the *control interface idle time*, denoted as $cif\_idle\_time(u, v)$. Therefore, below we give a more realistic example to show how an idle status occurs in the *RTBM*. Then, the two idle timers and adjusting rule are formally defined. Lastly, we provide an online algorithm for efficient maintenance.

In Fig. 8(a), node u has some packets for v at t_0. At the moment, node u finds that the link release time $link\_rel\_time(u, v) = t_4$ and control initial time $ctrl\_rel\_time(u, v) = t_4 - pre\_ctrl = t_2$. Ideally, if there is no contention before t_2, the data transmission can start at $t_5 = t_2 + pre\_ctrl + BF$ (recall that *BF* is not in the *pre_ctrl*). However, at t_1 an irrelevant RTS is detected, so node u has to update $ctrl\_ini\_time(u, v) = \max\{t_4 - pre\_ctrl, t_1 + NAV_{CTRL}\} = \max\{t_2, t_3\} = t_3$. Consequently, the data transmission should be deferred from t_5 to $t_7 = t_3 + pre\_ctrl + BF$. The deferral will result in a period of idle status from t_5 to t_7 on u's data interface. Therefore, we can estimate that $dif\_dile\_time(u, v) = t_7 - t_5$ at this time point.

Then, the control process starts at t_3 (see Fig. 7(b)). During the *BF*, another RTS is detected, so node u has to suspend the current process and update the new control initiation time to t_8. Consequently, the data transmission should be further deferred from t_7 to $t_{11} = t_8 + pre\_ctrl + BF$, resulting in a longer period of idle status on u's data interface from $[t_5, t_7]$ to $[t_5, t_{11}]$. However, during the sub period $[t_6, t_9]$, node u cannot transmit any data to node v even if the control channel is not congested, since all data channels (ch_1 and ch_3 of node u and ch_2 of node v) are blocked by some NAVs in this time interval. Thus, the $dif\_idle\_time(u, v)$ should be set as $dif\_idle\_time(u, v) = (t_{11} - t_5) - (t_9 - t_6)$ to reflect the deferral purely incurred by the control channel congestion. We call the time interval $[t_6, t_9]$ the *non-idle time*.

After a *DIFS* and *BF*, node u sends a RTS and waits for the CTS from v (see Fig. 8(c)). However, before the RTS arrived, the control channel of v has been blocked by another RTS so that u cannot obtain any reply from v before the end of the time out period at t_{10}. As a result, the starting time of the data transmission is further deferred from t_{11} to t_{12}. That is, the $dif\_idle\_time(u, v)$ should be updated as $(t_{12} - t_5) - (t_9 - t_6)$. This example shows that the idle status of u's data interface could be incurred by the control congestion at both node u and node v.

Continuing the example, in Fig. 8(d), the data transmission starts at t_{12} and ends at $t_{17} = t_{12} + NAV_{DATA}$. During this period, the control interface of u is released after sending the RES and reserved for the next control process after $t_{16} = t_{17} - pre\_ctrl$. In addition, the control channel is contended during the sub-period from t_{14} to t_{15}. Therefore, the control interface idle time is the sum of the two periods from t_{13} to t_{14} and from t_{15} to t_{16}, i.e. $cif\_idle\_time(u, v) = (t_{16} - t_{15}) + (t_{14} - t_{13})$.

Now, we formally define the two idle timers based on the above observations. Let $data\_tx\_time^*(u, v)$ and $data\_tx\_time(u, v)$ stand for the *earliest* and the *actual* starting times of a data transmission from u to v, respectively. At any time t, the data interface of u is *idle* if and only if

(i) $data\_tx\_time^*(u, v) \leqslant t < data\_tx\_time(u, v)$;
(ii) $link\_rel\_time(u, v) < t$.

For a transmission from u to v, $dif\_idle\_time(u, v)$ is the total time satisfying (i) and (ii). In Fig. 8, $data\_tx\_time^*(u, v) = t_5$, and $data\_tx\_time(u, v) = t_7$, t_{11}, and t_{12}, respectively, in Figs. 8(a)–(c). The period of time not satisfying (ii) is $[t_6, t_9]$, i.e. the non-idle time. Similarly, at any time t, the control interface of u is *idle* if and only if

(i) $data\_tx\_time(u, v) + T_{RES} + \tau < t < data\_tx\_time(u, v) + NAV_{DATA} - pre\_ctrl$;
(ii) $ch\_rel\_time(u, 0) - T_{type} < t$, where *type* is the type of the received control frame.

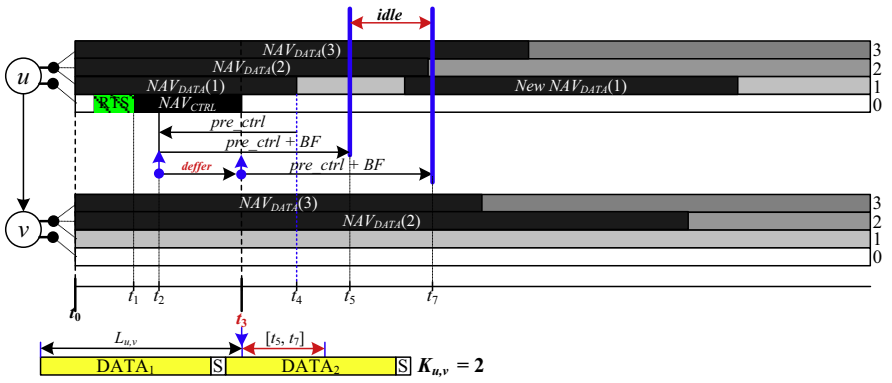
For a transmission from u to v, the $cif\_idle\_time(u, v)$ is the total time satisfying (iii) and (iv).

With the two idle timers, at the start point of each control process (i.e. t_3, t_8, and t_{10}), $K_{u,v}$ can be updated as

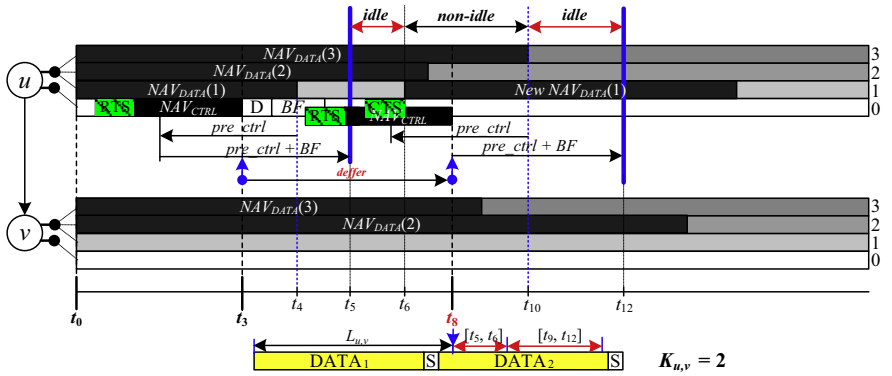
$$K_{u,v} = \min_{1 \leqslant k \leqslant Q_v} \left\{ \sum_{i=1}^{k} (T_{DATA_i} + SIFS) \geqslant L_{u,v} + dif\_dile\_time(u, v) \right\},$$

where Q_v is the number of data packets remaining for v, and $L_{u,v}$ is the time of the previous data transmission. As shown in Fig. 8(c), $K_{u,v} = 3$ at t_{10}, since

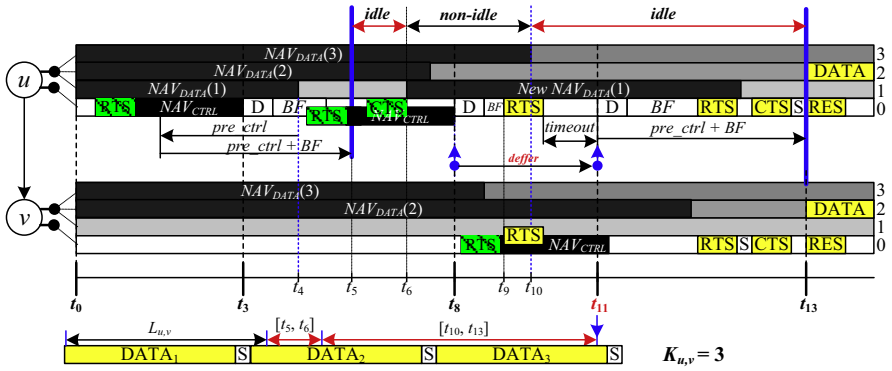
$$(T_{DATA_1} + SIFS) + (T_{DATA_2} + SIFS) + (T_{DATA_3} + SIFS) \geqslant L_{u,v} + (t_{12} - t_9) + (t_6 - t_5).$$



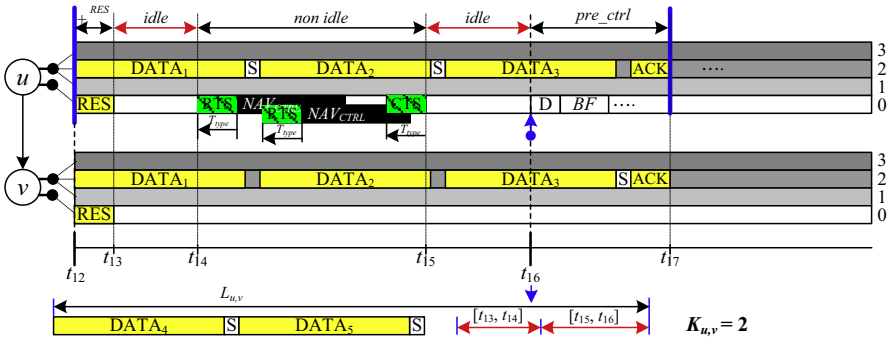
(a)



(b)



(c)



(d)

Fig. 8. Dynamic Data Aggregation (a) expending at t_3, (b) expending at t_8, (c) expending at t_{10} and (d) shrinking at t_{17}.

On the other hand, at the end point of each data transmission (i.e. t_{17}), $K_{u,v}$ can be updated

$$K_{u,v} = \max_{1 \leqslant k \leqslant Q_v} \left\{ \sum_{i=1}^{k} (T_{DATA_i} + SIFS) < L_{u,v} - cif\_dile\_time(u, v) \right\}.$$

Note that $L_{u,v}$ should be updated according to the time of the previous data transmission when used. As shown in Fig. 8(d), $L_{u,v} = t_{17} - t_{12}$, and $K_{u,v} = 2$ since

$$(T_{DATA_4} + SIFS) + (T_{DATA_5} + SIFS) < L_{u,v} - (t_{16} - t_{15}) + (t_{14} - t_{13}).$$

In accordant with the above definitions, an algorithm is presented below to maintain the $K_{u,v}$. Additional time symbols used in this algorithm are listed in Table 2. The corresponding values for Fig. 8 are shown in Table 3. The algorithm is designed in an online fashion and takes only constant time in each step. Thus, it is very efficient and practical.

Online algorithm: *DDC* Component
Initial:

$K_{u,v} = 1$; $L_{u,v} = T_{DATAi} + SIFS + T_{ACK} + 2\tau$;

Whenever node v is chosen as a target at T_{curr}*:*

$ctrl\_ini\_time^*(u, v) = \max\{T_{curr}, link\_rel\_time(u, v) - pre\_ctrl\}$;
$data\_tx\_time^*(u, v) = ctrl\_ini\_time^*(u, v) + pre\_ctrl + BF$;
$dif\_nid\_end(u, v) = data\_tx\_time^*(u, v)$;
$dif\_nid\_time(u, v) = 0$;

Before ctrl_ini_time(u, v):

Once the $link\_rel\_time(u, v)$ is deferred at T_{curr}, update
$dif\_nid\_time(u, v) = dif\_nid\_time(u, v) + link\_rel\_time(u, v) - \max\{T_{curr}, dif\_nid\_end(u, v)\}$;
$dif\_nid\_end(u, v) = \max\{data\_tx\_time^*(u, v), link\_rel\_time(u, v)\}$;

At ctrl_ini_time(u, v):

$data\_tx\_time(u, v) = ctrl\_ini\_time(u) + pre\_ctrl + BF$;
$dif\_idle\_time(u, v) = data\_tx\_time(u, v) - data\_tx\_time^*(u, v) - dif\_nid\_time(u, v)$;
$K_{u,v} = \min_{1 \leqslant k \leqslant Q_v} \{\sum_{i=1}^{k} (T_{DATA_i} + SIFS) \geqslant L_{u,v} + dif\_dile\_time(u, v)\}$;

At data_tx_time(u, v):

$cif\_nid\_time(u, v) = 0$;
$cif\_nid\_end(u, v) = data\_tx\_time(u, v) + T_{RES} + \tau$;

Before data_tx_time(u, v) + $NAV_{DATA} - pre\_ctrl$*:*

Once the $ch\_rel\_time(u, 0)$ is deferred at T_{curr}, update
$cif\_nid\_time(u, v) = cif\_nid\_time(u, v) + ch\_rel\_time(u, 0) - \max\{T_{curr} - T_{type}, cif\_nid\_end(u, v)\}$;
$cif\_nid\_end(u, v) = \max\{data\_tx\_time(u, v) + \sigma_{res}, ch\_rel\_time(u, 0)\}$;

At data_tx_time(u, v) + $NAV_{DATA} - pre\_ctrl$*:*

$cif\_idle\_time(u, v) = NAV_{DATA} - pre\_ctrl - cif\_nid\_time(u, v)$;
$L_{u,v} = \sum_{i=1}^{K_{u,v}} (T_{DATA_i} + SIFS) + T_{ACK} + 2\tau$;
$K_{u,v} = \max_{1 \leqslant k \leqslant Q_v} \{\sum_{i=1}^{k} (T_{DATA_i} + SIFS) < L_{u,v} - dif\_dile\_time(u, v)\}$;

4.3. Enhanced channel selection strategy

The *ECS* aims at improving the reusability of data channels. The main idea is based on exploiting the release times of channels and interfaces at neighboring nodes to select a free data channel that will cause the least influence to nearby transmissions.

Table 2
Time symbols in the online algorithm of the *DDC* component.

Status	Meaning
*ctrl_ini_time**(*u*, *v*)	Earliest control initiation time with *v*
dif_nid_time(*u*, *v*)	Non-idle time of *u*'s data interface before a data transmission to *v*
dif_nid_end(*u*, *v*)	End point of the non-idle status of *u*'s data interface before a data transmission to *v*
cif_nid_time(*u*, *v*)	Non-idle time of *u*'s control interface during a data transmission to *v*

Table 3
Corresponding values of the used statues for Fig. 8.

*ctrl_ini_time**(*u*, *v*); *ctrl_ini_time*(*u*, *v*)	*data_tx_time**(*u*, *v*); *data_tx_time*(*u*, *v*)	*dif_nid_time*(*u*, *v*); *cif_nid_time*(*u*, *v*)	*dif_nid_end*(*u*, *v*); *cif_nid_end*(*u*, *v*)	*dif_idle_time*(*u*, *v*) *cif_nid_time*(*u*, *v*)	$K_{u,v}$
t_2	t_5	0	t_5	0	1
t_3	t_7	0	t_5	$(t_7 - t_5)$	2
t_8	t_{11}	$(t_9 - t_6)$	t_9	$(t_{11} - t_5) - (t_9 - t_6)$	2
t_9	t_{12}	$(t_9 - t_6)$	t_9	$(t_{12} - t_5) - (t_9 - t_6)$	3
t_{17}	–	$(t_{15} - t_{14})$	t_{15}	$(t_{16} - t_{13}) - (t_{15} - t_{14})$	2

The concept is shown in Fig. 9. Nodes *a* and *c* are in the interference range of node *u*, and nodes *b* and *d* are in the interference range of node *v*. According to the specified NAVs, nodes *a*, *b*, *c*, and *d* cannot transmit to any other node, respectively, before t_6, t_9, t_2, and t_7, since prior to these time points, there is neither an interface nor a channel released (note that the interface of node *d* will be released by t_7). In other words, the earliest possible time to start a transmission from *a*, *b*, *c*, and *d* are t_6, t_9, t_2, and t_7, respectively. Assume that nodes *u* and *v* have decided to communicate on ch_1. See what happen to nearby nodes. Since ch_1 is not the first channel released to node *a*, node *a* can still transmit at t_6. Likewise, the earliest possible starting time of node *d* remains t_7. However, to nodes *c* and *d*, since ch_1 is no longer the first channel released to them, they cannot start any transmission before ch_2 and ch_3 are released to them at t_8 and t_9, respectively. As a result, the earliest possible starting times of *b* and *c* are increased (postponed). Our goal is to coordinate a data channel that is

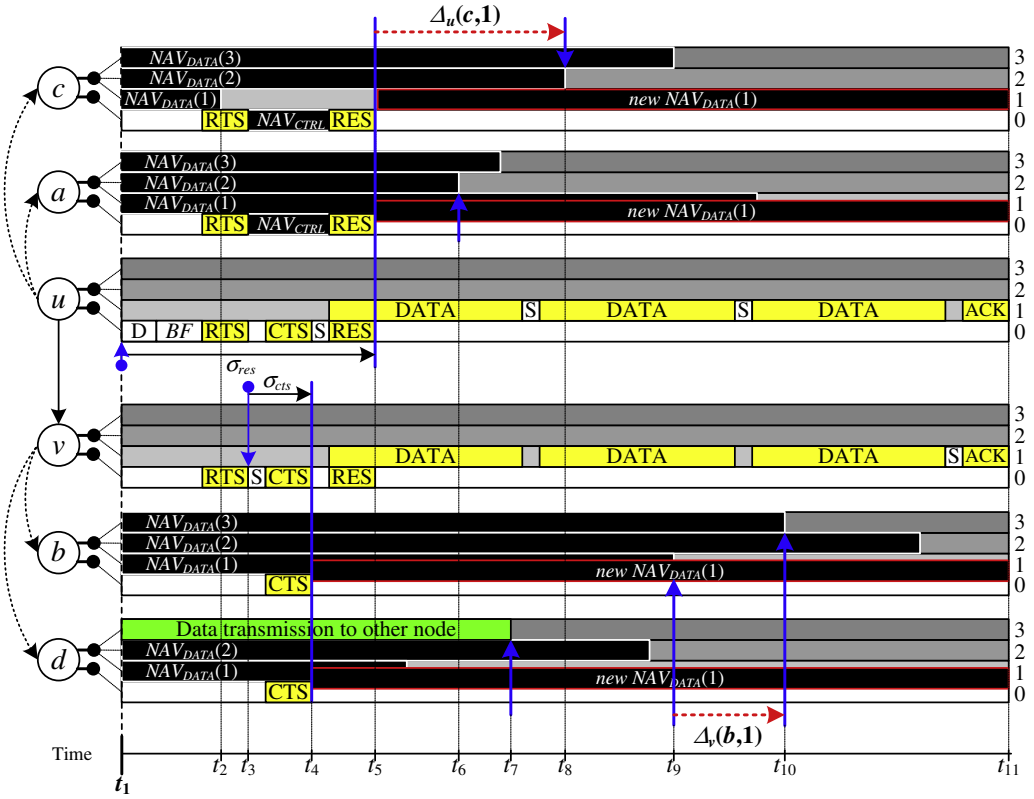


Fig. 9. Increments to starting times of possible transmissions at nearby nodes of *u* and *v* if selecting *ch*1.

not only free to the sender and receiver themselves, but also adds the least total delay to the starting times of possible transmissions at nearby nodes.

Now, we formally describe the *ECS* below. Consider a sender u and a receiver v. As the control process is initiated at *ctrl_ini_time*(u, v), node u firstly identifies a list of data channels that will be released to itself at the expected starting time of the transmission to v as follows.

$$FCL(u) = \{h \epsilon \mathscr{H} \mid ch\_rel\_time(u,h) \leqslant ctrl\_ini\_time(u,v) + pre\_ctrl + BF\}.$$

Note that by definition of the *CIP*, $FCL(u)$ must be non-empty at *ctrl_ini_time*(u, v). In addition, let N_u denote the set of nodes in u's interference range. For each $w \in N_u - \{v\}$, node u calculates the time that at least one data channel will be released to w as

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

The $CR_u(w)$ is called the *critical channel release time* of w, and the first data channel released to w is called the *critical channel* of w. Combining with the interface release time of w in IRV_u, we define

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

It is called the *node release time* of w. Clearly, node w cannot start any transmission before $NR_u(w)$.

Next, node u evaluates the cost for transmitting on each $h \in FCL(u)$. For each $w \in N_u - \{v\}$ and $h \in FCL(u)$, if nodes decided to transmit on channel h for a period of NAV_{DATA}, the $CR_u(w)$ could be enlarged (at least equally), and the new critical release time of w can be formulated as

$$CR_u^+(w|h) = \min\left\{ \begin{array}{l} \max\left\{ \begin{array}{l} CRT_u(w,h), \\ T_{curr} + (pre\_ctrl + BF) + NAV_{DATA} \end{array} \right\}, \\ \min\left\{CRT_u(w,h') | h' \epsilon \mathscr{H} - \{h\}\right\} \end{array} \right\}.$$

The above equation indicates that the original $CR_u(w)$ could be replaced by the release time of another channel $h' \in \mathscr{H} - \{h\}$, if the release time of the original critical channel of w is enlarged so that it is no longer critical to w. As shown in Fig. 9, at t_1, $CR_u(c) = t_2$, but $CR_u^+(c|1) = t_8$ The gap between t_2 and t_8 is due to the fact that the critical channel of c will be altered from ch_1 to ch_2 if u transmits on ch_1. Similarly, the corresponding node release time of w can be wrote as

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

Using these terms, the increment to the node release time of w resulted from transmitting on channel h can be characterized as

$$\Delta_u(w,h) = \max\{NR_u^+(w|h), T_{curr} + \sigma_{res}\} - \max\{NR_u(w), T_{curr} + \sigma_{res}\},$$

where $\sigma_{res} = pre\_ctrl + BF + T_{RTS} + \tau$, denoting the duration before other nodes can receive the RES from u. Notice that the $\Delta_u(w,h)$ neglects the increment before *ctrl_ini_time*(u, v) + σ_{res}, since the transmission is not influential to w before the RES is received by w. As shown in Fig. 9, although $NR_u(c) = CR_u(c) = t_2$ and $NR_u^+(c|1) = CR_u^+(c|1) = t_8$, since the RES will arrive at t_5, the $\Delta_u(c,1)$ is $(t_8 - t_5)$ instead of $(t_8 - t_2)$. Accordingly, if node u transmits on channel h, the total increment to the nodes release time of all neighboring nodes can be defined by

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

The $\Delta_u(h)$ will be sent to v along with a RTS frame for any $h \in FCL(u)$.

When node v received the RTS, it performs the same evaluation for each $h \in FCL(v) \cap FCL(u)$ and $w \in N_v - \{u\} - N_u$, where

$$FCL(v) = \{h \epsilon \mathscr{H} | ch\_rel\_time(v,h) \leqslant T_{curr} + 2SIFS + T_{CTS} + \tau\}.$$

That is, node v calculates

$$CR_v^+(w|h) = \min\left\{ \begin{array}{l} \max\left\{ \begin{array}{l} CRT_v(w,h), \\ T_{curr} + (\sigma_{cts} + SIFS) + (NAV_{DATA} - \tau) \end{array} \right\}, \\ \min\left\{CRT_v(w,h') | h' \epsilon H - \{h\}\right\} \end{array} \right\},$$

$$\Delta_v(w,h) = \max\{NR_v^+(w|h), T_{curr} + \sigma_{cts}\} - \max\{NR_v(w), T_{curr} + \sigma_{cts}\},$$

and

$$\Delta_v(h) = \sum_{w \epsilon N_v - \{u\} - N_u} \Delta_v(w,h),$$

where $\sigma_{cts} = SIFS + T_{CTS} + \tau$, denoting the duration before other nodes can receive the CTS from u.

Then, combining the transmission costs from the two sides, the total cost for communicating between u and v on a channel h can be defined as

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

The $\Delta_{u,v}(h)$ is the total increment to the node release time of all neighbors of u and/or v. In Fig. 9, $\Delta_{u,v}(1) = \Delta_u(1) + \Delta_v(1) = \Delta_u(a, 1) + \Delta_u(c, 1) + \Delta_v(b, 1) + \Delta_v(d, 1) = (t_5 - t_5) + (t_8 - t_5) + (t_8 - t_{10}) + (t_7 - t_7) = (t_8 - t_5) + (t_8 - t_{10})$. $= \Delta_u(a, 1) + \Delta_u(c, 1) + \Delta_v(b, 1) + \Delta_v(d, 1) = (t_5 - t_5) + (t_8 - t_5) + (t_8 - t_{10}) + (t_7 - t_7) = (t_8 - t_5) + (t_8 - t_{10})$. Similarly, we can find the $\Delta_{u,v}(2)$ and $\Delta_{u,v}(3)$. Finally, a channel $h \in FCL(v) \cap FCL(u)$ that has the least $\Delta_{u,v}(h)$ will be chosen as the communication channel, i.e. the h^*. The selected h^* will be sent back to u using the CTS. By definition, the h^* will be released to both u and v and has the least total increment to the node release times (starting times of possible transmissions) of potentially interfered nodes.

5. Simulations

In this section, we conduct simulations using the ns2 simulator. In order to evaluate how much performance gain each of the three components obtains, we have implemented the following versions for the *RTBM*:

- *RTBM* (*CIP*) consists of *CIP* only, has no flow control (i.e. $K_{u,v} = 1$), and follows the random channel selection strategy in [3].
- *RTBM* (*CIP* + *DDC*) is akin to the first version except that *DDC* is applied.
- *RTBM* (*CIP* + *DDC* + *ECS*) employs all designed components.

Note that *DDC* cannot be applied without *CIP*, since $K_{u,v}$ depends on the link release and control initiation times updated from *CIP*. Besides, if $K_{u,v}$ was always fixed on 1, the discrepancy between different data channels h's would be insignificant in terms of their communication costs $\Delta_{u,v}(h)$'s. Hence, we do not individually test *DDC* and/or *ECS* for the *RTBM*.

In addition, due to transmission failures or delays in updating control messages, the values stored in CRT_u and IRV_u could be stale or incomplete. The inaccuracy may lead to a series of invalid decisions in the *RTBM* (as shown in Fig. 6(c)). For this reason, we provide an ideal version, denoted as *RTBM**(*CIP* + *DDC* + *ECS*), where each node can directly access the information in its neighbors, to show how the performance degrades due to inaccurate information.

On the other hand, we compare the *RTBM* with the representative *DCC*-based protocol (*DCA*) in [3] and the parallel rendezvous protocol (McMAC) in [18]. The primary goal of McMAC is also to overcome the control bottleneck problem. The difference is that McMAC disperses control traffic across all available channels by allowing a sender switch to the hopping sequence of the intended receiver with a probability p. Here, we set p as the default setting in [18]. For all of our evaluations, the IEEE 802.11 DCF serves as the baseline protocol.

For each network under test, we generate 100 static nodes uniformly distributed on a 1500 m $\times$ 1500 m region. Each node is equipped with two interfaces, each with a default transmission range of 250 m. The interference range is assumed to be twice the transmission range. On the other hand, we provide 12 orthogonal channels. One of them is used for control purposes and the others are used for data transmission. The default channel bit rate is 11 Mbps.

On top of each generated network, we consider the throughput under two circumstances. In *single-hop communication*, we establish UDP flows with a variable bit rate over 200 random node pairs (i.e. each node communicates with four 1-hop nodes on average). Each communication pair is confined within the transmission range of each other. This case is intended to evaluate the link-layer performance. In *multi-hop communication*, we establish UDP flows with a constant bit rate over 20 randomly chosen node pairs. All sources and destinations are distinct nodes. Each packet is forwarded along with a shortest path in terms of the hop count. In both cases, the (average) packet arrival rate is 1 Mbps and the packet length is 1024 bytes. The throughput is defined as the total size of data packets successfully received by all destinations divided by the simulation time (100 s). Besides, each node u maintains a 50-packet queue for any 1-hop node v, i.e. $Q_v \leqslant 50$.

Table 4 lists the frame sizes of each compared protocol under $H = 11$. In addition to those in IEEE 802.11 DCF specifications (i.e. 20 bytes for RTS, 14 bytes for CTS, and 14 bytes for ACK), the DCA needs a 2-byte bitmap to carry the free channel list in RTS, and 2 bytes to indicate the selected channel and a parameter in CTS (see [3]). Thus, the frame sizes of RTS and CTS in DCA are 22 bytes (i.e. 22 = 20 + 2) and 16 bytes (i.e. 16 = 14 + 2), respectively. About the RTBM, its RTS additionally needs

Table 4
Frame sizes under $H = 11$ and system values of DSSS PHY.

	802.11 DCF/McMAC	*DCA*	*RTBM*
RTS length	20 bytes	22 bytes	44 bytes
CTS length	14 bytes	16 bytes	39 bytes
RES length	–	16 bytes	39 bytes
ACK length	14 bytes		
Propagation delay	5 μs		
Backoff slot time	20 μs		
SIFS	10 μs		
DIFS	50 μs		

2 bytes to encode the communication $\Delta_u(h)$ for each data channel h. Thus the size of RTS under $H = 11$ is 44 bytes (i.e. $44 = 20 + 2 + 2 \times 11$). Moreover, the CTS and RES needs 2 bytes for each duration of *if_duration*(v) and *ch_duration*(u, h), and 1 byte for the selected channel h^* (see Section 3.2). Thus, the sizes of the later two frames under $H = 11$ are 39 bytes (i.e. $39 = 14 + 2 + 2 \times 11 + 1$). Other system values specified for the DSSS PHY are also listed in Table 4. Although the *RTBM* needs larger frame sizes, as shown later, the benefit from our components can wholly compensate for such a drawback. The above contexts are the default settings. We also vary some parameters one at a time to assess the performance under different scenarios. Any result point of the following figures is averaged from 20 networks.

Fig. 10 reports the throughput versa the number of data channels (H). In comparison with the single-channel 802.11 DCF, the *DCA* can use multiple data channels to increase the throughput. When $H = 2$, it achieves a 28.98% gain in single-hop case and a 44.72% gain in multi-hop case. However, the improvement becomes very limited as more channels are provided. With 11 channels, the *DCA* obtains merely a 40.08% and 79.59% gain in the two cases, respectively. The results indicate that only a small portion of data channels were utilized due to the bottleneck in control channel.

The bottleneck problem can be mitigated by *CIP*. As shown in Fig. 10, the *RTBM*(*CIP*) has 4–42% throughput increments over the *DCA*. The reason is that *CIP* can prevent nodes from sending redundant RTSs and thus avoid unnecessary blocking from NAV_{CTRL}. Notice that *CIP* is particularly useful for small H, because a coordination process may fail when two nodes have no free data channels in common, and there are relatively fewer common channels when H is small. Nonetheless, *CIP* cannot entirely break through the limitation of the control channel, since a coordination process is still required before sending each data packet.

The *DDC* can substantially resolve this problem. As shown in Fig. 10, *RTBM*(*CIP+DDC*) has improved the baseline throughput by over 2.26 times in single-hop case and 3.49 times in multi-hop case, with just one additional data channel, i.e. $H = 2$. Moreover, for $H = 4$, it can further gain 2.86 times and 5.17 times performance in the two cases. However, since the traffic load is not quite heavy in default, the throughput reaches the limit when $H \geqslant 6$. As shown in Fig. 11, by pressuring nodes

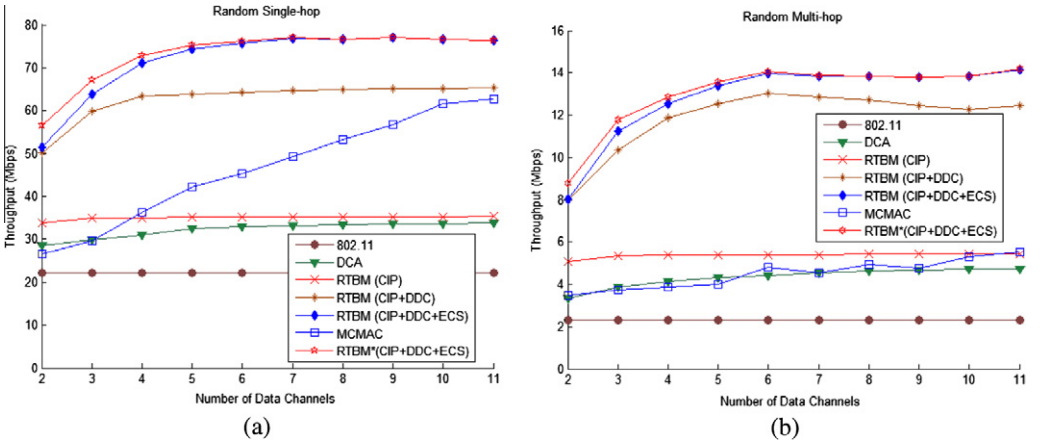


Fig. 10. Variation in the number of data channels.

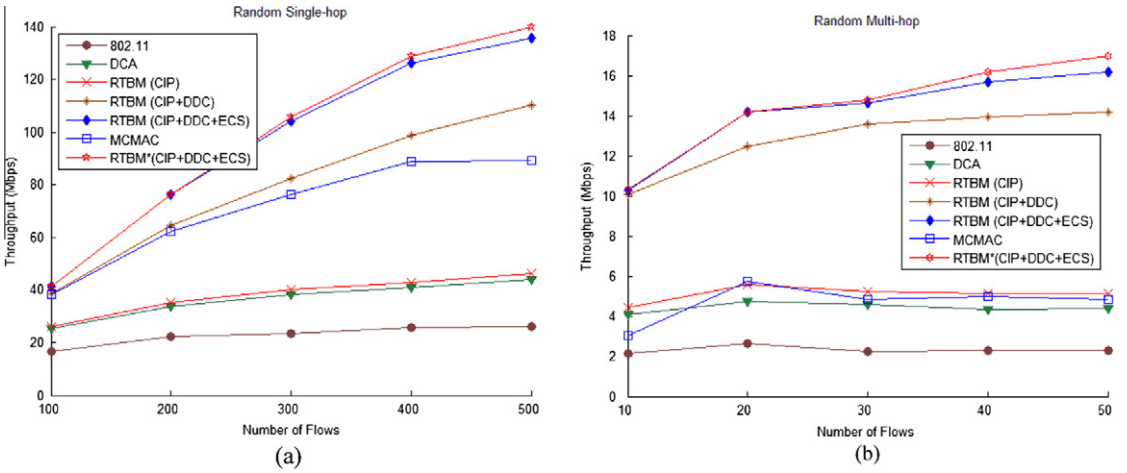


Fig. 11. Variation in the number of flows ($H = 11$).

with more data flows, the throughput is almost linear to the number of flows under $H = 11$. Contrarily, other protocols without *DDC* even degrade the throughput as the number of flows exceeds a certain limit.

The bottleneck problem is also resolved by McMAC. As shown in Fig. 10(a), the throughput of McMAC increases significantly when H raises. However, when H is small, McMAC does not perform well (even worse than the original *DCC* when $H = 2$). One reason is that the congestion in each channel is still heavy for a small H. In contrast, *DDC* can dynamically adjust the number of data packets being sent to balance the congestion in both control and data channels no matter how many channels are provided. Another reason is that the *DCC*-based approach can separate control and data traffic into different channels so that control and data signals are not interfering to each other. Note that although *RTBM* needs one additional channel for control purposes, we can see that the throughput of *RTBM*(*CIP* + *DDC*) with h channels is higher than that of McMAC with $h + 1$ channels in all cases. Moreover, McMAC has no significant improvement in a multi-hop case, because a node may frequently depart to different hopping sequences to forward messages, which however are always not visible to other nodes in the interference range, incurring a serious collision problem. In contrast, *RTBM* allows each node to acquire the status of other nodes and channels at any time.

The *ECS* can further enhance the throughput. As shown in Fig. 10, *RTBM*(*CIP* + *DDC* + *ECS*) shows clear gains over *RTBM*(*CIP* + *DDC*), especially when more data channels are provided. One observation explains this tendency: When H is large, each sender u and receiver v have more choices to find a data channel h which has a lower cost $\Delta_{u,v}(h)$. Particularly, under 11 data channels and 500 flows, *ECS* additionally contributes 26.37% of throughputs in single-hop case (15.41% in multi-hop case).

In Fig. 12, we vary the channel bit rate (R) to show how raw channel capacity relates to the throughput. As R is raised from 0.5 Mbps to 11 Mbps, we can see that 802.11 DCF gains approximately 9–10 times throughput, but there are relatively lower gains (no more than 4 times) for *DCA* and *RTMB*(*CIP*). The lower growth rates are caused by the following reason: with a higher channel bit rate, the data transmission is faster, which implies that a node may initiate more control processes within a shorter period of time. As a result, contention in the control channel becomes more intensive. *RTBM* can get rid of such limitations by using *DDC*. As shown in Fig. 12, *RTMB*(*CIP* + *DDC* + *ECS*) obtains 8.22 times throughput in the single-hop case and 9.39 times throughput in the multi-hop case as long as R is raised from 0.5 Mbps to 11 Mbps, which are very close to those achieved by 802.11 DCF, where no control bottleneck exists. The reason for this is that *DDC* can dynamically send multiple data frames to prolong the data transmission whenever the control channel is congested, i.e. data interface's idle time rises. Besides, *ECS* can provide more enhancements when R is large, because the greater the channel bit rate is, the more free data channels exist, providing more choices for finding the lower communication cost.

Fig. 13 demonstrates the impact of increasing transmission (interference) range We can see that the throughputs of 802.11 DCF, *DCA*, and *RTBM*(*CIP*) degrade drastically as the range increases. In contrast, *RTBM*(*CIP* + *DDC*) has only suffered a slight decrement in its throughput, since *DDC* can dynamically adapt to any increasing interference in the control channel. Moreover, *ECS* can help nodes to achieve better channel reusability. Hence, *RTBM*(*CIP* + *DDC* + *ECS*) is almost not affected by this factor. Surprisingly, the throughput can be even better when the range is more than 250 m. This phenomenon possibly stems from the fact that a larger transmission (interference) range can avoid the presence of hidden terminal nodes in the control channel.

Now let us compare with the ideal version. We can see that the throughput of *RTBM*(*CIP* + *DDC* + *ECS*) is very close to that of *RTBM**(*CIP* + *DDC* + *ECS*) especially when H is large (see Fig. 10), because the change of a channel release time can be less frequent if more other channels are offered. Besides, the throughput degradation is very small even under a heavy loading (see Fig. 11). Therefore, the inaccuracy of information has a little impact to our protocol.

Fig. 14 shows the normalized control overhead for each protocol (the total size of control messages being sent divided by that of the 802.11 DCF) over different number of flows. Consider the single-hop case: By comparing Fig. 11(a), we can see

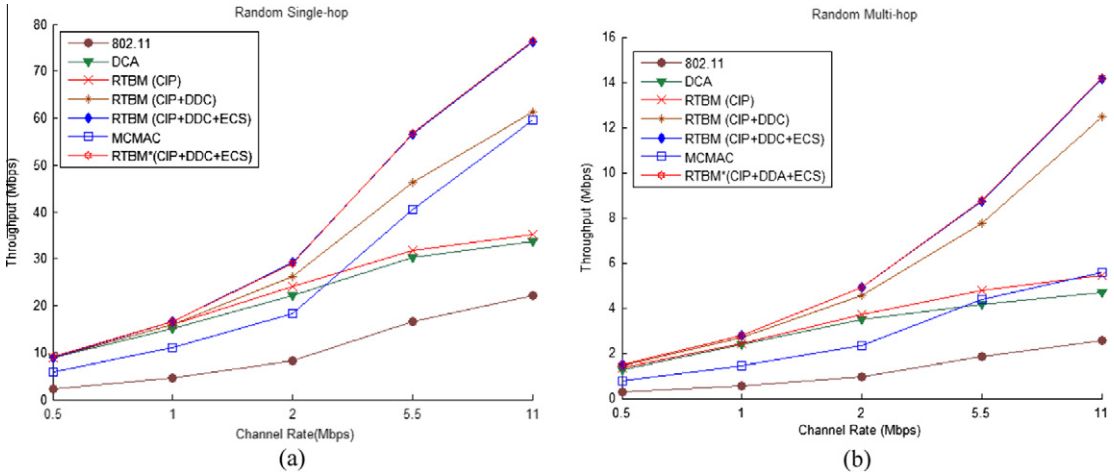


Fig. 12. Variation in channel bit rate ($H = 11$).

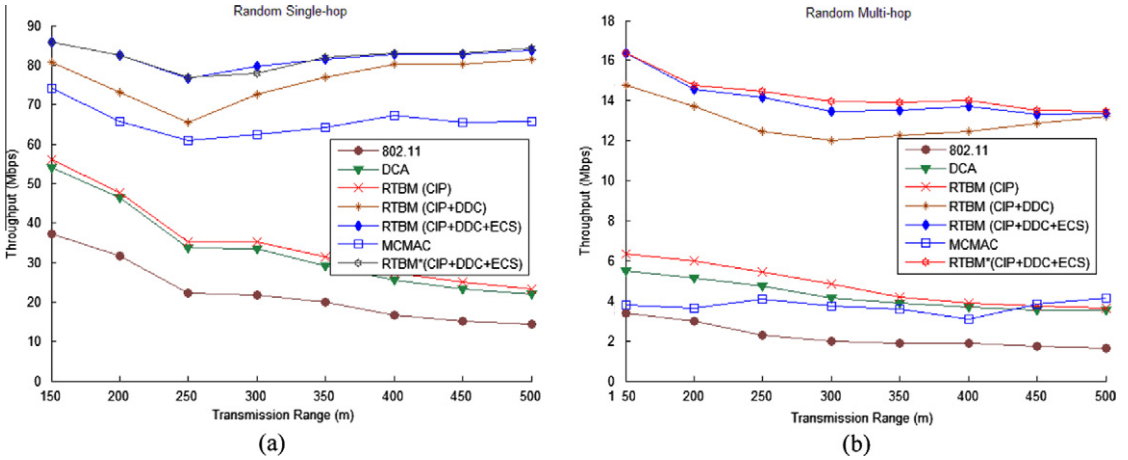


Fig. 13. Variation in transmission range ($H = 11$).

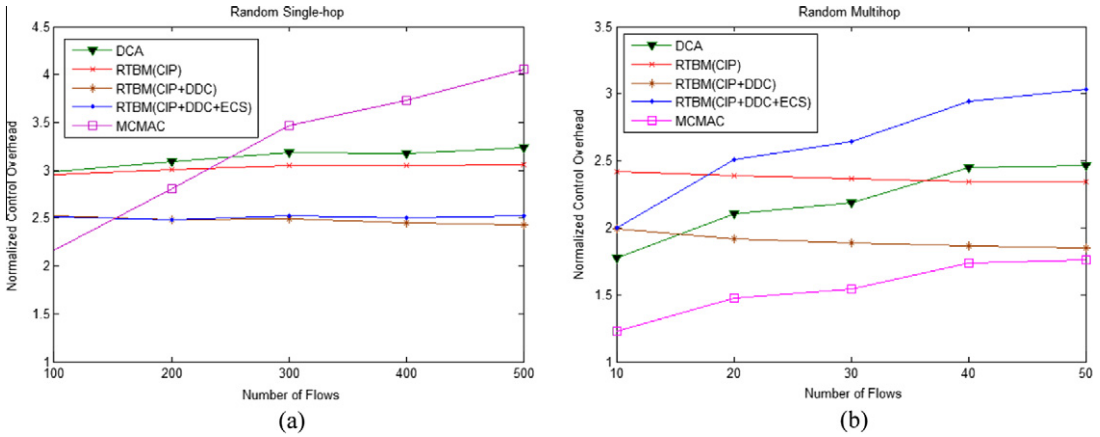


Fig. 14. Normalized control overhead ($H = 11$).

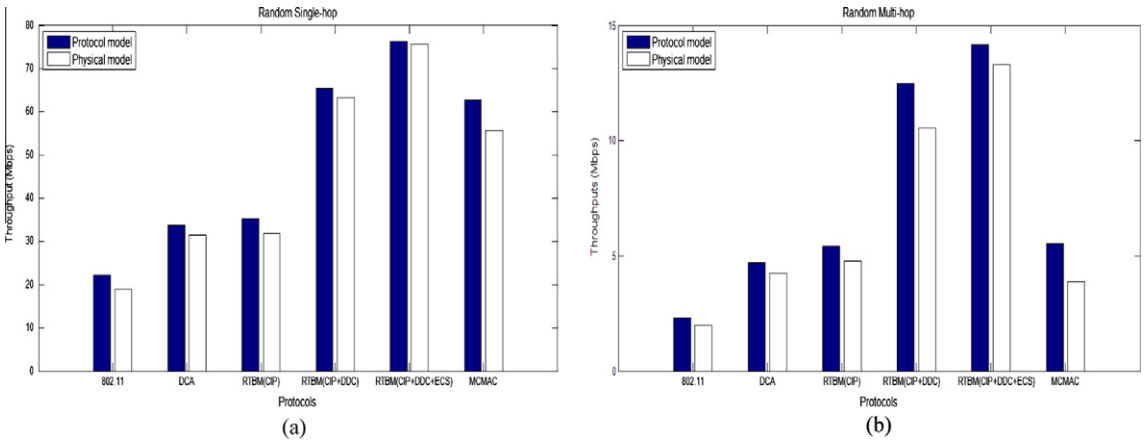


Fig. 15. Throughput under protocol vs. physical interference model ($H = 11$).

that the control overhead of McMAC goes up in accompany with more data throughput, because McMAC can spread control loading over different channels. By contrast, the throughput of *DCA* is saturated by a small number of flows, since the control overhead is congested in a single channel. Compared with *DCA*, although *RTBM* needs larger control frames sizes, it incurs

lower control overhead and achieves higher throughput in the three versions, because *CIP* can offer a better utilization of the control channel and *DDC* can increase the number of data packets being sent followed by each control process. Consider the multi-hop case: McMAC has the lowest control overhead and lower throughput, while *RTBM*(*CIP + DDC + ECS*) has the highest control overhead and throughput. The results indicate that the impact from data channels could be more serious in a multi-hop case.

Fig. 15 shows the throughout under protocol interference model versa physical interference model based on SINR. We can see that our protocol has only a slight degradation under the physical model especially when *ECS* is applied, while McMAC has a larger degradation in throughput. The reason is that *ECS* is designed to increase the reusability of data channels so that the interference problem is less serious under both of the two models. Note that the throughput degradation becomes more significant in multi-hop case for each protocol because in multi-hop case the physical model not only incurs more serious interference among different communication pairs (i.e. the inter-flow interference) but also worsens the interference between hops of the same flow (i.e. the inter-hop interference). Besides, for a transmission, there are more hidden terminal nodes in both control and data channels when multi-hop flows exist.

6. Conclusion

In this paper, we have proposed a release-time-based MMAC protocol to overcome the control channel bottleneck problem and the dynamic channel selection problem for the *DCC* approach. Three components have been investigated and incorporated into this protocol. The control initiation-time predation can reduce the redundant control overhead by properly predicting the control point to increase the chance for a successful coordination. The dynamic data-flow control can dynamically adjust the number of data packets to be sent according to the real-time condition of both the control and data interfaces. The enhanced channel selection can gain better channel reusability by selecting the channel that has the least influence to the transmission starting times of 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 rates of data channels are large. Both McMAC and *RTBM* with *DCC* can overcome the control channel problem. However, *RTBM* can provide higher throughput due to channel selection strategy. Besides, *RTBM* is a more suitable to a complex environment where multi-hop transmission and frequent link-layer broadcasting exist.

For future research, it is worthy to evaluate the performance of mobile nodes. In this circumstance, the quality of a channel selection strategy could be more influential to the final results. In addition, the bottleneck problem can be further alleviated if a candidate control channel is used, but the design is expected to be more complicated.

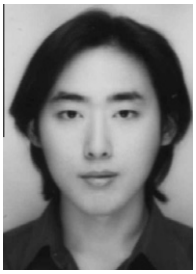
Acknowledgements

The authors would like to thank Yu-Hsiang Cheng for this helpful assistance in conducting the experimental results.

Reference

[1] I.-F. Akyildiz, X. Wang, Welin Wang, Wireless mesh networks: a survey, Computer Networks 47 (2005) 445–487.
[2] J. Mo, H.-S. So, J. Walrand, Comparison of multichannel MAC protocols, IEEE Transactions on Mobile Computing 7 (1) (2008) 50–65.
[3] S.-L. Wu, Y. Lin, Y.-C. Tseng, 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, 2000, pp. 232–237.
[4] S.-L. Wu, Y. Lin, Y.-C. Tseng, J.-P. Sheu, A multi-channel MAC protocol with power control for multi-hop mobile ad hoc networks, Computer Journal 45 (1) (2002) 101–110.
[5] P.-J. Wu, C.-N. Lee, On-demand connection-oriented multi-channel MAC protocol for ad-hoc network, in: Proc. of 3rd IEEE Conf. Society Conf. on Sensor and Ad Hoc Networks, 2006, pp. 621–625.
[6] W.-C. Hung, K.-L.-E. Law, A. Leon-Garcia, A dynamic multi-channel mac for ad-hoc LAN, in: Proc. of 21th Biennial Symp. On Communications, 2002, pp. 31–35.
[7] H. Koubaa, Fairness-enhance multiple control channels MAC for ad hoc networks, in: Proc. of 62th IEEE Vehicular Technology Conference, 2005, pp. 1504–1508.
[8] M. Benveniste, Z. Tao, Performance evaluation of a medium access control protocol for IEEE 802.11s mesh networks, in: Proc. of IEEE Sarnoff Symp., 2006, pp. 1–5.
[9] J. Zhang. Y. Wang, 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, 2003, pp. 1341–1345.
[10] N. Jain, S.-R. Das, A. Nasipuri, A multichannel CSMA MAC protocol with receiver-based channel selection for multihop wireless networks, in: Proc. of 10th Int'l Conf. on Computer Communications and Networks, 2001, pp. 432–439.
[11] A. Raniwala, K. Gopalan, T.C. Chiueh, Centralized channel assignment and routing algorithm for multi-channel wireless mesh networks, ACM Mobile Computing and Communications Review 8 (2) (2004) 50–55.
[12] A.-K. Das, H.-M.-K Alazemi, R. Vijayakumar, S. Roy, Optimization models for fixed channel assignment in wireless mesh networks with multiple radios, in: Proc. of 2nd Annual IEEE Communications Society Conf. on Sensor and Ad Hoc Networks, 2005, pp. 463–474.
[13] J. Crichigno, M.-Y. Wu, W. Shu, Protocols and architectures for channel assignment in wireless mesh networks, Ad Hoc Networks 6 (7) (2007) 1051–1077.
[14] J. So, N.-H. Vaidya, Multi-channel MAC for ad hoc networks: handling multi-channel hidden terminals using a single transceiver, in: Prof. of 5th ACM Int'l Symp. on Mobile Ad Hoc Networking and Computing, 2004, pp. 222–233.
[15] J. Chen, S.-T. Sheu, C.-A. Yang, A new multichannel access protocol for IEEE 802.11 ad hoc wireless LANs, in: Proc. of 14th IEEE Int'l Symp. on Personal, Indoor and Mobile Radio Communications, 2003, pp. 2291–2296.
[16] Z.-J. Haas, J. Deng, Dual busy tone multiple access (DBTMA)-a multiple access control scheme for ad hoc networks, IEEE Transactions on Communications 5 (6) (2000) 975.
[17] S.-L. Wu, Y.-C. Tseng, J.-P. Sheu, Intelligent medium access control for mobile ad hoc networks with busy tones and power control, IEEE Journal on Selected Areas in Communications 18 (9) (2000) 1647–1657.
[18] A. Tzamaloukas, J.J. Carcia-Luna-Aceves, Channel-hopping multiple access, in: Proc. of IEEE Int'l Conf. Comm. (ICC'00), June, 2000.

[19] Y. Li, H. Wu, N.-F. Tzeng, D. Perkins, M. Bayoumi, MAC-SCC: a medium access control with separate control for reconfigurable multi-hop wireless networks, IEEE Transactions on Wireless Communications 5 (7) (2006) 1805–1817.
[20] P. Kyasanur, J. Padhye, P. Bahl, On the efficacy of separating control and data into different frequency bands, in: Proc. of Broadband Networks, 2005, pp. 646–655.
[21] P. Bahl, R. Chandra, J. Dunagan, SSCH: slotted seeded channel hopping for capacity improvement in IEEE 802.11 ad hoc wireless networks, in: Proc. of ACM MobiCom, September 2004.
[22] H.-S. So, J. Walrand, J. Mo, McMAC: a parallel rendezvous multi-channel mac protocol, in: Proc. of IEEE Wireless Communications and Networking Conference, 2007, pp. 334–339.
[23] P. Kyasanur, J. So, C. Chereddi, N.H. Vaidya, Multichannel mesh network: challenges and protocols, IEEE Wireless Communications 10 (2) (2006) 30–36.
[24] A.A.K. Jeng, R.H. Jan, Role and channel assignment for wireless mesh networks using hybrid approach, Computer Networks 53 (12) (2009) 2225–2240.



Andy An-Kai Jeng received the B.S. degree in statistics from Tamkang University, Taiwan, in 2001, the M.S. degree in management information systems from National Chi Nan University, Taiwan, in 2003, and Ph.D degree in the computer science from National Chiao Tung University, Taiwan, in 2007, where is currently a post-doctoral researcher. His research interests include wireless networks, distributed algorithm design and analysis, scheduling theory, and operations research.

Rong-Hong Jan received the B.S. and M.S. degrees in Industrial Engineering, and the Ph.D. degree in Computer Science from National Tsing Hua University, Taiwan, in 1979, 1983, and 1987, respectively. He joined the Department of Computer and Information Science, National Chiao Tung University, in 1987, where he is currently a Professor. During 1991–1992, he was a Visiting Associate Professor in the Department of Computer Science, University of Maryland, College Park, MD. His research interests include wireless networks, mobile computing, distributed systems, network reliability, and operations research.

Chi-Yu Li received the B.S. degree in Computer and Information Science from National Chiao Tung University, Taiwan, in 2004 and M.S. degree in Computer and Information Science from National Chiao Tung University, Taiwan, in 2006. His research interests include WMN (Wireless Mesh Network) and wireless MAC protocols.

Chien Chen received his B.S degree in Computer Engineering from National Chiao Tung University in 1982 and the M.S. and Ph.D. degrees in Computer Engineering from University of Southern California and Stevens Institute of Technologies in 1990 and 1996. Dr Chen hold a Chief Architect and Director of Switch Architecture position in Terapower Inc., which is a terabit switching fabric SoC startup in San Jose, before joining National Chiao Tung University as an Assistant Professor in August 2002. Prior to joining Terapower Inc., he is a key member in Coree Network, responsible for a next-generation IP/MPLS switch architecture design. He joined Lucent Technologies, Bell Labs, NJ, in 1996 as a Member of Technical Staff, where he led the research in the area of ATM/IP switch fabric design, traffic management, and traffic engineering requirements. His current research interests include wireless ad-hoc and sensor networks, switch performance modeling, and DWDM optical networks.