Infrastructure-Based MAC in Wireless Mobile Ad-hoc Networks

by

Tiantong You

A thesis submitted to the

Department of Computing and Information Science

in conformity with the requirements for

the degree of Master of Science

Queen's University

Kingston, Ontario Canada

January 2002

0-612-65658-6

ABSTRACT

The IEEE 802.11 standard is the most popular Media Access Control (MAC) protocol for infrastructure-based wireless local area networks. However, in an ad-hoc environment, the Point Coordination Function (PCF), defined in the standard, cannot be readily used. This is due to the fact that there is no central authority to act as a Point Coordinator (PC). Peer-to-peer ad-hoc mode in the IEEE 802.11 standard only implements the Distributed Coordination Function (DCF). In this thesis, an efficient and on-the-fly infrastructure is created using our proposed Mobile Point Coordinator (MPC) protocol. Based on this protocol, we also develop an efficient MAC protocol, namely MPC-MAC. Our MAC protocol extends the IEEE 802.11 standard for use in multihop wireless ad-hoc networks implementing both the DCF and PCF modes of operation. The goal, and also the challenge, is to achieve QoS delivery and priority access for real-time traffic in ad-hoc wireless environments while maintaining backward compatibility with the IEEE 802.11 standard.

The performance of MPC-MAC is compared to the IEEE 802.11 DCF-based MAC without MPC. Simulation experiments show that in all cases the use of PCF benefits real-time packets by decreasing the average delay and the discard ratio. However, this may come at the expense of increasing the average delay for non-real-time data. The discard ratio for both real-time and non-real-time packets improves with the use of PCF. Therefore, our MPC-MAC outperforms the standard DCF IEEE 802.11 MAC protocol in multi-hop ad-hoc environments.

ACKNOWLEDGEMENTS

I am deeply indebted to my supervisors Dr. Hossam Hassanein and Dr. Hussein Mouftah. Their distinctive supervision and mentorship were a great source of help and guidance, throughout the course of my research.

I would like to thank my wonderful colleague and friend Ahmed Safwat. He gave me a lot of help with his great talent and experience in the wireless networks research area. Also, the help of other colleagues in the Telecommunications Research Lab is greatly appreciated.

I wish to express my gratitude to another special person, David Gagnon. David, this thesis owes so much to your support. I'm proud to know an honest sincere person like you. You made my stay in Kingston such an enjoyable and memorable experience. Thank you for being my best friend and I hope we can be friends forever.

I would also like to thank my wife. Without her love and encouragement, I could not have accomplished this piece of work.

LIST OF ACRONYMS

λ	Packet Creation Rate
ACK	Acknowledgment Packet
AD	Average Delay
CDMA	Code Division Multiple Access
CSMA	Carrier Sense Medium Access
CSMA/CA	Carrier Sense Multiple Access / Collision Avoidance
CTS	Clear to Send
DCF	Distributed Coordination Function
DIFS	DCF Inter Frame Space
DR	Discard Ratio
FDMA	Frequency Division Multiple Access
IFS	Inter Frame Space
L	Packet Length
LAN	Local Area Network
MAC	Medium Access Control
MIB	Management Information Base
MPC	Mobile Point Coordinator
MT	Mobile Terminal

N	Number of Nodes
NAV	Network Allocation Vector
P	MT Average Pause Time
PC	Point Coordinator
PCF	Point Coordination Function
PIFS	PCF Inter Frame Space
R	Radio Transmission Range
r-MT	Registered Mobile Terminal
RTS	Request To Send
S	MT Average Move Speed
SDN	Slot Defer Number (number of slot extension after IFS)
SIFS	Short Inter Frame Space
TDMA	Time Division Multiple Access
VBS	Virtual Base Station
Zone-MT	MT that is registered to an MPC

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

For wireless Local Area Networks (LANs), there are two different ways to configure a network: infrastructure-based and ad-hoc [1]. As Figure 1.1 shows, the infrastructure-based network structure used in wireless LANs is very similar to present day cellular networks. It comprises two levels – a stationary level and a mobile level. The stationary level consists of geographically fixed Base Stations (BSs) that are interconnected through wired or wireless means (which may use an identical radio frequency). The mobile level consists of Mobile Terminals (MTs), which communicate with BSs or each other through wireless links. The geographical area around a BS from which MTs can communicate with the BS within one hop is called a cell. Neighboring cells overlap with each other, thus ensuring the continuity of communication. These BSs are sometimes connected to landlines to widen the LAN's capability by bridging wireless nodes to other wired nodes. They are permanently turned on in fixed locations that have naturally been chosen to coordinate the wireless Local Area Network (LAN). So the BSs are sometimes called Access Points (AP) to the Internet. When the MT turns on, it first tries registering to a certain BS; thus, the BS can track it successfully. The BS normally uses a unique radio frequency (different from neighboring BSs) to communicate with its group members.

The communication between MTs that are registered to different BS's has to go through the wired connection between these BS's.

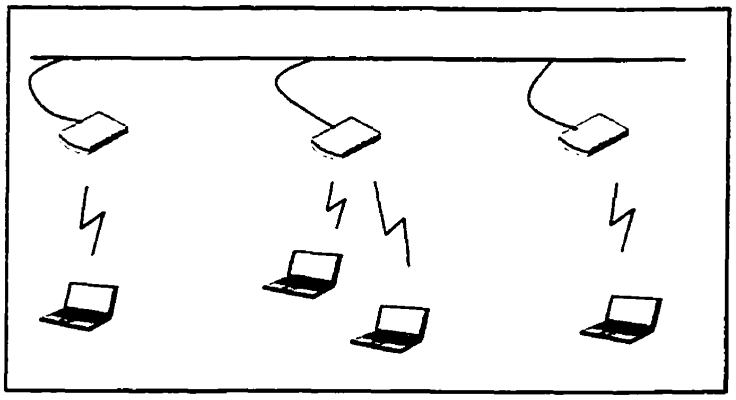


Figure 1.1 Access-point based wireless networks

In the ad-hoc type of wireless network portable devices are brought together to form a network *on the fly*. As shown in Figure 1.2, there is no infrastructure for the network,

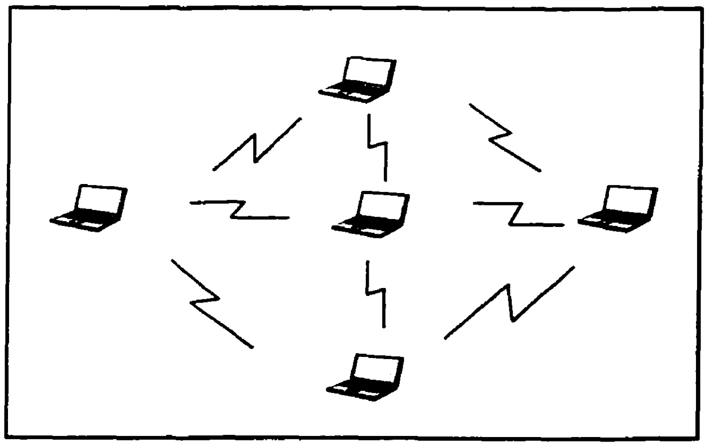


Figure 1.2 Ad-hoc wireless networks

that is, there are no fixed points and usually every node is able to communicate with every other node when all nodes are spread around a relatively small geographic range. As an example, consider a meeting in a conference room where employees bring laptop computers together to communicate and share design or financial information.

However, nodes may spread over a larger geographic range than the communication signal can reach. In this case nodes may have to communicate over multiple hops. The ad-hoc network has no central control, but it has the advantage of being relatively inexpensive, since it does not require the communication infrastructure of the fixed approach. An ad-hoc wireless network also has its value in the military and in emergency situations.

In wire-line networks, the network can have a very high bandwidth by increasing the number of links or link capacity. However, in wireless networks, there is only one medium that is shared by all the nodes that are in the same radio communication range, and the radio frequency bandwidth is limited. As well, packet collisions are unavoidable due to the fact that traffic arrivals are random and there is non-zero propagation time between transmitters and receivers. Therefore, Medium Access Control (MAC) schemes are used to coordinate access to the single channel in the network.

Cellular MAC protocols are mainly classified into two categories; allocation-based and code-based protocols. In allocation-based MAC protocols, each mobile terminal transmits its packets using a dedicated traffic channel or according to an agreed-upon time schedule. However, this results in the network bandwidth not being efficiently utilized whenever a mobile terminal does not use its assigned channel or time slot [2]. Frequency Division Multiple Access (FDMA) and Time Division Multiple Access (TDMA) [3] are the two basic access methods used by this class of protocols. In FDMA, users share the available spectrum in the frequency domain, and a user is allocated part of the frequency band called the traffic channel. Different users are assigned different traffic (frequency) channels on a demand basis. In cellular systems using TDMA, the available spectrum is used as a whole during a period of time, which is in turn divided into a number of time slots. An individual user is assigned a time slot that permits access to the frequency channel for the duration of the time slot.

In code-based protocols, all users use the same carrier frequency and may transmit simultaneously. The access method utilized by such protocols is known as Code Division Multiple Access (CDMA) [3], which uses mathematical codes to transmit and distinguish between multiple wireless conversations. Since the conversations are distinguished by digital codes, many users can share the same bandwidth simultaneously.

In the ad-hoc environment, there is no centralized authority to assign specific radio frequencies, time slots or codes to different mobile nodes that are totally distributed. Mobile terminals have to contend for the medium access by themselves. Carrier Sense Medium Access (CSMA) [4] is the main mechanism to implement medium access.

Consequently, transmissions of packets from distinct mobile terminals are more prone to overlap, resulting in packet losses. Retransmissions are required and a noticeable delay appears. To make the existing infrastructure-based wireless protocol usable in the ad-hoc environment and also to make the ad-hoc wireless networks more manageable for routing and MAC, developing an infrastructure for the infrastructure-less wireless mobile ad-hoc networks is of utmost importance. Therefore, in this thesis, a wireless mobile infrastructure is developed for ad-hoc networks. We propose an infrastructure-creation protocol for wireless mobile ad-hoc networks based on the concept of a Mobile Point Coordinator (MPC) protocol. We then develop an IEEE 802.11 backward compatible MAC protocol in which the MPC will replace the Base Station in cellular wireless networks.

This thesis is organized as follows. Chapter 2 discusses related work in MPC creation and MAC protocols for ad-hoc networks. In Chapter 3, the MPC infrastructure-creation protocol and MPC-MAC are described. In chapter 4, a simulation model is developed to evaluate the proposed MAC protocol. Finally, Chapter 5 presents conclusions and future work.

CHAPTER 2

RELATED WORK

In this Chapter, we first go over the recently proposed schemes for clustering in wireless ad-hoc networks. We then discuss some existing schemes used to implement MAC in wireless networks and some potential problems that may arise if these schemes are directly used in the ad-hoc environment.

2.1 Clustering in wireless ad-hoc networks

A number of clustering techniques have been proposed for wireless ad-hoc networks [5-19]. A mobile infrastructure is developed in [5,6] to replace the wired backbone infrastructure in conventional cellular networks. The algorithm divides the wireless mobile ad-hoc network into a set of clusters with no cluster-head; each cluster contains a number of MTs that are limited to being two hops away. In this scheme a cluster center node, which has the highest connectivity in the cluster, is chosen to ensure that distance between any two members in the cluster is limited to two hops. The main drawback of this protocol is the stability of the cluster in a high-mobility network. As the network changes slightly, the nodes' degree of connectivity is much more likely to change as well.

Contrasted with the scheme above, several heuristics have been proposed to choose cluster-heads [7-19], but they do not produce an optimal solution with respect to battery usage, load balancing and MAC functionality in an ad-hoc network. The criteria for choosing the cluster-head used by these heuristics include (i) Highest-Degree heuristic [7-9] (ii) Lowest-ID heuristic [7,13-15] and (iii) Node-Weight heuristic [16-19].

The Highest-Degree Heuristic computes the degree of a node based on the distance between that node and others. In the assumed graph model of the network, the mobile terminals are represented as nodes and there exists an "edge" between two nodes if they can communicate with each other directly. The node with the maximum degree is chosen to be a cluster-head. The neighbors of this cluster-head (nodes that connect to it by edge) become members of that cluster and lose the right of election process. This heuristic is also known as the highest-connectivity algorithm. Simulation experiments demonstrate that the system has a low rate of cluster-head changes but the throughput of the system is low. Typically, each cluster is assigned some resources, which are shared among the members of that cluster on a round-robin basis [7-9]. As the number of nodes in a cluster is increased, a gradual degradation in the system performance is observed by dropping the throughput of each user. This is an inherent drawback of the Highest-Degree heuristic since the number of nodes in a cluster is not bounded.

The Lowest-ID Heuristic, proposed in [7], assigns a unique ID to each node first, and the node with the minimum ID is chosen as a cluster-head. In some cases, the cluster-head can delegate its duties to the next node with the minimum ID in its cluster. Neighboring clusters are connected by "gateway" nodes that lie within the transmission range of two or more clusters. Simulation experiments show that Lowest-ID Heuristic outperforms the Highest-Degree heuristic, in terms of the throughput due to the less frequent cluster-head update in the ad-hoc mobile environment. The disadvantage of this scheme also comes from this stable cluster-head. The nodes with smaller IDs tend to experience battery drainage because of the bias toward using them as cluster-heads. Moreover, this scheme does not attempt to balance the load uniformly across all the nodes.

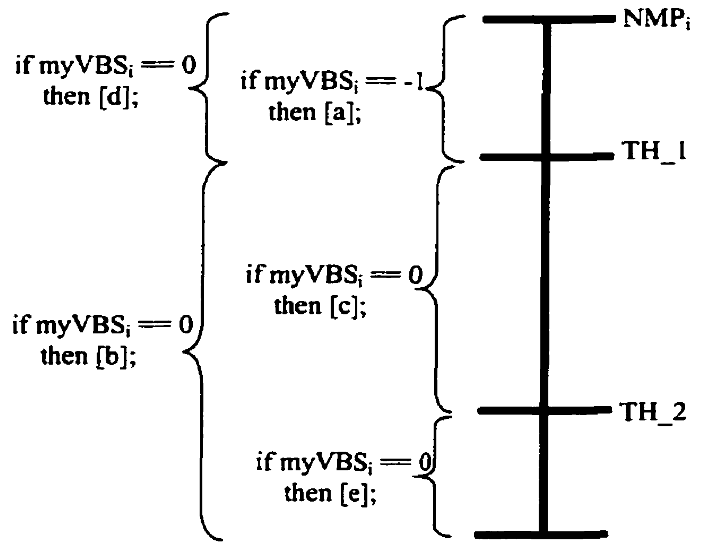
In deviation from the Lowest-ID mentioned above, (that assigns dynamic to the node ID first), A. Safwat and H. Hassanein have proposed Virtual Base Stations (VBS) [13-15] using just the unique IP that every node owns natively as the selection criteria. In the VBS selection scheme, some of the MTs, based on an agreed-upon policy, become in charge of all the MTs in their neighborhood, or a subset of them by electing one as VBS. If a VBS moves or stops acknowledging its presence via its so-called “hello” packets, for a period of time, a new one is elected. Every MT (zone_MT/VBS) will maintain a sequence number that reflects the changes that occur to that MT, and a my_VBS variable, which is used to store the VBS in charge of that MT. Hello messages sent by VBSs contain their current knowledge of the ad-hoc network. A VBS accumulates information about all other VBSs and their lists of MTs and sends this information in its periodic hello messages. On the other hand, zone_MTs only accumulate information around them.

MTs announce their IP numbers with their periodic "hello" messages. An MT sends a merge-request message to another MT if the latter has a smaller IP number. The receiver of the merge-request responds with an accept-merge message, and sets its my_VBS variable to 0 to indicate that it has become a VBS. When the MT receives the accept-merge, it sets its my_VBS variable to the IP number of its VBS. If an MT hears from another MT whose IP number is smaller than that of its VBS, it sends a merge-request message to the former. When it receives an accept-merge message, it updates its my_VBS field. The MT then sends a dis-join message to its previous VBS (the dis-join message is only sent if an accept-merge was received, otherwise it won't be sent). When the old VBS receives the dis-join, it removes the sender from its list of MTs, which it is in charge of.

The Node-Weight Heuristic is proposed in [16-19]. Such heuristics consider the "ability" of the nodes, i.e., a node's transmission power, a node's processing speed, etc. The value of node-weights is assigned based on the suitability of a node to be a cluster-head. A node is chosen to be a cluster-head if its node-weight is higher than any of its neighbor's node-weights. If there are two candidates with the same highest node-weight existing in the same cluster, then a smaller node ID is chosen to break a tie.

One special case of Node-Weight Heuristic selection is the Power-Aware Virtual Base Stations (PA-VBS) protocol [18-19]. In this protocol, the election of PA-VBS is based on their current residual battery capacity that has been categorized into a couple of energy thresholds. In the scheme, there is a constant parameter to define the unit power of measurement – MAX_POWER. All the MTs measure their current Normalized Power Value (NPV) by dividing their current instantaneous battery capacity by the MAX_POWER. The MTs announce their NPV in their periodic hello messages. An MT sends a merge-request message to another MT if the latter has an NPV greater than or equal to the former's, and a predetermined threshold. The receiver of the merge-request responds with an accept-merge message only if its NPV is above the first threshold, namely THRESHOLD_1, at the time it receives the merge request message, in which case it increments its sequence number by 1 to reflect the change, and sets its myVBS variable to 0. When the MT receives the accept-merge, it sets its myVBS variable to the ID number of its new VBS. If an MT hears from another MT whose NPV is larger than that of its VBS, it does not send a merge-request message to it as long as its VBS's NPV is above THRESHOLD_1. A dis-join message is sent by an MT to its VBS only if the transmissions of the VBS have not been heard by the MT for some timeout period. If the VBS receives the dis-join message, it removes the sender from its list of MTs, which it is in charge of.

MTs broadcast their current NPVs as part of their hello messages. Hello messages contain other useful pieces of information, such as the iAmNoLongerYourVBS flag. The iAmNoLongerYourVBS flag is used by a node acting as a VBS (see Figure 2.1) to convey to the MTs it is currently in charge of whether it can support them for another hello period or not. When this flag is set to false, the MTs receiving the hello message know that they will still be served by their VBS for another hello period, and therefore do not need to look for a new one for at least one more hello period. However, this flag will be set by a PA-VBS to true when its NPV drops below the second energy threshold, namely THRESHOLD_2.


if myVBS$_i$ == 0 then [d];
if myVBS$_i$ == -1 then [a];
NMP$_i$
TH_1
if myVBS$_i$ == 0 then [c];
if myVBS$_i$ == 0 then [b];
TH_2
if myVBS$_i$ == 0 then [e];

[a]: send a merge request message to an MT with NPV greater than mine.
[b]: merge request messages can be sent by the MTs supported by this VBS, provided that the receiver's NPV is greater than TH_1.
[c]: discard any merge request.
[d]: no merge request messages will be sent by the MTs supported by this VBS.
[e]: set the iAmNoLongerYourVBS flag to true.

Figure 2.1 PA-VBS decision-making based on the energy

PA-VBS produces a larger number of cluster-heads than VBS [13]. This is actually because PA-VBS distributes the workload amongst a number of nodes that satisfy the energy requirements, unlike VBS which tends to elect the same set of nodes with the smallest Ids, again and again. The simulation experiments in [18] show that the growth in the number of cluster-heads is linear with the number of MTs in the case of PA-VBS. On the other hand, the number of cluster-heads in the case of VBS remains almost constant. With networks with much larger populations, this will cause considerable MAC delays due to the large number of MTs that are simultaneously contending for medium access at a constant rate.

Regardless of the value of THRESHOLD_1, since VBS elects nodes based on their ID numbers, VBS remain as cluster-heads for longer periods than in the case of PA-VBS. However, in practice, this cannot be achievable since cluster-heads consume more energy than other MTs, and their battery power drains quicker. Hence, VBS is more prone to undergo disorder. This implies that using VBS will drain all the battery power of the cluster-heads until they can no longer operate. On the other hand, PA-VBS achieves load balancing amongst the nodes in the wireless ad-hoc network. However, since node weights usually vary, computing cluster-heads can become very expensive.

2.2 MAC in wireless networks

The Medium Access Control (MAC) layer is a set of protocols that are responsible for maintaining order in the use of a shared medium. Several QoS-based MAC protocols have been proposed for cellular wireless networks [20-30]. These present variations of TDMA or CDMA to support multimedia traffic. These are not of interest to us in this work. In this thesis, we only focus on wireless local networks MAC protocols [31-46] that consider either QoS or priority with emphasis on the IEEE 802.11 standard protocol, which is designed mainly for infrastructure-based wireless networks.

2.2.1. IEEE 802.11

The IEEE 802.11 standard [31-35] lays the specifications of both the physical (PHY) and medium access control (MAC) layers of the network. The PHY layer, which actually handles the transmission of data between nodes, can use either direct sequence spread spectrum, frequency hopping spread spectrum, or infrared (IR) pulse position modulation. IEEE 802.11 makes provisions for data rates of 1 Mbps, 2 Mbps, 5.5 Mbps or 11 Mbps. It operates in the 2.4 - 2.4835 GHz frequency band (in the case of spread-spectrum transmission), which is an unlicensed band for industrial, scientific, and medical (ISM) applications, and 300 - 428,000 GHz for IR transmission. Infrared is generally considered to be more securing from eavesdropping because IR transmissions require absolute line-of-sight links (no transmission is possible around corners). Radio frequency transmissions, on the other hand, can penetrate walls and be unknowingly intercepted by third parties. However, infrared transmissions can be adversely affected by sunlight, and the spread-spectrum protocol of 802.11 can provide some rudimentary security for typical data transfers.

The IEEE 802.11 standard has defined two MAC functions: Distributed Coordination Function (DCF) and Point Coordination Function (PCF) to handle the MAC protocol in wireless networks [31,32]. Collision detection, as employed in Ethernet, cannot be used for the radio frequency transmissions of IEEE 802.11. The reason for this is that when a node is transmitting it cannot hear any other node in the system that may be transmitting, since its own signal will be much stronger than any other signal arriving at the node. In the IEEE 802.11 protocol, the access time has been separated into super frames, just like TDMA, and each super frame consists of two-time periods, a PCF period and a DCF period as shown in Figure 2.2. Therefore, the DCF and PCF will be in charge of sending and receiving data in turn during a super-frame.

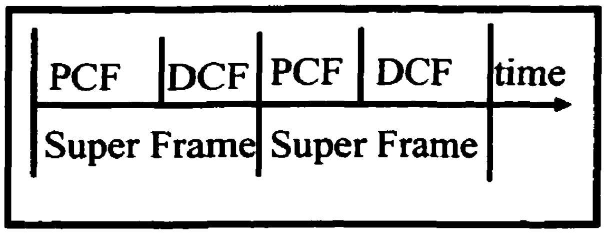


Figure 2.2 IEEE 802.11 MAC super frame

2.2.1.1. Distributed Coordination Function (DCF)

The distributed coordination function is the primary access protocol for the automatic sharing of the wireless medium between stations and access points. It uses the carrier-sense multiple access/collision avoidance (CSMA/CA) protocol for sharing the wireless medium. CSMA/CA is achieved by three main functions, (1) priority inter frame time space (2) random time slots, and (3) virtual sense mechanics. Every competing node, when it has some packets to send, first checks the Network Allocation Vector (NAV) to see whether it is zero - if it is non-zero (meaning another node already reserved the medium for a time period equal to the value in the NAV), it will wait. When the NAV is equal to zero, the sender begins to sense the medium. To avoid high collision probability when the channel first becomes clear (as more than one node may be waiting to send), the senders will sense the medium for some period of time; if the medium remains clear, the sender will send the packet after this waiting time period.

IEEE 802.11 has defined three different types of Inter Frame Space (IFS): DCF IFS (DIFS), PCF IFS (PIFS), and Short IFS (SIFS) [31,32]. DIFS is the IFS for the DCF. It is the longest time interval in these three types of IFS so it has the lowest priority when competing for the medium with the other two types of IFS. PIFS is IFS defined for the PCF. It is the medium time interval, so it has higher priority than DIFS. With the PIFS, the base station can gain control of the medium control without difficulty. The third one is SIFS, which has the highest priority with the shortest time interval. Some function types of data will use SIFS to compete for the medium because they can't suffer delay. For example, when a node sends out a packet, the only way to detect a collision is to wait for an Acknowledgement (ACK) from the destination node. If there is no ACK for some time period, it will consider that a collision has occurred and resend. Therefore, when the destination node receives a packet, it will send out an ACK immediately, otherwise the sending node will consider that a collision has occurred and send a duplicate packet. Therefore, the receiver uses SIFS to compete for the medium.

For example, as shown in Figure 2.3 (a), nodes "A", "B" and "C", are all mobile nodes and node "D" is an access point. As shown in Figure 2.3 (b), at time t_1, node "B" is trying to gain access to the medium by sensing it first. After waiting for DIFS until time t_2, the medium is still clear, so, node "B" sends packet 1 to node "C". While node "B" is

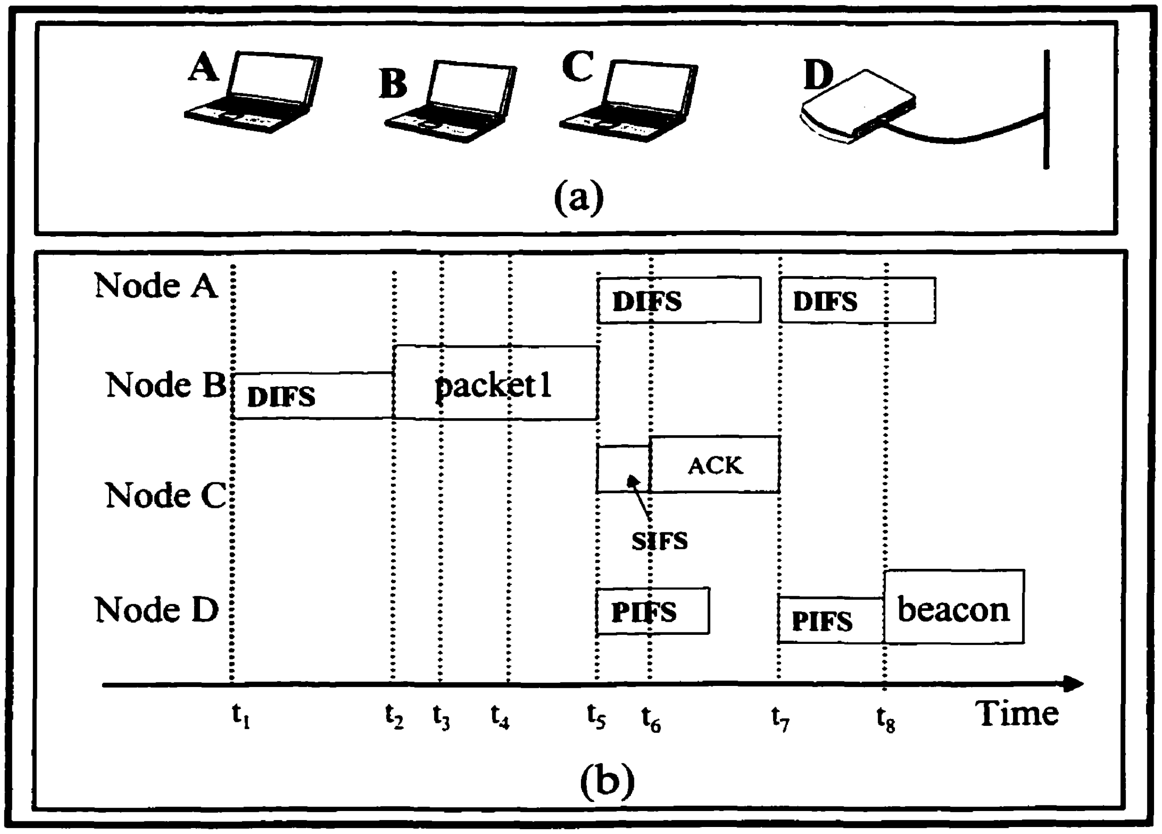


Figure 2.3 Example of priority in IEEE 802.11 MAC

sending packet 1, node "A" and node "D" both try to compete for the medium at time t_3 and t_4 respectively. Both sense that medium is busy till node "B" finishes sending at time t_5. At time t_5, the medium becomes clear again, but at this time, node "A" wants to compete for the medium to send the packet using DIFS, but node "D" (an access point) also wants to control the medium to initiate the PCF period, so, node "D" will use the PIFS which gives it higher priority than node "A". At the same time, node "C" also wants to respond with ACK to inform node "B" it successfully received packet 1. So it uses the SIFS to compete for the medium. In this case, node "C" will first reach the end of the sensing period and finds the medium is still clear, and will send ACK at time t_6. At the same time, node "A" and access point "D" will sense the medium as busy and then stand by for the next competition after node "C" finishes sending ACK at time t_7. From time t_7 on, the access point "D" will win the competition against node "A" and then announces the start of PCF by sending a beacon signal.

Another method is used in the IEEE 802.11 standard to avoid collisions when more than one MT competes for the medium with same DIFS interval. It is essentially an attempt to separate the total number of transmitting nodes into smaller groups, each using a different time slot (known as a back-off time slot). If the medium is detected to be idle, the node must first defer until the end of the DIFS interval and further wait for a random number of time slots (called the back-off interval) before attempting transmission. When retransmission is necessary, the back-off interval increases exponentially up to a certain threshold. Conversely, the back-off interval reduces to a minimum value when packets are transmitted successfully. At each back-off time slot, carrier sensing is performed to determine if there is activity on the medium. If the medium is idle for the duration of the slot, the back-off interval is decremented by one time slot. If a busy medium is detected, the back-off procedure is suspended and the back-off timer will not decrement for that slot. In this case, when the medium becomes idle again for a period of the DIFS, the back-off procedure continues decrementing from the time-slot, which was previously disrupted. This implies that the selected back-off interval now is less than the previous one. Hence, a packet that was delayed while performing the back-off procedure has a higher probability of being transmitted earlier than a newly arrived packet. The process is repeated until the back-off interval reaches zero and the packet is transmitted.

As Figure 2.4 shows, nodes "A", "B" and "C", are all mobile terminals located in the same communication range and node "A" and "B" both want to send a packet to node

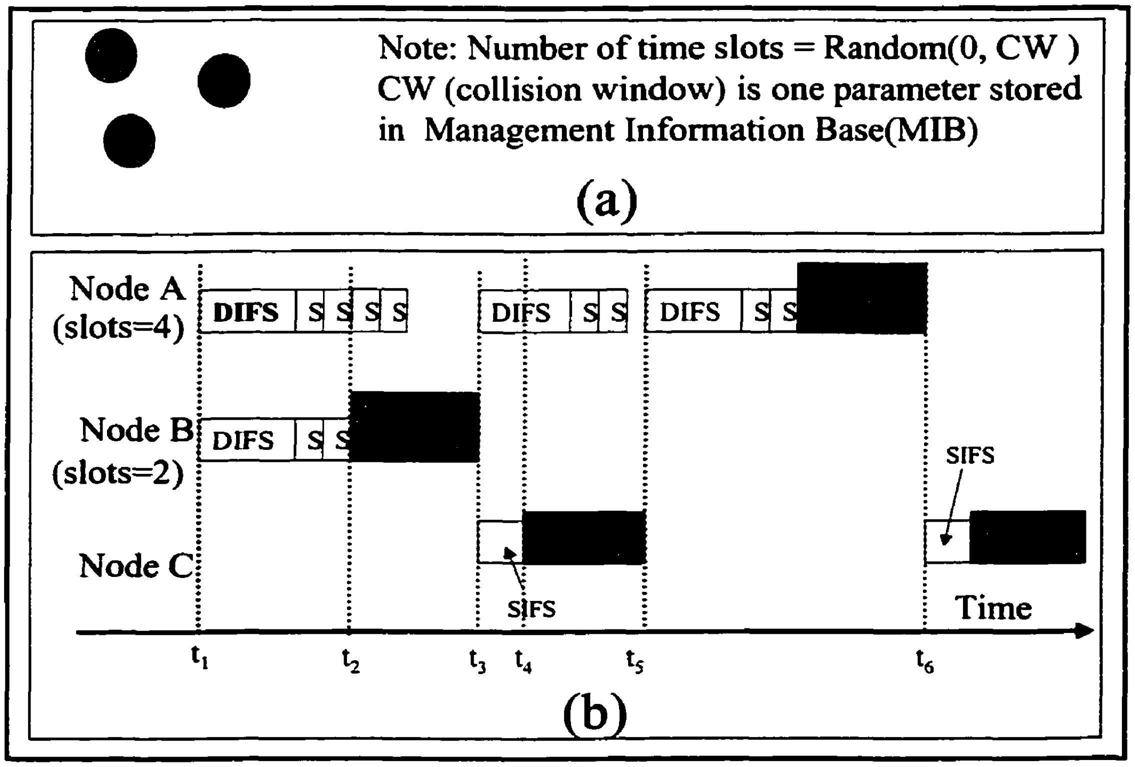

Note: Number of time slots = Random(0, CW)
CW (collision window) is one parameter stored in Management Information Base(MIB)
(a)
Node A (slots=4)
Node B (slots=2)
Node C
DIFS
S
SIFS
Time
t_1
t_2
t_3
t_4
t_5
t_6
(b)

Figure 2.4 Random time slots

"C". Before competing for the medium at time t_1, they both use the random function to create a time slot value that is from 0 to the value of the Collision Window (CW). CW is a parameter stored in the Management Information Base (MIB), whose value will be doubled or halved between a minimum and maximum value depending on the necessity of retransmission. Node "A" gets a slot time value equal to "4", but node "B" gets a value equal to "2". So, at time t_2, node "B" wins the competition but node "A" has already consumed two time-slots and has 2 time-slots left. After node "C" finishes responding the ACK at time t_5, node "A" will sense the medium for DIFS plus two time-slots. At this time, it gains access to the medium.

The priority and random time slots mechanisms can solve the competition problem when competing nodes can sense each other. But there is another problem called the "hidden node" problem [36], illustrated in Figure 2.5. Node "A" can communicate with node "B", and node "B" can communicate with node "C". However, node "A" cannot communicate with node "C". Thus, for instance, although node "A" may sense the channel to be clear, node "C" may in fact be transmitting to node "B". To ease the problem, a virtual sense mechanism is used in which the transmitting node first sends out a short ready-to-send (RTS) packet containing information on the length of the packet. If the receiving node hears the RTS, it responds with a short clear-to-send (CTS) packet. The CTS also contains information on the length of the packet. All the nodes in the transmission range of the source node and destination node hearing the RTS or CTS will set a value in the NAV to defer use of the medium for the length of time required to send the data packet. After this exchange of the data packet length information, the transmitting node sends its packet. When the packet is received successfully, as determined by a cyclic redundancy check (CRC), the receiving node transmits an acknowledgment (ACK) packet. This will guarantee that there is no collision in the data packet transmission but it also brings in additional "overhead". Therefore, only when the size of the data packet is larger than a certain threshold, will the sender use the virtual sense mechanism.

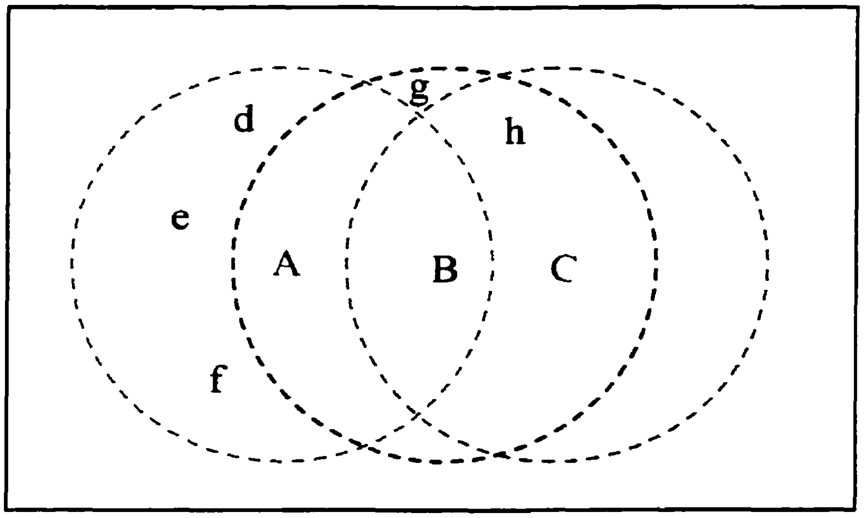


Figure 2.5 "Hidden node" problem

Figure 2.6 compares the data transmission with and without the virtual sense mechanism. In Figure 2.6, the section above the time line shows the data sent from node "A" to node "B" without using the virtual sense mechanism, while the section below the time line shows the data sent with virtual sense. In data transmission without virtual sense, if node

"C" sends in the time period t_0 to t_5, a collision will happen. With virtual sense, instead of sending the data packet directly, a node sends an RTS packet that sets a value equal to three times the SIFS plus packet length plus RTS plus ACK to all listeners' NAV. Then the destination responds with a CTS packet, which reserves the medium by setting all listeners' NAV to a value equal to two SIFS plus packet length plus ACK. Collision will also happen when node "C" sends data in the period from t_0 to t_3. Because of the much smaller size of RTS and CTS, the time interval t_0 to t_3 is usually much smaller than time interval t_0 to t_5. Thus the collision ratio of data transmission with virtual sense is much improved over direct data packet transmission. Another advantage of data transmission with virtual sense is that re-transmitting small-size RTS is more medium resource efficient than re-transmitting large-size data packets. The tradeoff in this case is that sending with virtual sense requires the extra time from time t_1 to time t_4.

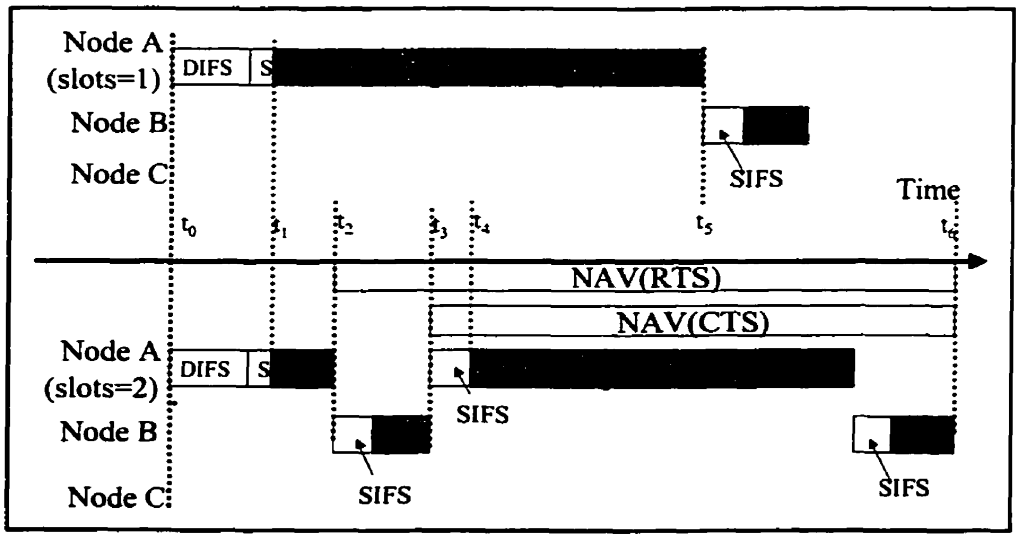


Figure 2.6 Packet transmission with virtual sense

There is another overhead that the virtual sense mechanism requires. In Figure 2.5, for example, node "C" is sending packet to "B" while node "A" wants to send a packet to node "B" using the virtual sense mechanism. Node "A" sends an RTS packet first; all the neighboring nodes like "e", "f" and "d" will all set to their NAV to a value, which is equal to the "time duration value" in RTS packet. If node "B" does not respond to the CTS packet because of a collision, nodes "d", "e" and "f" will suspend their DCF sending (if they have packets to send) for the time set in the NAV - even if the medium is clear.

2.2.1.2. Point Coordination Function (PCF) in IEEE 802.11

Real-time traffic (e.g., voice, video) requires bounded end-to-end delays beyond which the information loses its value and may be discarded. This is in contrast to the delay requirements for non-real-time data traffic, which are less stringent. DCF is not particularly suited for real time traffic support because it treats all packets equally without taking into account the sensitivity of certain types of data. Its connectionless nature does not schedule or prioritize time-sensitive real-time traffic and as a result it is unable to distinguish between such traffic and non-real-time traffic. The possibility of collisions, the use of a random back-off interval and the transmission of long packets may also lead to excessive delay variation (jitter). The use of a positive acknowledgement for collision and error detection in DCF may also degrade the transmission of real-time traffic since retransmission increases the delay.

Therefore, the IEEE 802.11 standard defines another function - Point Coordination Function (PCF) - to deal with real-time data [31-35]. The PCF is an optional connection-oriented capability within the standard. The PCF needs a Point Coordinator (PC) that initiates and controls the Contention-Free (CF) period where PCF is used. The PC first senses the channel for the PIFS period (getting priority over regular DIFS traffic) and then starts a CF period by broadcasting a beacon signal. This contains the possible maximum length of the CF period in the duration field of the data control header when it wins control of the medium. All terminals upon receiving the beacon will set a value (which is equal to "duration time" contained in the duration field of the beacon's control header) within their NAV. The length of the CF period depends on the coordination result from the neighboring PCs; the maximum value is the same as the super-frame. But if there is more than one PC using the same radio frequency, which may conflict with other PCs, then the length of the CF is smaller than the super-frame.

In the CF period, nodes will hold off sending any packets except when polled by the PC. When a node's turn comes, the PC sends a poll token to the node. The node then sends back any buffered real-time data after the SIFS period or an empty frame if it has no real-time data to send. If the terminal just responds with an empty frame, then the PC will not poll it again if there is time left when it has finished polling this round. Note that all packets are separated by the SIFS period. Priority polling mechanisms can be used if different polled users request different QoS levels. Users who are idle repeatedly are regarded as being shut down or moved out of range and are removed from the polling list in the next CF period. The PC can end the contention-free period at any time by transmitting a CF-end packet when the PCF duration time expires or the PC has no further frames to transmit or there are no more stations to poll.

In the cellular-like wireless network topology shown in the Figure 2.7, the access point naturally becomes the Point Coordinator (PC) because it is permanently located at a fixed place. When an MT turns on, it will first try to register to one access point. MTs in the overlapping area of more than one access point (like nodes "f" and "g") will choose one

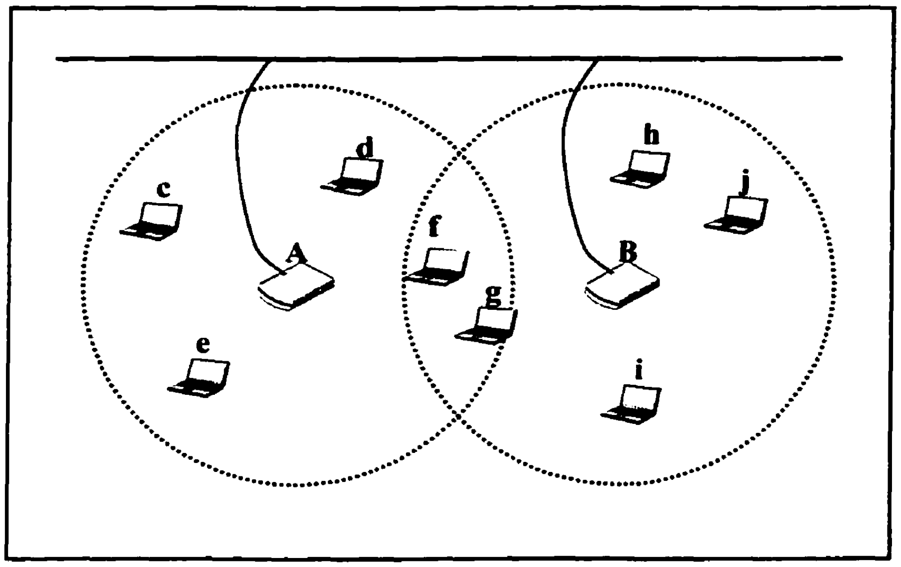


Figure 2.7 Cellular-like wireless local networks

access point depending on the received signal's strength from the access point. For instance, node "f" may register with "A", but node "g" may register with "B". When initialized by the access point, these two cells could use different radio frequencies to communicate between their own group members. Therefore, there is no collision in the overlapping area when these two access points go into the PCF period at the same time. If they use the same radio frequency, the access points can also coordinate between themselves through the wired connection. With the coordination, the maximum length of the CF period that access point "A" announces will be half of the super-frame, the next half of the super-frame will be assigned to the "B". For example, when access point "A" gains the medium and starts the PCF, system control by access point "B" can use the DCF to send packets. Before the CF duration time expires, "A" will poll its zone-MTs following the polling list – "c", "d", "e", "f", in a round robin fashion till all buffered real-time data has been sent. Finally, "A" will send the CF end signal to release the CF period.

2.2.2. Other MAC protocols

Several other MAC protocols have been proposed for local wireless and/or wireless ad-hoc networks. We review some of them below.

2.2.2.1. High Performance Radio Local Area Networks (HIPERLAN)

In Europe, ETSI (Europe Telecommunication Standardization Institute) also formed a group called HIPERLAN [37,38] to study a standard for local area networks that is different from the IEEE standard. The HIPERLAN is targeted to the more ambitions goal of supporting much higher bandwidths than IEEE 802.11 LAN, namely a high-speed (24 Mbps) wireless LAN standard for distributed networks. Any nodes that have data to send first sense the medium for the period required to transmit (1700 bit). If the medium is idle during this whole period, then the node transmits its packet immediately. If the channel is busy, it triggers the competition mechanism that has three phases: prioritization phase, contention phase (including elimination sub-phase and yield sub-phase), and transmission phase (see Figure 2.8).

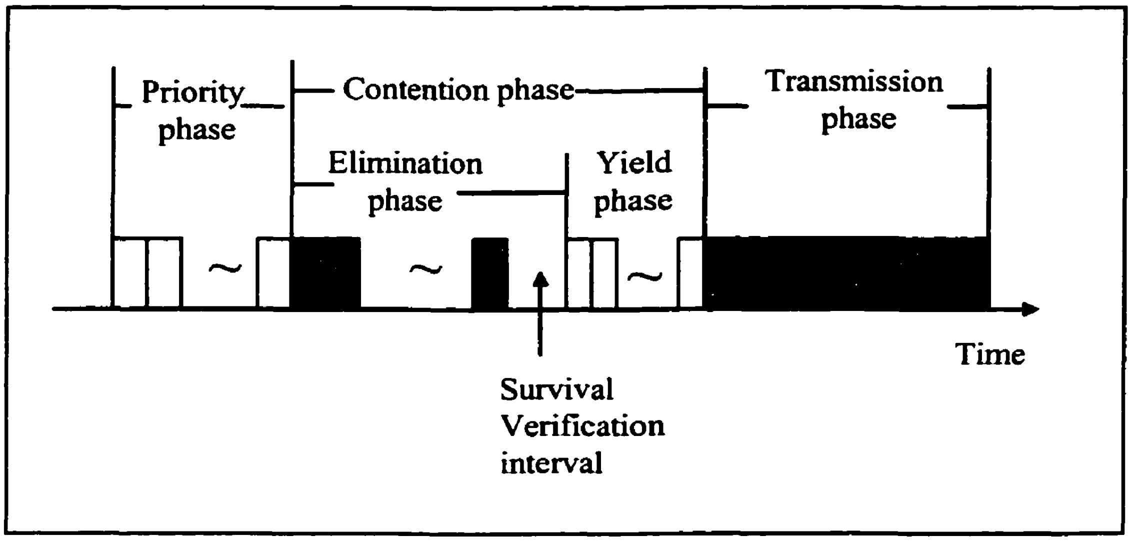


Figure 2.8 HIPERLAN MAC

The priority of a packet is derived from the duration of packets waiting in the queue divided by a time unit parameter. The more a packet is delayed, the higher its priority (less slot defer). In the priority phase, the node is sensing the medium and will quit the competition if the medium becomes busy. After sensing the medium to be free for the priority phase, the node transmits a noise signal for a random number of slots (Elimination phase), and at the end of Elimination phase, the node will sense the medium again - if the medium is busy, it will quit. After that is the Yield phase. During this phase, the node further senses the medium and defers for a random number of slots. If no transmission is detected, the node starts and completes its data transmission. Ideally, only one node can complete the whole process and send a packet.

An example is shown in Figure 2.9. At time t_0, there are four neighboring nodes "A", "B", "C" and "D" competing for the medium. Node "A", "B", and "C" have a priority defer that is 1 slot and begin to send a noise signal at time t_1. Node "D" has priority defer equal to 2 slots, thus at time t_2 it senses the medium is busy and quits. Nodes "A", "B" and "C" keep on competing during contention phase. Node "C" finishes the elimination sub-

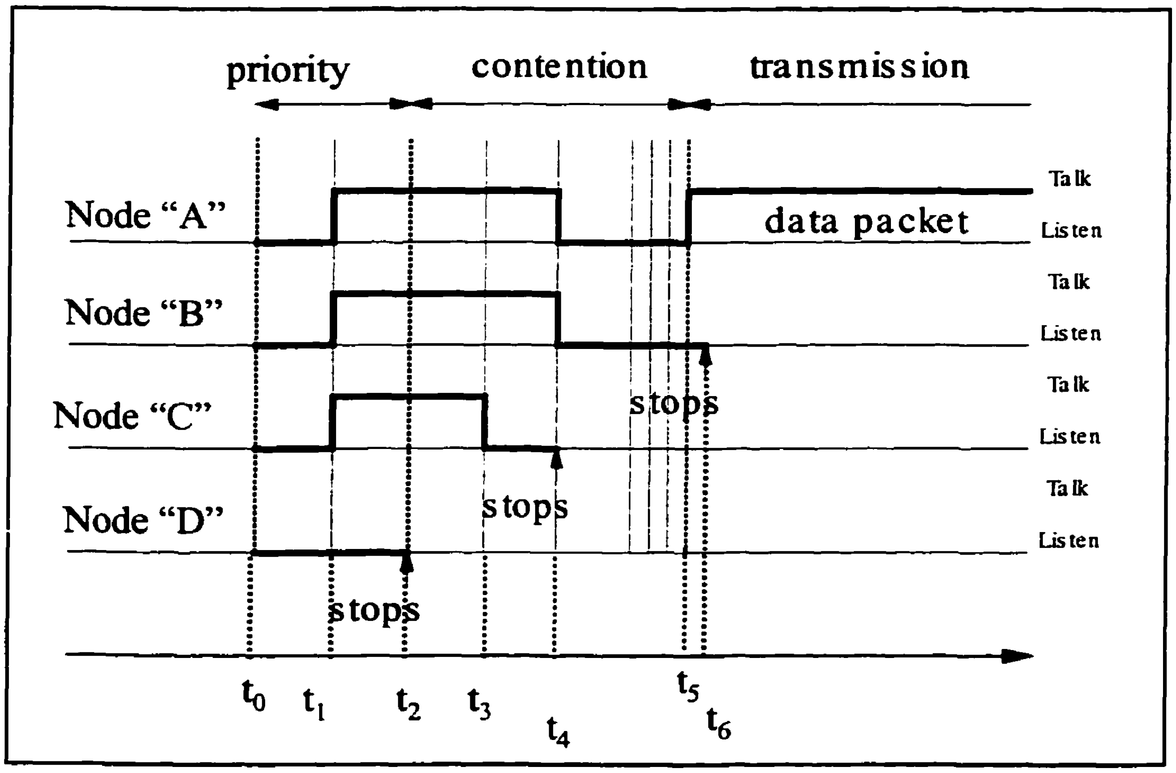


Figure 2.9 HIPERLAN MAC

phase earliest at time t_3, senses the medium is busy, and quits at time t_4. Node "A" and "B" finish the elimination sub-phase at the same time, verify the medium is idle and then step into the yield phase. Node "B" randomly creates a slot defer number is larger than node "A". So finally node "A" wins the contention and transmits the data at time t_5, and node "B" quits at time t_6. After the whole process, although it happens infrequently, collision could still happen. In the case above, if node "A" and node "B" create the same yield slot number, then they both will transmit the data at the same time.

2.2.2.2. Black burst mechanism

The black burst MAC protocol is proposed for wireless ad-hoc networks. Unlike cellular-like wireless networks, in an ad-hoc network, there are no fixed base stations, and every mobile terminal can be in a state of mobility. Packet transmission is based on distributed access. To give priority to real-time packets, a mechanism called Black Burst (BB) has been proposed [39,40] that modifies the IEEE 802.11 standard. Based on the CSMA/CA, the BB mechanics define three types of IFS: Short IFS (SIFS in IEEE 802.11), Medium IFS (PIFS in IEEE 802.11)), Long IFS (DIFS in IEEE 802.11). Instead of waiting for polling, real-time nodes contend for access to the channel after the Medium IFS rather than after the Long IFS used by data nodes. Thus, real-time nodes as a group have priority over data nodes. Real-time nodes first sort their access rights by jamming the channel with pulses of energy, denominated BB's, before sending real time packets. The length of BB transmitted by a real-time node is an increasing function of the contention delay experienced by the node, measured for the instant ($T_{duration}$) when an attempt to access the channel has been scheduled until the channel becomes idle for the Medium IFS. Similar to the "slot time" idea in the DCF, an integral number of BB slot units form the BB. The minimum BB slot number is "1", as it needs at least one BB slot unit to gain the channel first from the non-real-time data nodes. The BB duration is given by $BB_{duration} = 1 + \left\lfloor T_{duration} / T_{unit} \right\rfloor \times BB_{slot}$. Here, the $BB_{duration}$ is the period of time that the competing MTs are sending the BBs. $T_{duration}$ is the period of time that the MTs are waiting for the medium clear. T_{unit} is a constant parameter which defines the length of time used to count for the MTs waiting for the medium to clear. BB_{slot} is another constant parameter that defines the length of unit BB slot time .

After the BB period, a node with real-time data will sense the medium for the sensing time period to check if the medium is clear. If the channel is clear, it will send its real-time packet - otherwise it will quit the competition. Therefore, instead of a shorter slot time extension winning the medium in the non-real-time data group, longer BB duration will win access to the medium.

Figure 2.10 shows an example of the BB mechanism. The wireless network consists of four mobile terminals - A, B, C, and D. They can all communicate with each other. In this case, node "C" has a series of data packets to send to node "D". As node "C" sends one packet, node "A", plans to send a real-time packet at time t_0, and node "B" plans to send a real-time packet at time t_1. They both find the medium is busy and wait till time t_2.

At time t_2, node "D" will respond with ACK, this packet has highest priority. So, node "D" will win the competition over nodes "A" and "B", then send the ACK. When node "D" finishes sending the ACK at time t_4, node "C" schedules to send its next non-real-time data packet still using the Long IFS to compete, but nodes "A" and "B" use the Medium IFS to compete. At time t_5, real-time nodes "A" and "B" sense the medium is clear, and begin to send the BB signal, causing node "C" to quit the competition. Simultaneously, using the "BB equation", node "A" and node "B" calculate their respective BB durations. Node "B" calculates a value of 2, but node "A" calculates a

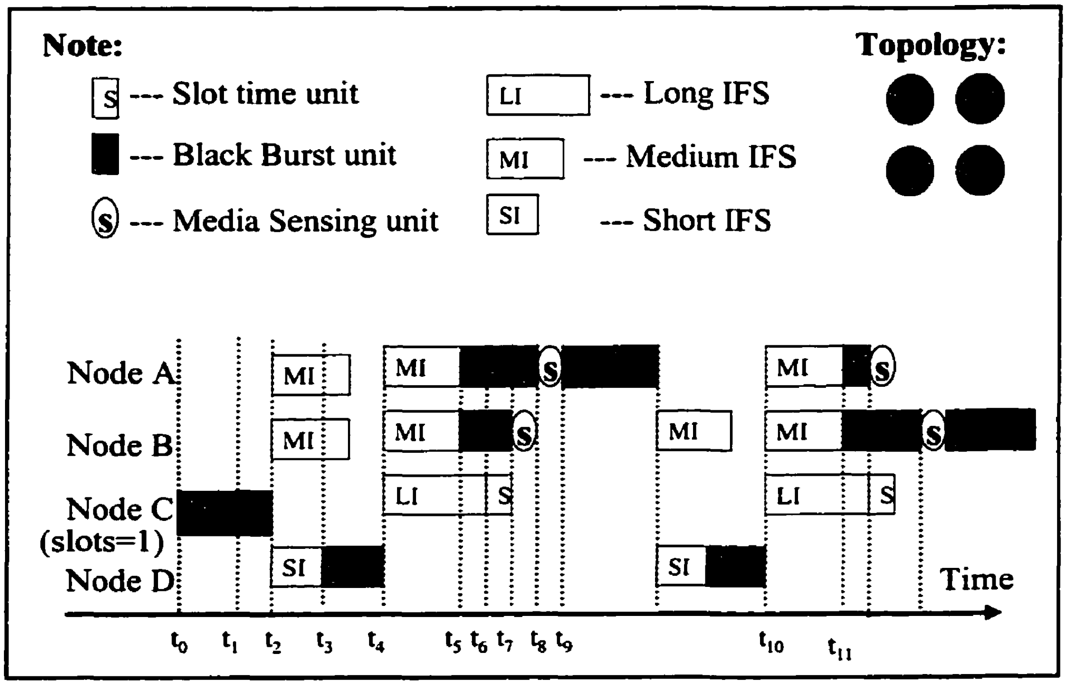


Figure 2.10 Black burst contention mechanisms

value of 3 because it has been waiting longer (time duration t_5-t_0 is longer than time duration t_5-t_1). After two BB time slots, at time t_7, node "B" senses the medium and finds the medium is busy so it quits the competition. After BB duration, node "A" senses the medium is clear, and sends its real-time packet. The next competition occurs from time t_{10} and node "B" has been waiting longer: its BB duration will be calculated from time interval time t_1 to time t_{11}, but node A will only calculate from time t_{10} to t_{11}. Thus node "B" will win access to the medium this time. From this example, we can see that the BB mechanism does not cause starvation for real time data.

2.2.2.3. Dual Busy Tone Multiple Access (DBTMA)

The "hidden terminal" problem described in the Section 2.2.1 is always a challenge that MAC protocols for ad-hoc networks want to solve. Related to the "hidden terminal" problem, there is another problem called the "exposed terminal" problem that also affects the efficient usage of the medium in the wireless networks. The opposite of the "hidden node", the "exposed node" is one that is within the range of sender but out of range of the destination. When such a node transmits data during the period that the sender is transmitting data, it does not interfere with the data transmission of the sender. Thus in theory, an exposed node can have a parallel conversation with another terminal out of range of the intended receiver to improve the utilization of the bandwidth.

The Dual Busy Tone Multiple Access (DBTMA) [41,42] uses two out-of-band tones to decouple communication in the two directions (see Figure 2.11). The entire channel has been separated into a control channel and a data channel. Data packets are transmitted over the data channel, while control packets (e.g., RTS and CTS) are transmitted over the control channel. The low frequency end of the channel is reserved to transmit a busy tone (BT_t), and the high frequency end of channel is reserved to receive a busy tone (BT_r). When the destination node receives the RTS packet and finds that it is able to receive the packet (e.g., the medium is clear), it then sets up its BT_r signal and replies with a CTS packet. Upon the source receiving the CTS packet, it sets up its BT_t signal and starts the data transmission. All the nodes in the transmission range of the receiving node can sense the BT_r signal and defer from transmitting. All the other nodes in the transmission range of the sender will determine that they cannot receive the other node's data. However, if a node cannot sense the BT_r signal, then it is capable of transmitting data. Through this mechanism, hidden terminals back-off and exposed terminals will be allowed to use the channel [42]. Hence, it can be seen that DBTMA defines the following set of communication rules:

1. Both the transmitter ("A") and receiver ("B") are in the Idle states before the transmission.

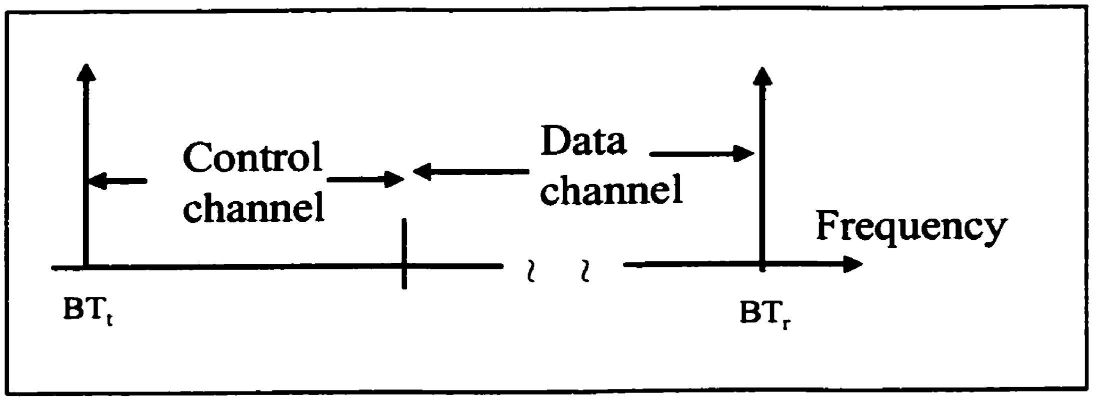


Figure 2.11 DBTMA frequency chart [42]

2. When “A” receives a data packet for transmission to the destination “B”, it goes into a state where it contends for medium access.
3. “A” tries to sense the BT_r signal. It backs off in the presence of a BT_r signal and goes to the next step otherwise.
4. “A” transmits an RTS packet, sets up a timer and waits for a CTS message from “B”.
5. “B” receives the RTS packet from “A”. It tries to sense the BT_r signal. It stays in the Idle state during the detection of BT_r signal and goes to the next step otherwise.
6. “B” sets up the BT_r signal and sends out a CTS packet. It then starts a timer and prepares to receive the packet.
7. “B” receives the data packet from “A”. It sets off the BT_r signal and returns to its idle state.

The disadvantage of DBTMA is that it wastes battery capacity by forcing the wireless node to continuously sense the medium for the BT_r and BT_r signals. And it reserves a special medium bandwidth as a control channel that keeps idle at the most of time. It also reserves a frequency for BT_r and BT_t (note: the frequency near the BT_r and BT_r should also be reserved to avoid noise), which will waste the bandwidth of the medium.

2.2.2.4 Multiple Access with Collision Avoidance By Invitation (MACA-BI)

To deal with the "hidden terminal" problem, Talucci proposed a protocol – MACA, by invitation (MACA – BI) [43]. It supposes that the receiver can predict its future reception times in a network with periodic data traffic. It initiates the reception by sending out a ready-to-receive (RTR) message to request data. All the other nodes that hear the RTR, will defer, thus solving the "hidden terminal" problem. However, since most transmission instances cannot be predicted due to the burst characteristic of data traffic, this mechanism just keeps silent in most cases, but once it predicts that neighbors have packets to send (i.e., piggyback "have more bits" by the previous packet), it will trigger the MACA-BI protocol to improve efficiency.

2.3 Summary

Recently, a lot of research has been done on local wireless MAC protocols. The IEEE 802.11 is the most popular MAC for wireless LANs. It consists of Distributed Coordination Function (DCF) and Point Coordination Function (PCF). The DCF is based on the CSMA/CA mechanism and thus works efficiently even without an access point controller. However, the proper operation of PCF needs a Point Coordinator (PC) to administer the data transmission among a group of MTs. In the IEEE 802.11 wireless local networks, the access point naturally is chosen to act as the PC. However, in the ad-hoc environment, there is no infrastructure and nodal mobility may be high. The lack of a point coordinator renders the PCF protocol of the IEEE 802.11 ineffective. So, transmission of all packets (non-real-time and real-time) is through the DCF. Real-time packets cannot then achieve their QoS requirement.

Other MAC protocols designed for the ad-hoc network are based on the principle that the ad-hoc network is totally distributed. So, the direction of design to improve the QoS for the real-time packets is by optimizing the CSMA/CA, trying to give the real-time packets higher priority when they compete for the medium. However, all such proposals do not conform to the IEEE 802.11 protocol, and hence pose compatibility and economic issues.

We have noted that there are great economic benefits in reusing all the existing functions defined in the IEEE 802.11 standard, and that central control can achieve the most effective management in the wireless network, not just in MAC but also in routing and many other functions. We, therefore, propose a Mobile Point Coordinator (MPC) selection protocol that will select some MTs as MPCs to function as access point coordinators in ad-hoc environments. Based on this MPC system, an efficient MAC protocol that is compatible with IEEE 802.11 has also been proposed. Our proposed framework is provided in Chapter 3.

CHAPTER 3

MOBILE POINT COORDINATOR MEDIUM ACCESS CONTROL

In this chapter, we first describe the MPC infrastructure-creation protocol (Section 3.1), and then describe the MPC-based MAC protocol (Section 3.2). Section 3.3 presents the chapter summary.

3.1. The MPC architecture: an introduction

In this section, the MPC infrastructure-creation protocol is described. Section 3.1.1 gives a detailed description of the MPC creation algorithm. Section 3.1.2 explains the protocol through a number of illustrations.

3.1.1 Description of the MPC protocol

In our scheme, some of the mobile terminals (MTs), based on an agreed-upon policy, regulate medium access using the Point Coordination Function (PCF) for all or a subset of their neighboring MTs. This is achieved by electing some to be Mobile Point Coordinators (MPC). In an ad-hoc wireless network, nodes are neither stationary nor are they permanently turned on. Besides, there are no wired connections. Hence, compared to a fixed cellular base station, an MPC is mobile, temporarily available and shares no wired connection to other MPCs.

3.1.1.1 Potential MPC problems

Due to the physical limitations of an MPC compared to a fixed base station, some MAC-related problems arise. These degrade the operation of the original IEEE 802.11 MAC scheme. We could categorize the problems into three types: (1) problems caused by the movement of the MPC, (2) problems caused by an MPC shut down, (3) problems caused by absence of coordination between neighboring MPCs.

In Figure 3.1, lowercase letters represent zone-MTs, while uppercase letters represent the MPCs, and the number associated with the letter indicates the time order. Nodes "a",

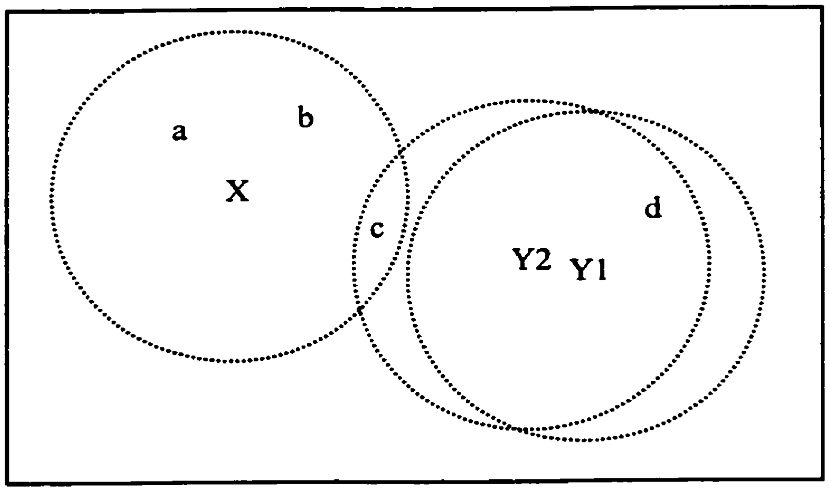


Figure 3.1 Example of problems caused by MPC mobility

"b", and "c" are associated with MPC "X" and node "d" is associated with MPC "Y". When "Y" is at the location Y1, nodes "X" and "Y" can start the PCF at any time, with no interference between them. However, when Y moves to Y2, node "c" will lie in the overlapping area between X and Y. If MPC "X" and MPC "Y" start the PCF at the same time, a collision will take place if "X" polls c while "Y" is transmitting.

Similarly, the movement of an MT could also affect the relationship between the neighboring MPCs as shown in the Figure 3.2. When "c", which was originally registered to "X" at location c1, moves to c2, a collision may happen at "c" if both MPCs are operating in the PCF mode at the same time. In addition, as shown in Figure 3.2, when MT "c" moves from c2 to c3, it will lose contact with its original MPC, "X", and will try to re-associate with MPC "Y". If "X" and "Y" were fixed base stations, the new access point "Y" would inform the old access point "X" of the re-association of node "c"

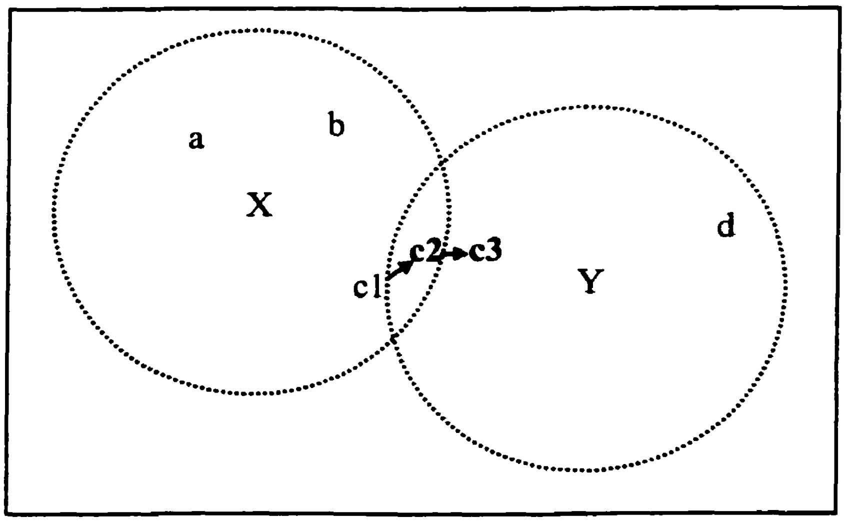


Figure 3.2 Example of problems caused by MT mobility

through the wired connection. However, in an ad-hoc network environment, "X" would not be informed of the new position of node "c".

In fixed cellular systems, base stations operate permanently. With the coordination between fixed neighboring access points, the next PCF period is always predictable. This will guarantee real-time packet sending and receipt within an acceptable delay bound. In the MPC system, however, MPCs could shut down at any time. If an MPC shuts down during the DCF period, then the zone-MT will never reach the CF period, and the performance of the system will degrade to that of a totally distributed one with no centralized control. In this case, MTs can only use DCF to send and receive data till the system selects a new MPC. If the MPC shuts down when it just sends out the beacon to reserve the medium, or when it finishes polling part of the zone-MT, a new problem arises. In this case, the zone MTs will never receive the CF-end signal to release the medium, and the medium resources will be wasted.

One potential solution is to select an MT as a temporary MPC to finish this round of the CF period. However, this causes another potential problem: can the temporary MPC cover all the MTs that registered to the original MPC? For example, in Figure 3.3, if MPC "A" shuts down suddenly after it announces the start of the PCF, every node will suspend the DCF while waiting for the poll. If a node (say node "b") recognizes the shut down of its MPC, it may carry out the duties of the previous MPC. Node "b" will be unable to poll nodes “g”, “i”, and “e”, which are out of the communication range of node “b”.

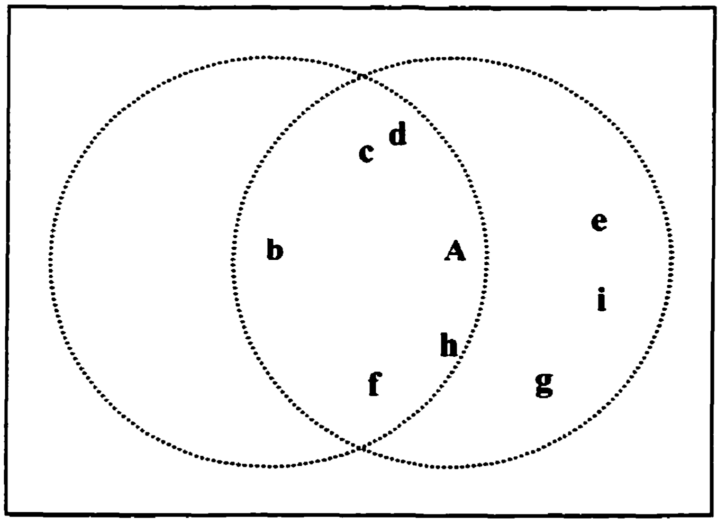


Figure 3.3 Potential problems caused by MPC shut down

Another possible problem is caused by the lack of effective coordination between neighboring MPCs. As shown in Figure 3.2 above, if there is a zone-MT in the overlap area of two (or more) MPCs, these MPCs should coordinate to avoid both polling MTs in the overlap area. Unlike fixed base stations, which could coordinate through their wired connection, the potential coordination method in the MPC system is to go through one MT in the overlap area if the neighboring MPCs cannot communicate directly. This will cause some potential problems, as illustrated in Figure 3.4. First, we need a method to select one and only one relay node in the overlap area if there is more than one node in this area (Say node “h” becomes the relay node between MPC “Y” and “Z”). A potential problem is that the relay node can also move or shut down. For example, if "h" moves from h1 to h2, it will lose contact with MPC "Y". If there are more than two MPCs, as shown in Figure 3.4, coordination between them becomes even more complicated. In this case, MPC "Y" has two neighboring MPCs , "X" and "Z", and needs to coordinate when to schedule the next PCF start time. If "Y" broadcasts coordination information, relay nodes between "Y" and "Z", and between "X" and "Y", will forward the coordination data to "X" and "Z" at the same time. If a relay node like node "h" moves from location h1 to h2 (outside the overlap area), then coordination between "Y" and "Z" will fail, but "Y" does not get any failure-related feedback. To guarantee successful MPC coordination, "Y" must send the coordination message to its neighboring MPCs one at a time.

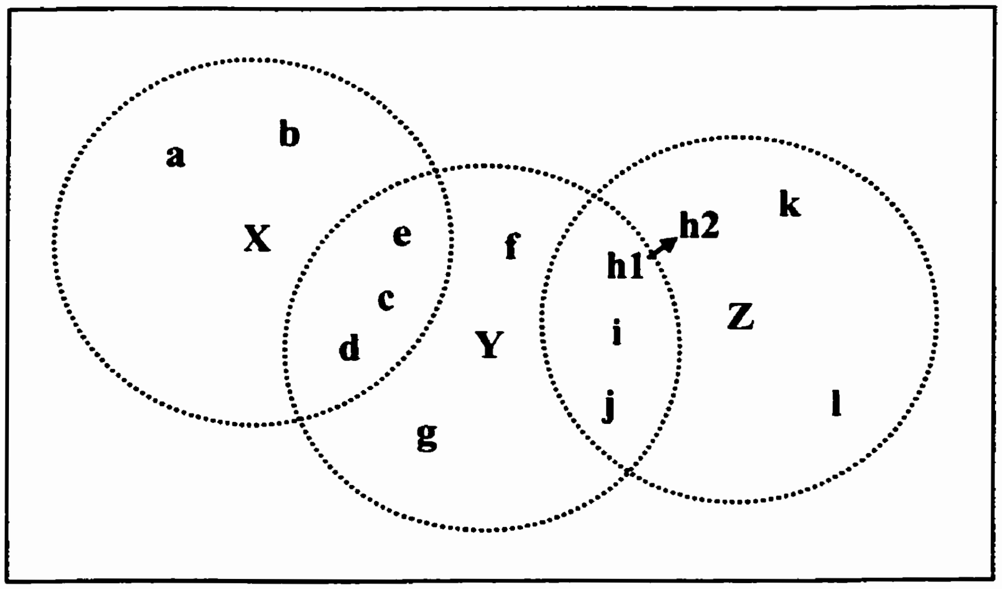


Figure 3.4 MPC coordination problem

3.1.1.2. Goal of MPC selection protocol

To overcome the potential problems discussed in the previous subsection, we propose to use a different range for MPC nodes than the normal communication range. In our scheme, The MT can be in one of three modes of operation: (1) free MT (an MPC with no registered MTs), (2) zone-MT (registered to an MPC other than itself) or (3) MPC. As Figure 3.5 shows, node "A" is an MPC, and nodes "b" and "c" are zone-MTs that are registered to node "A". Every MT, regardless of its current operating mode, lies in the center of two imaginary circles. In Figure 3.5, the area within the solid circle around node

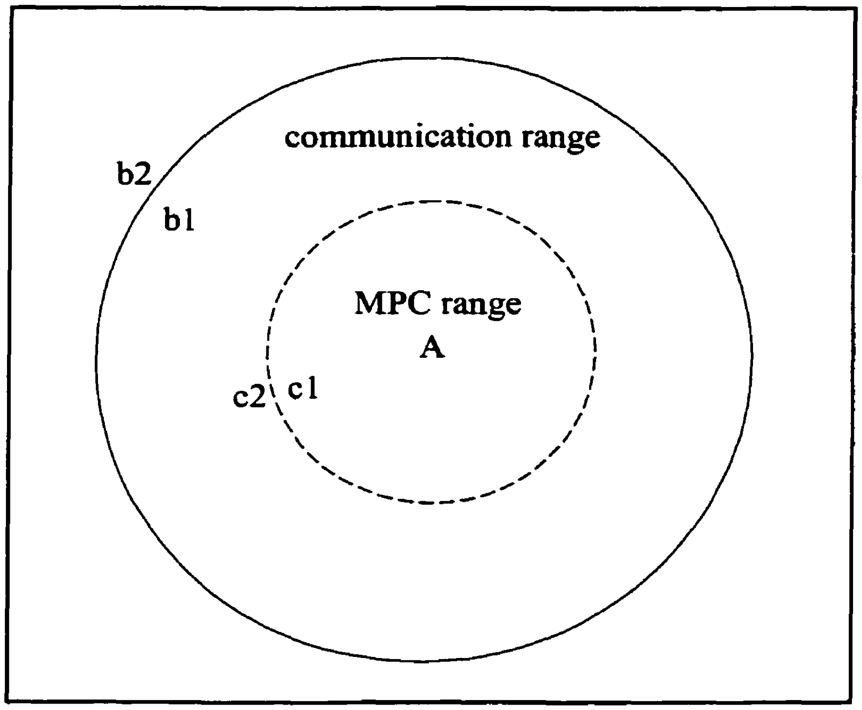


Figure 3.5 Communication range and MPC range

"A" is the Communication Range (resembling the so-called "cell" in cellular networks) of node "A". All other nodes in this range, such as "b" at time b1 and "c" at time c1 can communicate with node "A" directly. The dashed circle, whose radius is half that of the communication circle defines the MPC range of node "A". Ideally, all the MTs that are registered to MPC "A" are within its MPC range. Every node declares its existence by sending a "hello" message, and the receiver of the "hello" message decides in which area the sender is located, depending on the strength of the received radio signal. If the receiver can hear the signal, then the sender must be in its Communication Range; if the received signal is stronger than the value of the so-called MPC candidate threshold, then the sender is within the receiver's MPC Range.

In an ad-hoc wireless mobile environment, MTs are moving in and out neighboring MTs' MPC range frequently. One special case is when MTs stay on or vibrate on the MPC candidate threshold. To optimize the stability of our dual range MPC system, we can set up a neutral area between the inner range and outer range. If the MT wants to register to another MT, the latter's signal must reach the MPC candidate threshold + Δ, where Δ is a constant. If an MT wants to disjoin from an MPC, the signal_strength must reach, or be below, the MPC candidate threshold minus Δ. As Figure 3.6 shows, node "a", which is originally registered to "A", moves in a zigzag pattern exactly on the border of the inner circle. In Figure 3.6 (a), node "a" will be kept busy by repeatedly registering and deregistering to MPC "A". But, as shown in Figure 3.6 (b), "a" remains registered to "A".

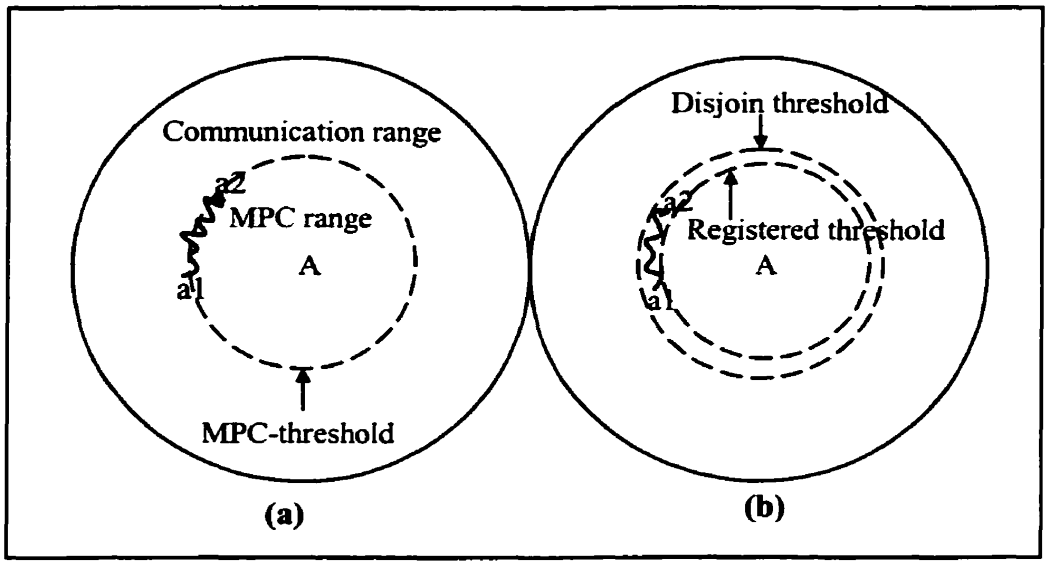


Figure 3.6 Improving system performances using the signal thresholds

3.1.1.3. Advantages of the dual range MPC architecture

With this dual-range MPC architecture, we could solve, or at least ease, the problems caused by the limitations of ad-hoc wireless networks, which have been described earlier.

As shown in Figure 3.7,

- It will help the MPC to better track MTs. When the zone-MT re-associates with a new MPC or becomes a free MT, it may be still within the communication range of the original MPC, and can hence inform the old MPC of its new affiliation. For example, node "a" after moving from a1 to a2, and registering to MPC "Y", can still inform node "X" of the re-association.

➢ MPCs can communicate directly. Note that the radius of the inner circle is half or less than that of the outer circle. If the inner circles have overlapping area, which is a situation that may cause problems, the MPCs can coordinate with each other directly. If two MPCs, such as "Y" and "Z" in Figure 3.7 above, are outside the communication range of each other and start the PCF period at the same time, then the overlap area of the dotted circle and the solid circle is much smaller than the overlap area of the two solid circles. Thus movements like node "b" moving from b1 to b2 have no effect on the relationship between MPCs "Y" and "Z".

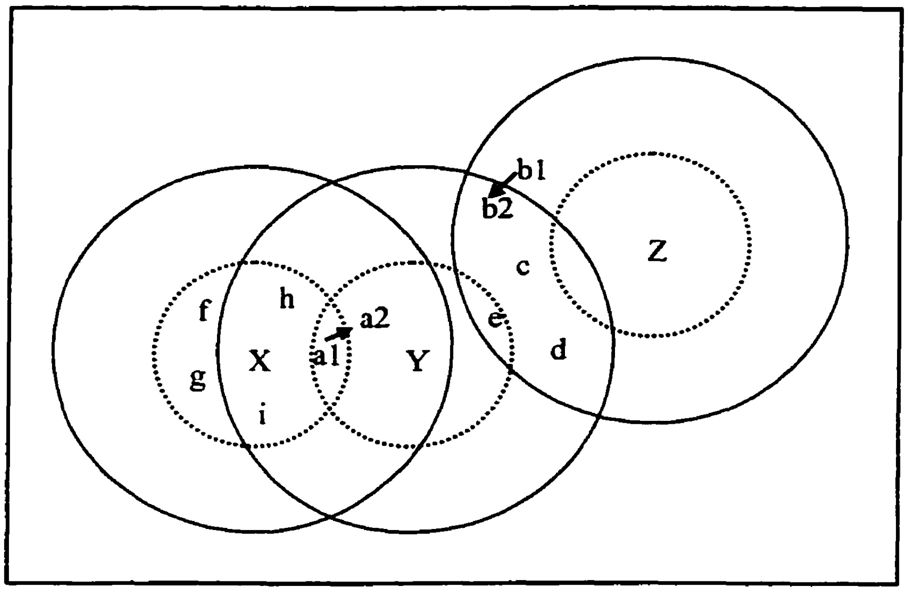


Figure 3.7 Alleviated problematic scenario

Having an MPC range smaller than the communication range eases the collision problem between the neighboring MPCs, but does not solve it completely. For example, if "X" and "Y" start the PCF at the same time, then a collision may still happen at node "e" instead of nodes "b", "c", "d", and "e".

➢ Due to the smaller inner range, all group members of a certain MPC can contact each other within one "hop". Therefore, if an MT suspects that the MPC was shut down, then it can take over the job of the MPC to finish polling.

3.1.1.4. MPC system parameters and components

When an MT is turned on, it will broadcast its so-called "hello" message periodically. The "hello" message contains the node's local information and information about the wireless ad-hoc network. Explanations of the fields contained in the hello message are listed below:

- **My_ID** – this is the MAC ID of the MT - and is unique to every MT.
- **My_MPC** – this parameter indicates the ID of the MT's MPC. If the MT is a zone-MT, its My_MPC variable will set to the ID of the MPC to which it is registered. If the MT is an MPC it will be set to "0". If the MT is a free MT, its My_MPC variable will be also set to "0".
- **My_sequence_num** – any connectivity change will result in an increase of the sender's My_sequence_num. It is initialed to "0" when the MT is turned on.
- **My_reg_MT_num** - this variable stores the number of the zone-MTs that are currently registered to an MPC. Both the zone-MTs and the free MTs will have their My_reg_MT_num set to "0".

- **My_neighboring_MPC_num** - this variable parameter counts the number of the MPCs or free MTs that are in this MT's communication range.
- **Probing_MT_list** - this parameter contains the IDs of all the zone-MTs that are registered to a particular MPC. Both the zone-MTs and free-MTs will set this parameter value to null.

Other parameters that need to maintained by all MTs include:

- **inner_outer_threshold**_(MPC candidate threshold) – A constant radio signal strength value. If the receiving radio signal is stronger than this value, the sender will be considered currently in my MPC range, and thus suits being an MPC candidate.
- **need_MT_list_search_for_MPC** – A flag used to indicate where to find the best MPC. When the MT turns on or the zone-MT disjoin from an MPC, this value is set to "true".
- **MT list** – A table maintained in every MT's MIB to store information of the neighboring MTs, which are currently in its communication range. Each entry will contain the following: (1) My_ID (2) My_MPC (3) My_reg_MT_num (4) My_sequence_num (5) My_neighboring_MPC_num (6) Last_received_time (7) received_signal_strength. The first five parameters are exchanged by the "hello" message. The Last_received_time records the time of the last message received from a neighboring MT. The received_signal_strength records the signal strength of last message received from this MT.

- **"Data sending" queue** – A queue maintained by every MT's MAC layer to store the non-real-time, real-time, and management data packet, prior to transmission.
- **"Control frame" stack** – Another buffer in the MAC layer used to buffer the control frames that cannot suffer delay such as ACK, and CTS. The "control frame" stack has higher priority than the "data sending" queue in data sending.

3.1.1.5. MPC selection

Figure 3.8 shows the process of MPC selection. All the MTs will periodically broadcast "hello" messages by calling the hello_from() routine (see Appendix A). All the MTs in the sender's communication range can receive the "hello" messages if there is no collision. The receiver will record or update the sender's local information contained within the "hello" message, including the receiving time and signal strength by calling the hello_receipt_from() routine (see Appendix A). If the MT does not receive any signal from a certain MT that is listed in its "MT list" within a timeout period, it will regard the target MT as out of its communication range and will delete it from its "MT list". Another time that a MT can realize a neighboring MT has shut down or has moved out of its communication range is when it sends packet to this destination MT without receiving an ACK for several times continuously. This will be further described later in Section 3.2.

After having been turned on for a period of time exceeding the "observing period", as an MT needs enough time to collect the neighboring MTs' information to avoid selecting an

MPC depending on partial information, the MT selects an MPC from its list of possible candidates.

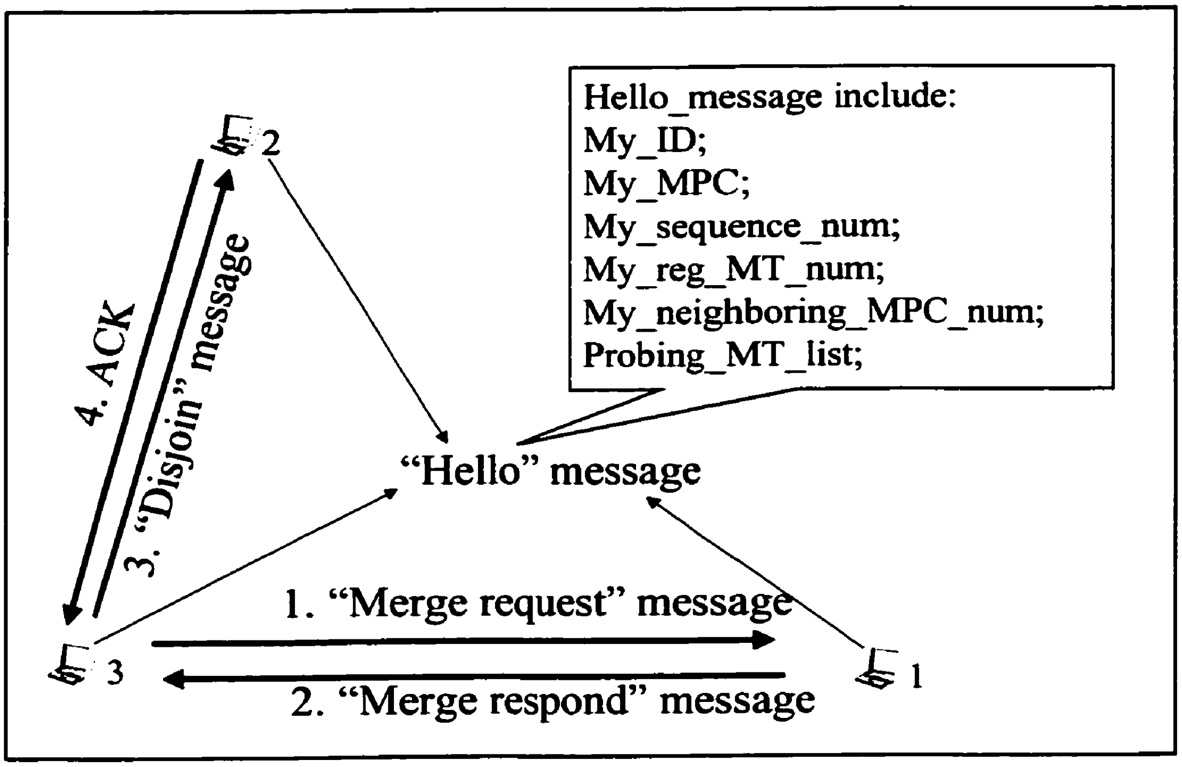


Figure 3.8 MPC selection mechanism

The criteria upon which an MPC is chosen are as follows:

I. An MT whose My_MPC value is "0" will join an MPC whose signal strength is at or above the inner_outer_threshold.

II. MPCs with largest My_reg_MT_num are given preference. This MPC selection criterion tends to discourage multiple cluster-heads within MPC range of each other.

According to this criterion, if two MPCs move within MPC range of each other, the smaller group will finally merge into the larger group.

III. MPCs that can communicate with a large number of other MPCs are given preference. This is because the MPC with a higher My_neighbor_MPC_num has more MPCs or free MTs to coordinate with directly.

IV. If the all the above three criteria are the same for two or more candidate MPCs, then the MT will select the MPC with the lowest ID.

After choosing the candidate MPC, if the candidate is not itself, the MT will send a "merge request" message to the candidate MPC during the DCF period by calling the merge_request_from() routine (see Appendix A). The candidate MPC will call the merge_request_receipt_from() routine (see Appendix A), and will either send back a "merge response" message with "merge accept" or "merge reject" information. The latter occurs only when there is a limitation on the number of zone MTs (access control mechanism) or if the receiver has moved out of the MPC range of the sender. If the sender of the "merge request" message does not receive the "merge response" message on time, it will retry until it receives the response message. If the merge request is accepted, then the MT will change my_MPC to reflect its new MPC and increase the My_sequence_num by 1. The MT will send a "disjoin" message by calling the "disjoin_from()" routine (see Appendix A), to its old MPC, if it had one. The old MPC will respond with an ACK to announce the receipt of the message and call the disjoin_receipt_from() (see Appendix A) routine to update the topology parameters. The previous MPC's My_sequence_num will be incremented by one, and its My_reg_MT_num will also change to reflect the revised number of group members.

3.1.2. MPC illustrated

This section explains the operation of the MPC infrastructure-creation protocol by means of illustrations. Each of the following examples represents a scenario, and its set of corresponding actions performed by the mobile terminals running the MPC protocol.

In Figure 3.9, at time "0", nodes "X" and "Y" are MPCs; nodes "a", "b", "c", and "d", are all registered to "X", so the My_reg_MT_num of "X" is 4. Node "e" is registered to "Y", thus My_reg_MT_num of "Y" is 1. When node "f" is turned on at time "1", after exchanging "hello" messages, it finds out that there are three candidates: "X", "Y", and itself, which are in the MPC range of node "f".

However, the zone-MTs of node "X" are 4 compared to 1 for node "y" and 0 for itself. Therefore, node "f" will register to "X" by sending a "merge request" message, and in turn will receive a "merge accept" message. Thus, MPC "X" will have 5 MTs registered to it. At time "2", node "g" turns on; finding no existing MPCs in its MPC range and so becomes a free MT. Node "h" turns on at time "3", the free MTs "g" and "h" become the MPC candidates because they exist in the MPC range of each other. They both are free MTs, as there are no MTs registered to them at that time. Based on the My_neighboring_MPC_num, node "h" can reach two MPCs, "X" and "Y", but node "g" can only reach a single MPC, “X”, directly, so node “g” becomes a zone-MT of the new MPC “h”.

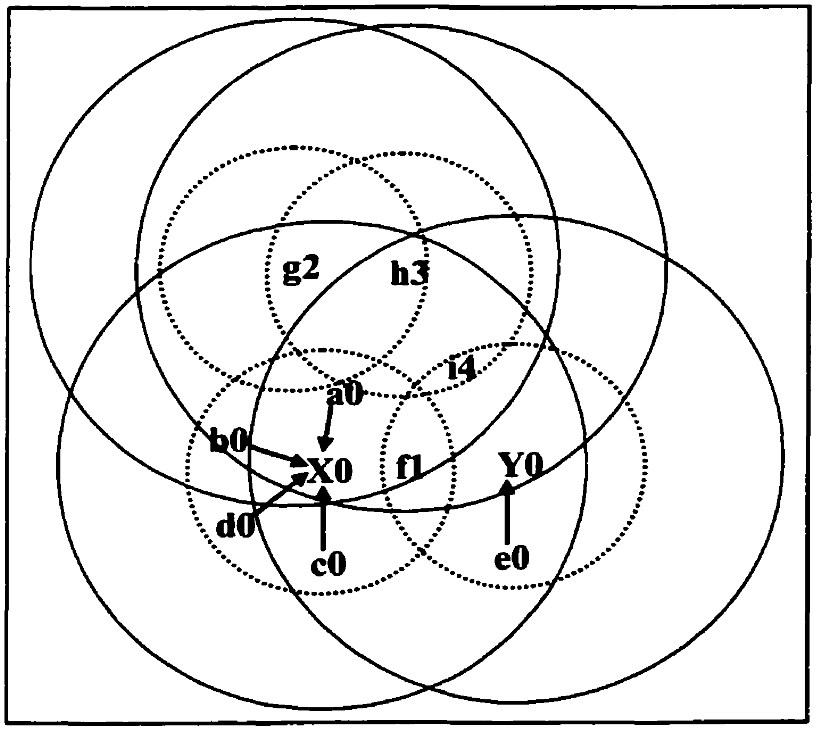
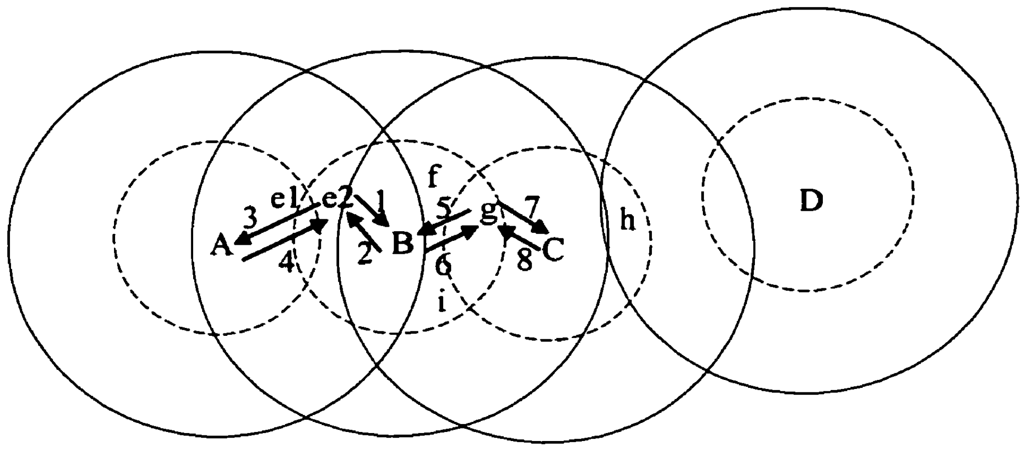


Figure 3.9 Finding the MPCs

When node “i” is turned on at time “4”, it has three MPC candidates: “h”, “Y” and itself to choose from. Node “h” and “Y” tie on My_reg_MT_num, (both equal to 1), and My_neighboring_MPC_num, which are equal to 2. It has to use the fourth MPC selection criterion (smallest ID) to arbitrarily choose one as its MPC (that is node “h”). So, node “i” will register to MPC “h”.

Sometimes, a zone-MT may need to disjoin from the old MPC and re-associate with a new MPC or just disjoin from the original MPC. This is due to one of the following reasons:

- When the MT moves out of the MPC range of its original MPC. This is shown in Figure 3.10 for node "e" which was originally registered to MPC "A" at location e1.



Note:

1: merge-request	2:merge-accept	3: disjoin	4: ACK
5: merge-request	6:merge-accept	7: disjoin	8: ACK

Figure 3.10 Changing MPC affiliation

- If a zone-MT finds out that a neighboring MPC is more suitable than its current MPC. In Figure 3.10, "e" was initially registered to "A", so the My_reg_MT_num of "A" is equal to 1. Nodes "f", "i" are registered to "B", so the

My_reg_MT_num of "B" is equal to 2. Node "g" and "h" are registered to "C", so the My_reg_MT_num of "C" is also equal to 2. After node "e" re-associates with "B", the My_reg_MT_num of "B" increases to 3. Node "g" will now find that "B" is more suitable to become its MPC because the My_reg_MT_num of "B" is larger than that that of "C". Node "g" will re-associate with "B" and disjoin from "C".

The second MPC selection criteria (largest My_reg_MT_num), with the help of the re-association mechanism, causes the smaller MPC group to become smaller and finally disappear. In Figure 3.11, at time "0", there are three MPCs, "X", "Y", and "Z". Nodes "a", "b", "c", and "d", are all registered to "X". Nodes "e", and "f" are both registered to "Y", and nodes "g" and "h" are registered to "Z". When node "f" moves from f0 to f1, at time "1", MPC "X" will be within MPC range of node "f", and it finds that "X" has four MTs registered to it, but its original MPC "Y" has only two MTs registered to it. Node "f" will send a "merge request" message to "X", and node "X" responds with a "merge response" message. Node "f" will then send a "disjoin" message to "Y", node "Y" will respond with an ACK message. MPC "X" will now have five MTs registered to it, while MPC "Y" will only have one MT registered to it. As the result of exchanging information by "hello message", the next step of "MT leaving" will occur. Node "e" now finds that MPC "Z" has more MTs registered to it than its original MPC, "Y", and re-associate with "Z". MPC "Y" now becomes a free MT because no MT is registered to it. It will also be registered to MPC "Z" and finally become a zone-MT. Therefore, the whole MPC group of "Y" has been absorbed by its neighboring MPC groups, "X" and "Z", and disappears.

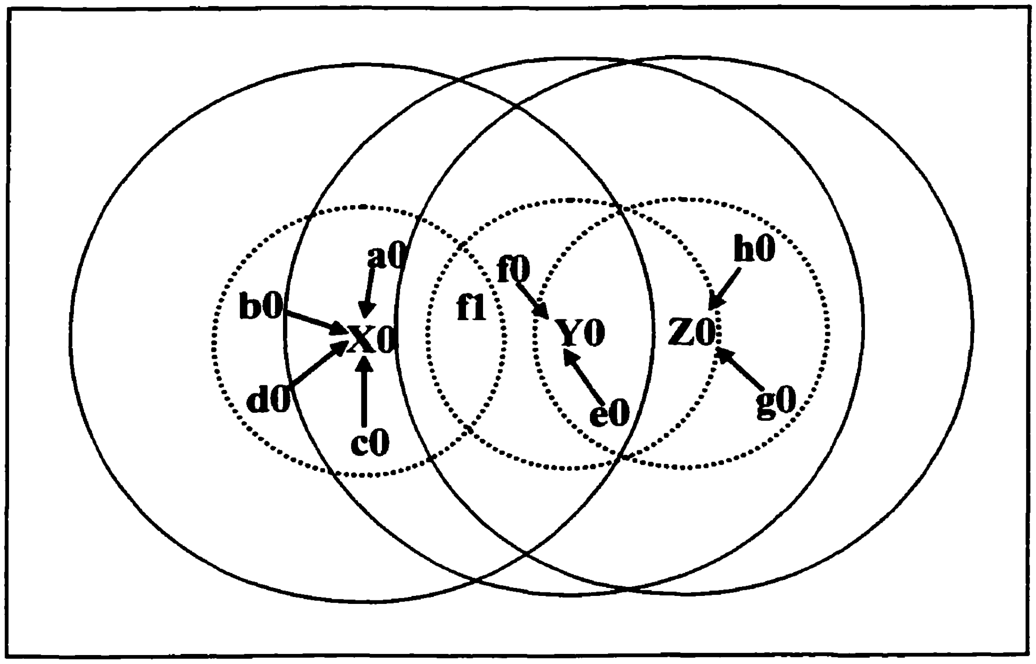


Figure 3.11 Re-association of MPC

Figure 3.12 shows the case when the MPC is shut down during the PCF period. Originally, there are two MPCs, "X" and "Y", in this network. Nodes "a", "b", "c", "d", and "e", are registered to MPC "X", and Nodes "f", and "g", are registered to MPC "Y". Immediately after MPC "X" initiates the PCF period by sending the proper beacon, "X" is shut down. In this case, another MT, say node "a", becomes the new MPC temporarily in charge of nodes "b", "c", "d", and "f", to finish this round of the PCF period. Every MT that originally registered to "X" will reset their My_MPC parameter to the ID of node “a”, and increase their My_sequence_num by 1. After the PCF period, they go into the common process of MPC formation: some nodes, like “d” and “e”, find that they are outside the MPC range of “a”, the new MPC; Node “d” thus tries to associate with “Y”. Node “e” finds itself to be the best candidate to be an MPC, and thus becomes a free MT.

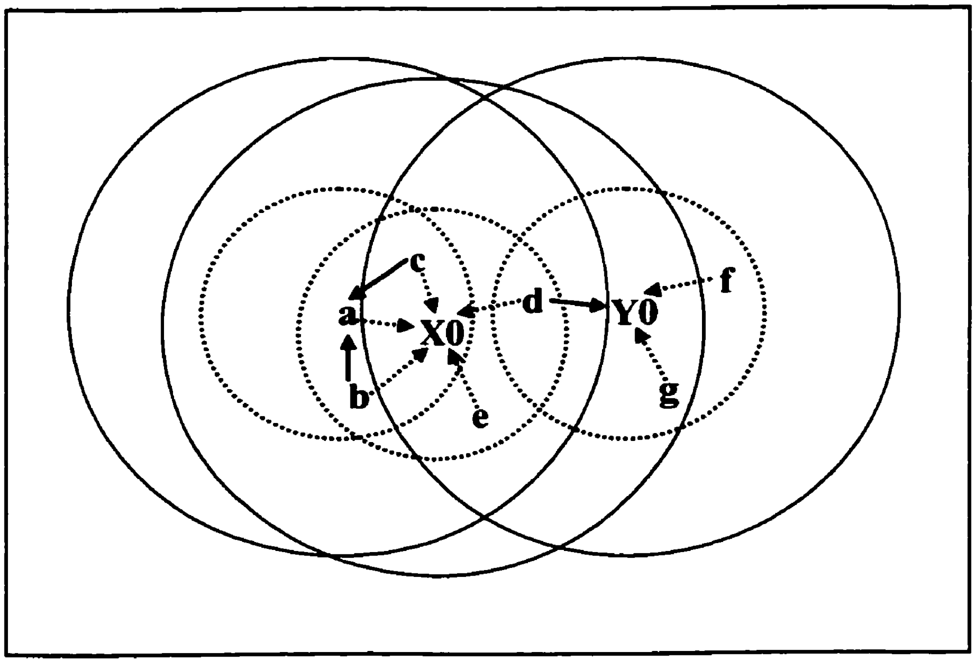


Figure 3.12 shut down of MPC

3.2. MPC-MAC

In this section, we describe an IEEE 802.11 compatible MAC protocol for wireless ad-hoc networks. Our protocol is based on the MPC infrastructure-creation scheme described in the previous section. Section 3.2.1 gives an overview of the protocol. Section 3.2.2 describes the MPC-MAC system parameters. Section 3.2.3 provides a description of the protocol, while Section 3.2.4 explains the protocol by illustrations.

3.2.1. Protocol overview

This MPC-MAC protocol utilizes the MPC system presented in Section 3.1. When an MT turns on, it is initialized as a free MT. All MTs run the MPC creation protocol, and some are elected to act as MPCs. The MAC protocol consists of DCF and PCF components. There is no synchronization between different MTs.

3.2.1.1. Basic DCF and PCF in MPC-MAC

In the DCF active period, every MT, regardless of its mode of operation, can compete to acquire the medium for data packet transmission. In DCF mode, not only non-real-time data packets, management data packets, and control packets can be sent, but also real-time data packets. However, during the PCF active period, all the MTs, except for MPCs and free MTs, are prevented from competing for the medium until being polled by their MPC. When an MT is polled by its MPC, it will search for real-time data packet to send. If the polled MT has no real-time packet to send, it will answer with an empty data packet. Therefore, when the DCF and PCF act in turn, the real-time data packets have a better QoS level because they can be sent at any time of the super-frame period. On the other hand, non-real-time packets can only be sent during the DCF active period. If the PCF controller is an MPC, it will send its buffered real-time data packet and poll message to poll the MTs, which are registered to it in a round robin manner. A free MT just sends its buffered real-time data packets during the PCF period. If the nodal density becomes heavy in a single communication cell, to increase the efficiency of PCF, the MPC may poll part of zone-MTs in its polling list. Here, a call admission control mechanism would play a very important role in separating the zone-MTs that have been given admission, from all the zone-MTs in the polling list.

As described in Section 3.1.1.4, in the MAC layer, the data packet buffer consists of a "data sending" queue and a "control frame" stack. If the buffer is not empty, it will trigger the DCF to work. The DCF will try to send all the data packets in the "control frame" stack first, then the packets in the "data sending" queue one by one when the value of its NAV is equal to 0. Before sending any data packets in the "data sending" queue, it should make sure that the "control frame" stack is empty.

As in the IEEE 802.11 MAC standard, each MT is in PCF mode once per super-frame time period. A MPC will suspend the DCF (by setting its NAV to a non-zero value), and will initiate the PCF period. After controlling the medium, the MPC sends its buffer's real-time data packets, if any, and polls the MTs registered to it one by one. A Free MT will check the "data sending" queue first for real-time data. If it does not find any real-time packets in the queue, it does not initiate a PCF period. However, if it finds real-time data packets in its buffer, it will suspend its DCF period, and go into the PCF active period. Therefore, the PCF active period for a free MT is only for sending its own real-time packets.

Within the MPC-based MAC framework, several MPCs and free MTs can simultaneously have the PCF initiation rights, and may co-exist in the same communication range. As shown in Figure 3.13, MPCs "A", "B" and free MT "c" co-exist in each other's communication range. They compete for the medium using the PIFS, which is shorter than the DIFS. So, they always beat their neighboring nodes that run the DCF. However, collisions may happen among the PCF initiators if they

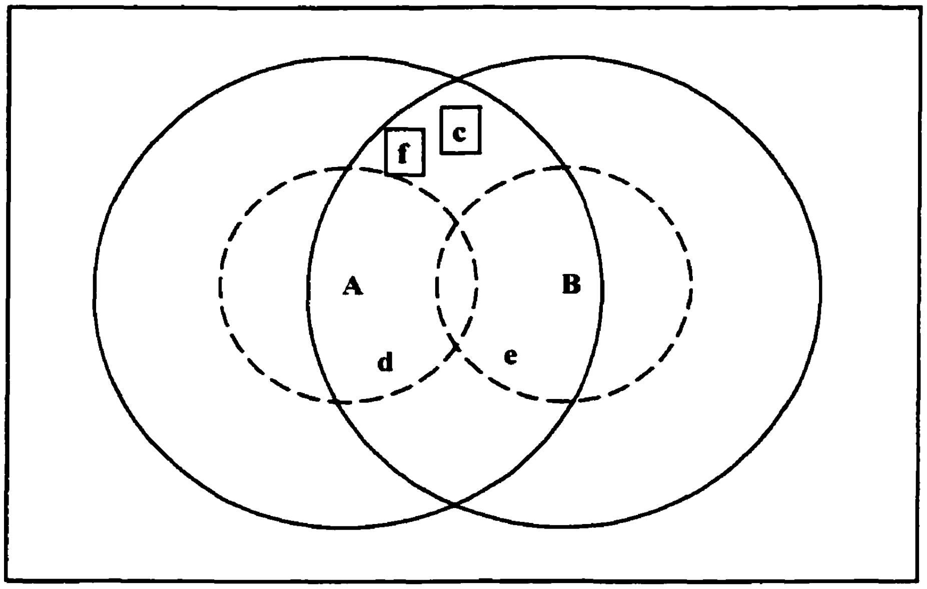


Figure 3.13 Three roles of MT

simultaneously contend for the medium using the PIFS only. To solve this problem, we propose to use the random Slot Defer Number (SDN), as is done in the IEEE 802.11

DCF, to extend the PIFS to several time slots to avoid collision among the MPCs and free MTs.

However, even when the randomly created SDN is used to extend the PIFS, collisions may still take place, or even the PCF of an MPC or a free MT may lose the competition to the DCF of a neighboring zone-MT. Figure 3.14 shows a scenario in which a PCF initiator suffers collision, and also a scenario where it loses the competition with DCF nodes. At time t_0, MPC "A" plans to initiate PCF with an SDN value of 1, but node "d" wants to send packets in DCF with an SDN value 0. At time t_1, MPC "A" senses the medium and realizes that it is occupied by node "d", thus it loses medium competition to a DCF node. At time t_3, the medium becomes idle again, but now node "e" joins the medium acquisition competition with "A". The medium sensing time of node "A" is the same as that of node "e". Therefore, node "A" sends the beacon signal, while node "e" sends its "RTS" packet. They both suffer collision. Because the "beacon signal" is a very short data packet, node "A" finishes transmitting earlier at time t_5, senses that medium is busy, realizes that it has suffered collision and reschedules the next medium competition immediately. At time t_6, node "e" finishes sending the "RTS" packet, and waits for the "CTS" packet (which will not arrive because of the collision). At the same time, node "A" senses the medium becoming idle again, and finally it successfully initializes the PCF by sending a "beacon signal" at time t_7. After "A" finishes its PCF at time t_8, MPC "B" and free MT "c" plan to schedule their PCF at the same time, and they unfortunately choose the same random SDN, and suffer collision by sending the beacon signal at the same time. Because the length of the "beacon signal" is the same, they finish sending at the same time, but there is no feedback to show that a collision has occurred. Hence, there is no way to detect the collision.

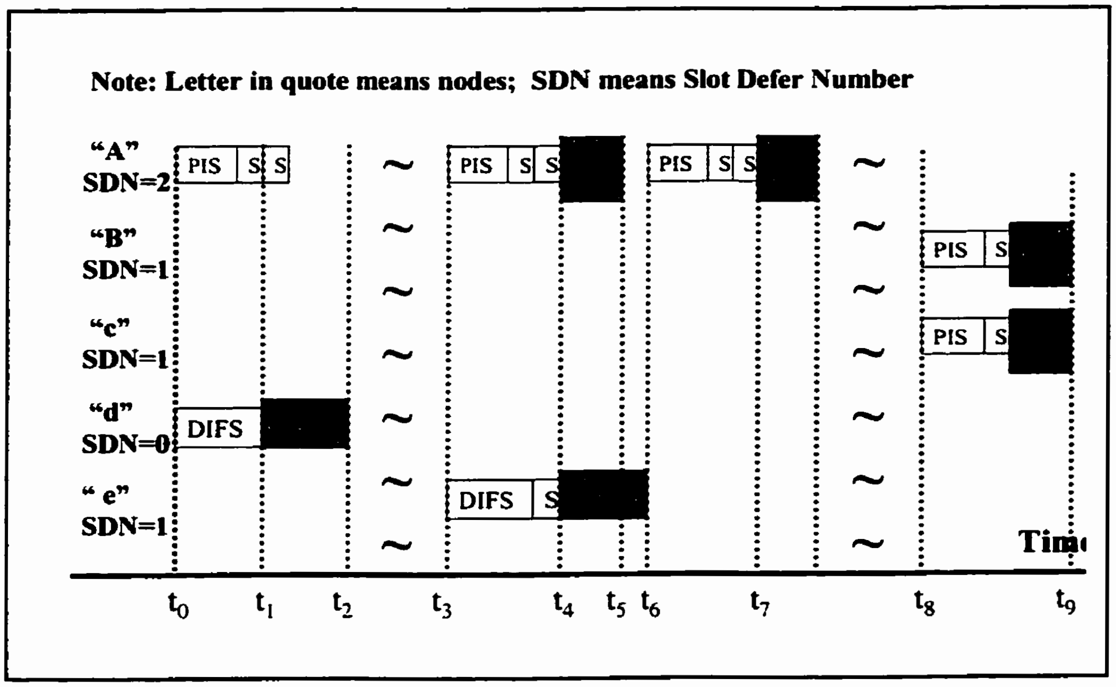


Figure 3.14 PIFS collision

3.2.1.2. Enhanced PCF mechanism

Unlike the DCF, in which the data transmission is unpredictable, the beginning of a PCF period is predictable, because the time interval between the PCF initial time points is about the length of the super-frame time interval. We can use this characteristic to avoid PCF initiation collisions. When it is time for an MPC or a free MT to initiate the PCF, it will first check the recent PCF records of its neighboring MPCs and free MTs, estimate its possible competitors for the medium, and create a competitor list. The position of a competitor in the list will match its last PCF initial order sequence. The SDN will be the same as the position number in the list. Normally, if the real-time traffic is light, PCF periods of co-existing MPCs and free MTs will spread sparsely in one super-frame time period. As Figure 3.15 shows, in the first super-frame period (time t_0 to t_3), real-time traffic is light, each initiator's PCF duration is short, and they all fit in the predicted time frame established by history (for example, node "A" finishes PCF at time t_1 before the predicted PCF initial time of node "B" which is at time t_2). In this period, any PCF initiator when trying to initiate the PCF will find that there are no candidates to compete with for medium access. The PCF initiators will use the PIFS with an SDN value of 0 to compete for the medium. If the real-time traffic is light, the PCF initiators will always have higher priority over users of DCF and no collision will take place between the PCF initiators.

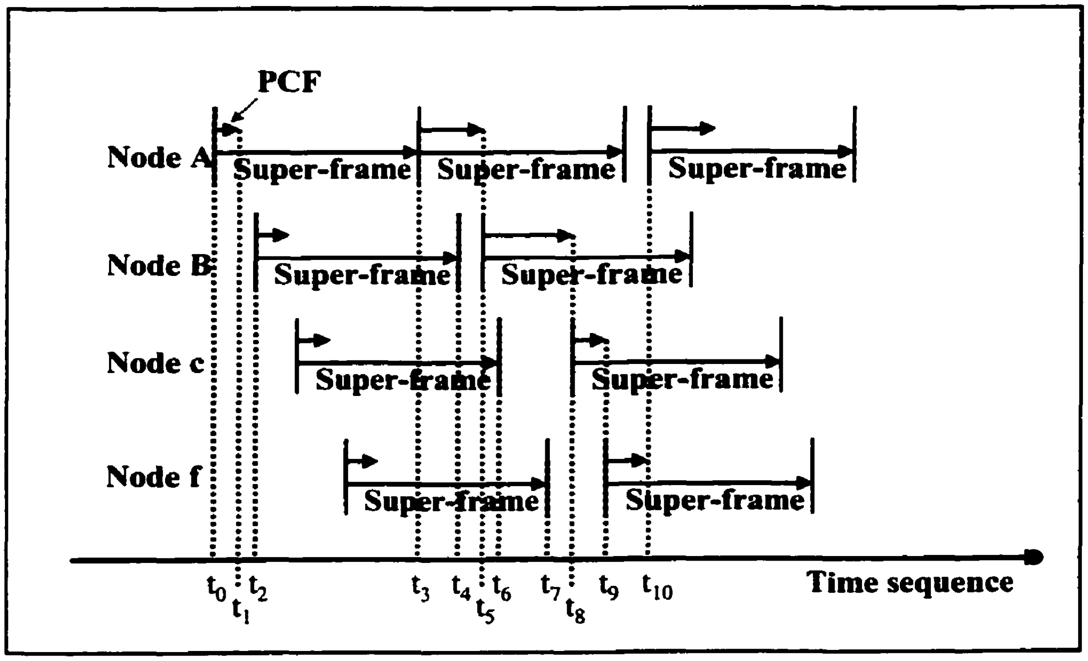

PCF
Node A
Node B
Node c
Node f
Super-frame
t_0 t_1 t_2 t_3 t_4 t_5 t_6 t_7 t_8 t_9 t_{10}
Time sequence

Figure 3.15 The enhanced PCF mechanisms

If the real-time traffic becomes heavy, some PCF durations will become longer. An initiator's PCF period may extend over the predicted start time of the next initiator's PCF. As shown in Figure 3.15, in the second round, the real-time traffic becomes heavy, the PCF period of node "A" lasts up to time t_5 (beyond the second predicted PCF start time, t_4). When "B" initiates the PCF in the second round, it still gets no other candidate to compete with it, thus it waits only for PIFS and then gains control over the medium. When node "B" finishes its PCF at time t_8, it is beyond its predicted time, and there are two nodes, "c" and "f", to compete for the medium. Node "c" finds that it is the first candidate in the predicted sequence, thus it gets an SDN of "0", whereas node "f" gets an SDN of "1". Node "c" gains access to the medium. If node "c" is shut down or just quits the PCF, node "f" has the chance to acquire the medium. If all nodes suffer postponement of their predicted PCF initiation time, this causes wasted bandwidth. Such an unpredictable situation cannot even be controlled in cellular-like wireless LANs, and is beyond the scope of this thesis.

In our protocol, the MPC will inform all its group members of the list and the order by which the MTs will be polled using the "hello" message. When the MPC announces the start of the PCF, all the group members will keep track of which MT will be polled next. When the PCF starts, every member MT will copy the polling MT list into volatile memory. With the process of polling, deleting the member that responds back with an empty data frame will modify the temporary polling list. With this method, every member can calculate the time it will be polled. If the MT does not hear the polling signal, it assumes its MPC must have been shut down, or might be out of range. In this case, another group member can take over as an MPC. Since all zone MTs maintain the polling order, the new MPC can continue the current PCF period. An example will be illustrated in Section 3.2.4.

3.2.2. MPC-MAC system parameters

The following system parameters are used by our MPC-based MAC protocol:

- **my_next_PCF_poll** – A variable that will be initialized when the zone MT receives the beacon from its MPC. It will be modified when it receives the information from the MPC and count down with the time going. When it reaches "0", the MT will doubt its MPC has shut down. During the PCF period, it always has a positive value, but during the DCF period, its value will be set to "-1".
- **polling_list** – A table maintained by every MT to keep track of its group members that have the same MPC as itself. The MPC will update it and broadcast it in the "hello" message.
- **unfinished_polling_list** – For a specific PCF period, this list is used to track which MT in the group has sent all its real-time data, and hence will not be polled again.
- **RTSThreshold** –If the data packet size is larger than the value of this constant, the virtual sense mechanism will be triggered.

- **my_MPC_is_in_PCF** – A boolean variable used to indicate whether the group the MT belongs to is currently in PCF mode or not.

3.2.3. Description of the MPC-MAC protocol

Here we will describe our MPC-MAC. When a packet arrives in the buffer, it will trigger the DCF_send() routine (see Appendix B), which will send the control frame packet in the "control frame" stack using SIFS to compete for the medium. The transmission of a packet in the "control frame" stack is not limited by the NAV and state mode. Even if the NAV is not "0" or the MT is in PCF mode, the transmission of control data like ACK, CTS is not deferred. So, in our MPC-MAC protocol, after the polled MT sends a real-time packet, before the MPC polls the next MT, a gap (about SIFS+ time to send an ACK) should be reserved to let the destination MT confirm with ACK. Using DCF_send() to send the packet in the "data sending" queue, will be limited in the DCF mode, and the NAV is set to "0". If the condition is not satisfied, the routine will suspend itself till the NAV reaches "0" or the MT goes into DCF mode. When competing for the medium for the packet in the "data sending" queue, the routine will use DIFS + randomly created SDN. After winning the medium, it will send the packet directly (for a small size packet) or send "RTS" first (for a large size packet). In the latter case, all the MTs around it will call the RTS_receipt_from() routine (see Appendix B) to set a value in NAV. If the receiver is the destination of the RTS, it will send the CTS packet. Similarly, the receiver of the CTS will call CTS_receipt_from() routine (see Appendix B) to set the NAV.

In the MPC system, there can be several PCF initiators co-existing together, which need to share a super-frame time period. Therefore, in an MPC system, in a super-frame, there may be several PCF periods and DCF periods fragmented and interleaved together. The MT uses "my_MPC_is_in_PCF" to coordinate the DCF and PCF in the super-frame. This variable is set to "true" in only one PCF period fragment that is initiated by its native MPC, and is set to "false" at other times. When the MT receives a beacon signal (no matter whether the sender is its MPC or not), in its NAV, a value equal to the length of PCF duration will be set. The same thing happens, when it receives the "CF_end signal", in its NAV, the corresponding value will be reset to "0". If the sender of the "CF_end signal" is the MT's MPC, then the variable "my_MPC_is_in_PCF" will be set to "false".

In the MPC, the PCF_initiate() routine (see Appendix B) will be activated periodically. It uses PIFS and a calculated SDN to compete for the medium. The method to create SDN is different from that in DCF, which is created randomly. In PCF, the MPC could check the neighboring MPCs' beacon record for "SDN's value equal to the number of MPC around me which should have sent a beacon but still have not till now". After this MPC wins the medium, it will send the beacon signal to initiate a PCF period. All the MTs around it will call the PCF_Beacon_receipt_from() routine (see Appendix B) to reserve the medium by setting NAV and initiating my_next_PCF_poll, estimating the time to being polled that may trigger an MT to replace its MPC if the MT is not polled for a certain period of time. When the MT replaces the MPC in the PCF period, it will call the PCF_send() routine (see Appendix B) to finish the current PCF. After it initiates the

PCF, the MPC will call the PCF_send() routine to poll and send real-time packets within the MPC group. All the group members, including the MPC, will maintain their unfinished_polling_list, and the MPC will follow the sequence in the list to poll the MTs, one by one, by sending poll tokens to the MT. All the group members will call the poll_receipt_from() routine to update the time to be polled. If the MT is the destination of the poll token, it will send a real-time packet from its buffer or send an empty frame if it has no real-time packets to send. The MPC and all other group members will delete the empty frame sending MT from the unfinished_polling_list. After all the real-time packets have been sent or the PCF reserved time has expired, the MPC will send the "CF_end signal" to finish the PCF period. The MTs around the MPC will call the CF_end_receipt_from() routine (see Appendix B) to set the NAV to "0" and unlock the data-type packet's sending with the DCF.

All the functions and packets' formats in MPC MAC are defined in IEEE 802.11. Our MPC creation algorithm goes through the layer above the MAC. The objective and challenge to design our MPC MAC protocol is to make it follow all the rules of IEEE 802.11. However there are still some differences with the IEEE 802.11 standard. First, in the IEEE 802.11, the MTs only have the function of being polled by a Point Coordinator (PC), and never work as a PC. However, in our MPC MAC, an MT also needs the related functions to work as MPC to initialize the PCF period and poll other MTs to send real-time packets. The MT will have extra functions to replace the original MPC when it shuts down in the PCF period (The PC in the IEEE 802.11 never shuts down). The MPC MAC uses the neighboring MPC's beacon record to estimate the SDN after the PIFS, which always has a non-zero value. However, in IEEE 802.11, the PC always uses PIFS without an SDN to compete for the media because the neighboring PCs could coordinate through a wired connection. There is an MPC range inside the communication range in our MPC MAC. The MT only tries to register to the MPC that is located in its MPC range and tries to disjoin from the MPC if the MPC moves out of the MPC range. However, in IEEE 802.11, there is only one communication range, the MT can register to any PCs that it can contact.

3.2.4. MPC-MAC illustrated

This section explains the operation of the MPC-MAC protocol by means of illustrations. Each of the following examples represents a scenario, and a set of corresponding actions performed by the MAC layer.

In Figure 3.16, When MPC "A" is about to initiate the PCF; it finds out that it is the only MPC contending for the medium. Therefore, when the value in its NAV reaches 0, it senses the medium for a PIFS period, and sends a "beacon signal" afterwards. All the nodes in its communication range will set their corresponding NAVs to the value of the PCF duration. "A" then begins to poll the MTs registered to it in the following order: "a, b, c, d, e, and f". "A" sends a poll token or a real-time packet piggybacked with a poll token to "a". Node "a" then searches its data buffer and finds real-time packets, and sends a real-time packet back to "A". MPC "A" and all the group members assume that this node may have more real-time packets to send, and still keep it in their unfinished_polling_list. MPC "A" then sends a poll token to "b", which has no real-time data and just sends an empty frame. "A" and all group members will now delete this node from their unfinished_polling_list. The MPC will continue polling the other MTs until there are no nodes in the unfinished_polling_list, or until the reserved PCF duration is over.

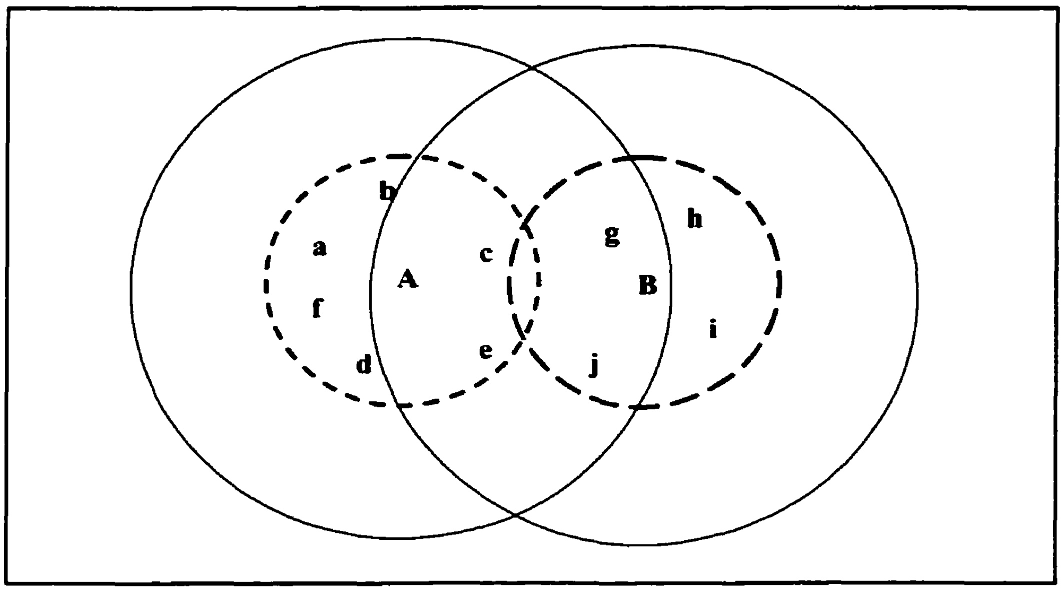


Figure 3.16 MPC shut down

If the MPC is shut down during this polling process, for instance after polling "c", then every group member having maintained the unfinished_polling_list (that is, "a", "c", "d", "e", and "f") can become the temporary MPC. When node "c' finishes sending its packet, node "d" finds that the MPC did not poll it on time, and will become the MPC, and begins polling "e, f, a, c" and itself. Normal MPC selection can resume before the next PCF period is initiated.

Consider Figure 3.16 above. If MT "h" wants to send a large data packet, it first sends an RTS to MT "g". When MPC "A" is in the PCF period, no matter whether "g" is registered to "A" or not, MT "g" will not send back a CTS (because the medium is already reserved) even if it had successfully received the RTS from MT "h". This is because its NAV was set to the value contained in the beacon signal of MPC "A" when it announced the beginning of the PCF. In this case, MT "h" may try to send the data packet or RTS to MT "g" several times, but will not receive the CTS. MT "h" will quit trying to send and conclude that MT "g" was shut down or is out of the communication range, and so it will delete "g" from its list of MTs. The situation is rectified after "h" receives the next "hello" message of MT "g". It then adds "g" again to its list. If the sending MT regards the destination MT as shut down or out of the communication range, even it had tried sending several times, it will not increase the value of the Collision Window (CW). Otherwise, the sending MT will consider that the failure was caused by collision, and hence double the value of its CW. It should be noted that small packets could be sent directly. The receiver can respond with an ACK (even if the medium is reserved), thus there will be no problem.

As Figure 3.17 shows, MT "d" was initially registered to MPC "A" at location d1. When "d" moves from d1 to d2, it finds out that MPC "B" is more suitable to be its MPC, and sends a "merge request" message to "B", and "B" sends back a "merge accept" message to "d". If "A" gains medium control to go into the PCF period before "d" sends the "disjoin" message to "A", then, node "A" will still regard "d" as one of its registered MTs, and polls it regularly till node "d" has a chance to send the "disjoin" message in the next DCF period.

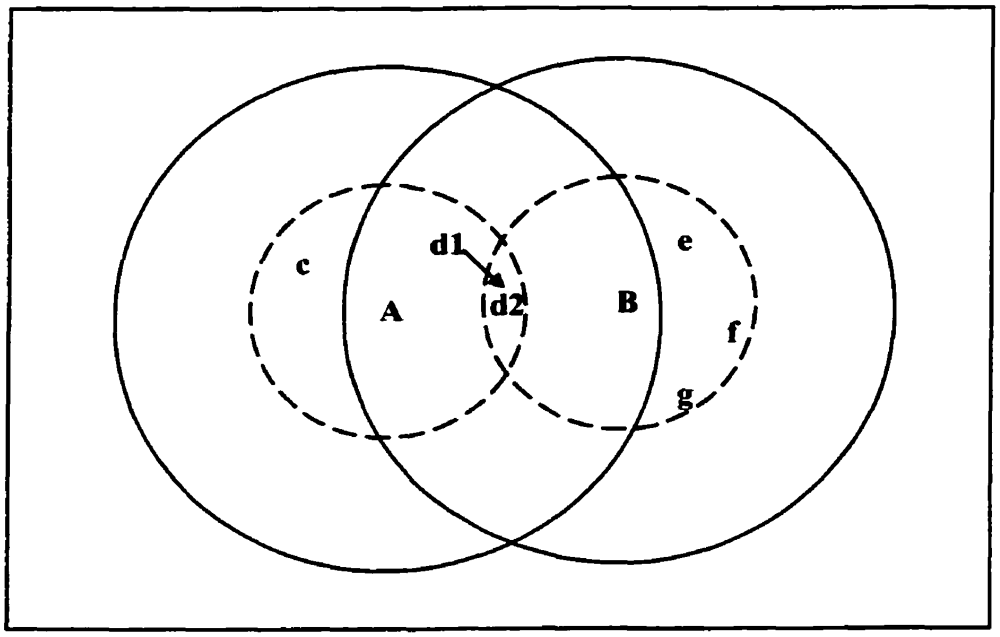


Figure 3.17 Duplicate MPC problem

3.3. Summary

This chapter provided an in-depth explanation of the operation of our proposed MPC infrastructure-creation protocol, and the operation of our proposed MAC protocol for wireless ad-hoc networks. Illustrative examples of MPC-based infrastructure-creation and MPC-based MAC were also presented.

We presented an overview of the routines that will be called on when an MT exchanges "hello" messages, "merge request" messages, "merge response" messages, and "disjoin" messages, with neighboring MTs. Furthermore, we explained how the most current data about the network topology is gathered and maintained, and how the MPC infrastructure is established and maintained throughout the wireless ad-hoc network.

In the MAC layer, a "data sending" queue exists and is used to buffer the incoming data packets that need to be relayed, or the packets created by the application(s) running on the MT and need to be sent. The MAC-layer "control frame" stack is used to buffer the high-priority control frames, such as ACK, and CTS.

CHAPTER 4

PERFORMANCE EVALUATION

In this chapter, we study the performance of our MPC wireless MAC protocol. The performance of the IEEE 802.11 in an ad-hoc wireless environment, where only DCF can operate, was examined and tested, and compared to the performance of our proposed MAC protocol. In our protocol, the DCF and the PCF (as defined in the IEEE 802.11) can co-exist. A packet-level simulator was developed using the Java programming language in order to monitor, observe and measure the performance of our protocol, using different input parameters. There are no other independently published results to compare with the results provided in this chapter.

4.1. Simulation model

4.1.1. Experimental setting

In our simulation experiments, the channel capacity is set to 2 Mbps. All the MTs are assumed to be within a 400 × 400 unit grid. The three IFS periods, SIFS, PIFS, DIFS and the slot defer time have been set to an effective length of 2, 3, 14 and 1 octet, respectively. There is a gap between the PIFS and the DIFS, which enables several neighboring MPCs to compete for the medium without interference with the DCF nodes.

To avoid collision between MPCs, several neighboring MPCs may wait for the PIFS plus a calculated number of defer slots provided that the total does not exceed 14 octet periods (equal to DIFS). (Note: Here, to improve the performance of PCF, we have slightly modified the IEEE 802.11, which defines the difference between PIFS and DIFS as one time slot.) The size of the Collision Window (CW) is between 8 and 128. When the MT suffers repeated collisions, the CW may reach 128. When an MT is successful for 4 consecutive times, the CW will be halved till it reaches 8. In our simulations, when packet transmission is attempted 3 times without receiving an ACK, the packet will be discarded. If the MT discards 3 packets in a row for the same destination, the destination MT is removed from the "MT list". The PCF period is periodic with the super-frame being equal to 1 second. "Hello" messages are exchanged every 0.2 seconds. If the MT does not hear from its neighboring MT for 2 seconds, it will remove it from its "MT list".

Each experiment tests the behavior of the system for a given number of nodes, N, for 60 seconds. Our simulations consist of two stages: the network initiation stage, and the testing stage. During the network initiation stage, the MTs are created one at a time. When a MT is generated, it tries to establish neighborhood and MPC associations with the existing MTs. After all specified MTs have been created for 5 seconds, the simulation then goes into the testing stage where data is collected for an additional 60 seconds.

4.1.2. Mobility model

In the simulation, every MT is in one of two mobility states: moving or pausing. When in the moving state, the MT moves towards its target location determined in the last pause period, with a specific speed (randomly generated with a mean equal to Move Speed (S)). When the MT reaches its target location, it will reach its pausing state. The length of the pausing period is also randomly generated with a mean equal to a user defined input parameter, Pause Time (P). During the pause period, it will determine the next target location and its moving speed. Hence, the MT's lifetime will consist of moving periods (each with its own speed and direction), and pausing periods (of different time intervals) in turn. The creation of S, P and the next target location is by Eq. (4-1). Here, the function "random (x)" creates a value from a uniform distribution (from 0 to 1). Another variable parameter related to the mobility model is the transmission range, R, which is discussed later (Section 4.2.2). The MPC range is always set to half of the transmission range.

$$\text{Value} = \text{lowest bound} + 2*\text{mean} * \text{random}(x) \qquad 4\text{-}1$$

4.1.3. Traffic model

There are a number of packet types: MAC control packets (ACK, RTS, CTS, etc.), application/data packets (real-time and non-real-time), and MPC-related packets (hello, merge request, merge response and disjoin). The length of the control packets and MPC-related packets is set to 20 and 80 octets, respectively. The length of the data packets is determined by the parameter Packet Length, L.

When an MT receives a data packet addressed to it, it creates and sends an ACK to confirm receiving the packet. When the MT plans to send a data packet whose length is over RTSThreshold, it will create and send an RTS packet first to reserve the medium and trigger the destination MT to respond back with a CTS packet. In our simulation experiments, RTSThreshold is set to a value such that the RTS-CTS exchange is used at all times.

The creation of data packets is determined by three parameters: Mean Rate (MR), Peak Rate (PR) and Burst Length (BL). MR is used to describe the value of the average packet creation rate during this MT's whole lifetime. Real-time traffic is not distributed evenly; and usually comes in bursts, as shown in Figure 4.1. Data creation is concentrated in certain ON periods (a-b, c-d and e-f) with intervening idle/OFF periods (b-c and d-e). The

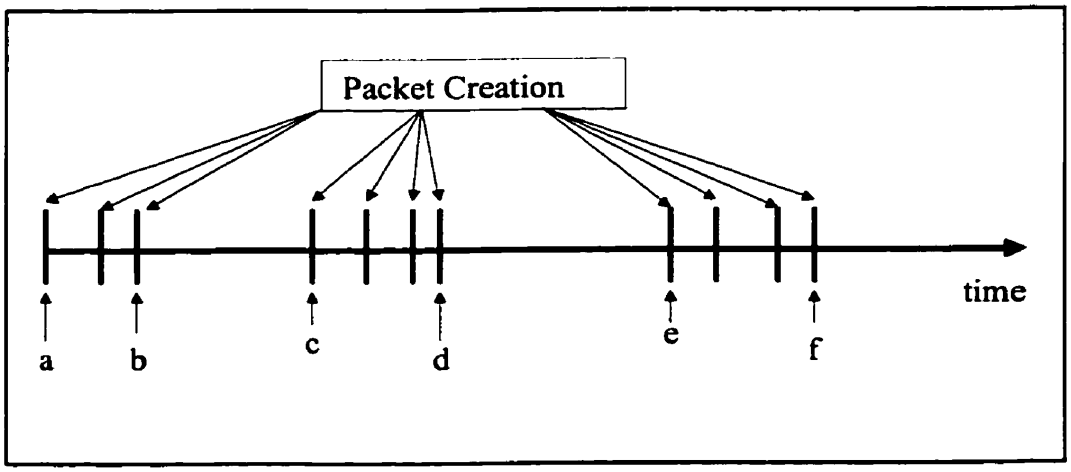


Figure 4.1 Packet creation model

"BL" is used to describe the average length of the ON period. The packet generation rate during the ON period is given by PR, which is higher than the MR. In order to simplify our experiments, we always set PR to 2 × MR, and BL was set to 0.25 seconds. The user can define the Packet Rate simulation parameter, λ = MR. When creating a data packet, an MT will choose a destination that it believes to be within its communication range. After the data packet is created, it will be pushed into its "data sending" queue. The "data sending" queue maximum size is 20 packets. If the queue becomes full, no new packets are created. In the ON period, packet inter-arrivals follow an exponential distribution function. We use Eq. (4-2) to calculate the next arrival time of packets. (Here, λ is the packets' arrival rate and x is a random number created between 0 and 1.

$$\text{Next arrival time} = \text{current time} - \tfrac{1}{\lambda} \bullet \log(X) \qquad 4\text{-}2$$

Table 4.1 provides all the simulation parameters and their nominal values. (Note: All parameters except λ, R, P, S, N and L are constant. Most parameters' value settings come from the IEEE 802.11 standard documents [31-35]. Other parameters like DataSendingQueueSize, Grid size, reportInterval_time, etc. where assumed to the realistic values.)

Parameter	Meaning	Nominal value
dataRate	The transmission rate	2 Mbps
PR_r (2λ)	Peak Rate of real-time traffic	10 packets/sec
MR_r (λ)	Mean Rate of real-time traffic	5 packets/sec
BL_r	Burst Length of real-time traffic	0.25 sec
PR_c (2λ)	Peak Rate of non-real-time traffic.	10 packets/sec
MR_c (λ)	Mean Rate of non-real-time traffic	5 packets/sec
BL_c	Burst Length of non-real-time traffic	0.25 sec
MPC_radius (R/2)	Inner communication range used to establish the MPC infrastructure	25.0 units
communication_radius (R)	Real communication cell range.	50.0 units
Grid size	Simulation grid size	400 × 400 unit
avg_movingSpeed	Average moving speed of the nodes for every moving period	10 units/sec
avg_pauseTime (P)	Average length of the node's pause period. This is the same as avg_movingSpeed. The method by which to create a value for a specific pause period is depends on moveModel	4 sec
DataSendingQueueSize	The size of the node's MAC queue	20 packets
NodeNum (N)	The number of nodes in the simulation. Nodes are created one by one	40 nodes
Packet_L_min (L)	The minimum length of real-time and non-real-time packets	1000 octets
Packet_L_max (L)	The maximum length of real-time and non-real-time packets. We choose the same value as the minimum packet length	1000 octets
RTSThreshold	If the data packet size is larger than this value, it will trigger the virtual sense mechanism	400 octets
withPCF	This is the parameter used to control the type of network we want to simulate. It can have one of two possible value: True – wireless ad-hoc network False – wireless ad-hoc network with MPC creation	True or False
CW_minimun	The minimum Collision Window (CW) size. The CW is a parameter used to create the SDN.	8
CW_maximum	The maximum CW value	128
CW_decrease_threshold	The required number of consecutive successful transmissions to reach CW_minimum from CW_maximum	4
PacketL_hello	Size of the "hello" message	80 octets
PacketL_mergeRequest	Size of the "merge request" message	80 octets
PacketL_mergeRespond	Size of the "merge response" message	80 octets
PacketL_disjoin	Size of the "disjoin" message	80 octets
PacketL_function	Size of the control frames, like ACK, RTS, CTS, and beacon	20 octets
SlotUnitLength	The effective length in bytes of one slot defer period	1 octet
SIFSLength	The effective length in bytes of the SIFS	2 octets
PIFSLength	The effective length in bytes of the PIFS	3 octets
DIFSLength	The effective length in bytes of the DIFS	14 octets
MPC_replace_IFS	The effective length in bytes of the IFS for the doubting MT competing to replace the original MPC	1000 octets
PollRespondExpired	The effective length in bytes of the time duration during which a polled MT is expected to respond to its MPC	6 octets

WaitingExpired	The ACK timeout period, starting from the time the corresponding data packet had been transmitted	2000 octets
ExistDoubtCredit	The maximum number of transmission failures after which a destination is to be removed from the node's "MT list"	3
ShortRetryLimit	Maximum number of time to attempt transmitting a packet whose size is smaller than RTSThreshold	3
LongRetryLimit	Maximum number of time to attempt transmitting a packet whose size is larger than RTSThreshold	3
poll_tryLimit	The maximum number of times to poll an MT after which it will be deleted from "MT list" in case no response has been received	3
SuperFrameLength	Length of the super frame period	1 sec
max_PCF_duration	The maximum length of the PCF period that an MPC can reserve	0.5 sec
HelloIntervalLength	The interval between "hello" messages	0.2sec
ExistSuspiciousLimit	The neighboring MT whose "hello" message is not heard for this value will be deleted from the "MT list".	2 sec
reportInterval_time	The length of the batch interval	10 sec
collectInterval_num	Number of batch intervals in one simulation experiment	6
InitialWaitLength	When a node is turned on, it remains silent for this period to collect information.	0.4 sec
nodeAvgCreateInteval	This value defines the time interval between nodes creation in the simulation initiation stage	1 sec
simulation_time	The length of the entire simulation time	200 sec
node_avg_duration	The average lifetime of a node	100 sec

Table 4.1 The simulation parameters

4.1.4. Performance metrics

The following are the performance metrics of interest in this thesis: (1) the transmission delay, which is measured from the time when the packet is created until the time the MT receives an ACK from the destination MT of this sent packet. We calculate the **Average Delay (AD)** of all the successfully received packets. (2) The **Discard Ratio (DR),** which is the ratio of the number of discarded packets to the total number of packets sent. There are several reasons for an MT to discard a packet. For example, before sending the packet, the MT checks the packet's destination to see whether it is reachable, if the packet's destination is not in the MT's reachable neighborhood MT list, the MT discards the packet without trying to send it. This kind of discarding does not occupy any medium resources. The packet may also be discarded because sending failed 3 consecutive times. When we calculate DR, we count the packets that are sent but failed and are finally discarded relative to the total number of packets that are sent (both those that are successful and those that fail). (3) The **Compensation Rate (CR)**, which is a measure of the performance loss of non-real data to the performance gain of real time data. With the help of the PCF, the average delay of real-time packets always decreases, but at the same time, the average delay of non-real-time packets often increases. CR is given by Eq. (4-3) below. A large value for CR means that the PCF brings in a greater negative effect on non-real-time packet's AD than the positive effect on AD of real-time packets. Ideally, CR should be less than 1. CR can be used to represent the benefit of PCF; a smaller CR leads to a better PCF performance.

$$CR = \frac{AD\ of\ non\ real\ time\ with\ PCF - AD\ of\ non\ real\ time\ without\ PCF}{AD\ of\ real\ time\ without\ PCF - AD\ of\ real\ time\ with\ PCF} \quad (4\text{-}3)$$

4.2. Discussion of the results

A number of simulation experiments were conducted in order to study the effects of Packet Arrival Rate (λ), Radio Transmission Range (R), MT Mobility (including Moving Speed – S, and Pause Time – P), Number of Nodes (N), and Data Packet's Length (L). The results are shown in Figures 4.2–4.7. In these Figures, Column-I shows the Average Delay of real-time packets and non-real-time packets with and without the PCF, and

Column-II shows the Discard Ratio of real-time and non-real-time packets with and without the PCF.

4.2.1. The effect of packet arrival rate

The results of the experiments in this section show the effect of changing the packet arrival rate (λ), and are shown in Figure 4.2. In all the experiments, with the increase of λ, both the Average Delay (AD) and the Discard Ratio (DR) increase. Comparing the AD of the experiments with the PCF to the AD without the PCF, we found that using the PCF benefits the MAC performance by decreasing the AD and DR of real-time packets in all cases. In Figure 4.2A-I, with the PCF, the AD of real-time packets decreased by 11% but that of non-real-time packets increased by 10% when λ is 1 packet/sec (CR=0.93). When λ increases, the PCF gives more benefit to real-time packets than the negative effect on non-real-time packets. When λ is 3.5 packet/sec, the AD of real-time packets decreased by 20%, while the AD of non-real-time packets only increased by 14% (CR= 0.66).

From Figure 4.2A-II, we see the advantage of the PCF, which decreases the DR of both real-time packets and non-real-time packets. When λ is 1.5 packets/sec, the DR of real-time packets decreased by 43% and at the same time, the DR of non-real-time packets also decreased by about 5%. The reason is as follows: when we lower the DR of real-time packets this implies that collisions of real-time packets decrease, and this would ease the traffic load on the medium. Then during the DCF period, less MTs take part in the competition for the medium, and that leads to lower collision of non-real-time packets, and thus the DR of non-real-time packets may also decrease.

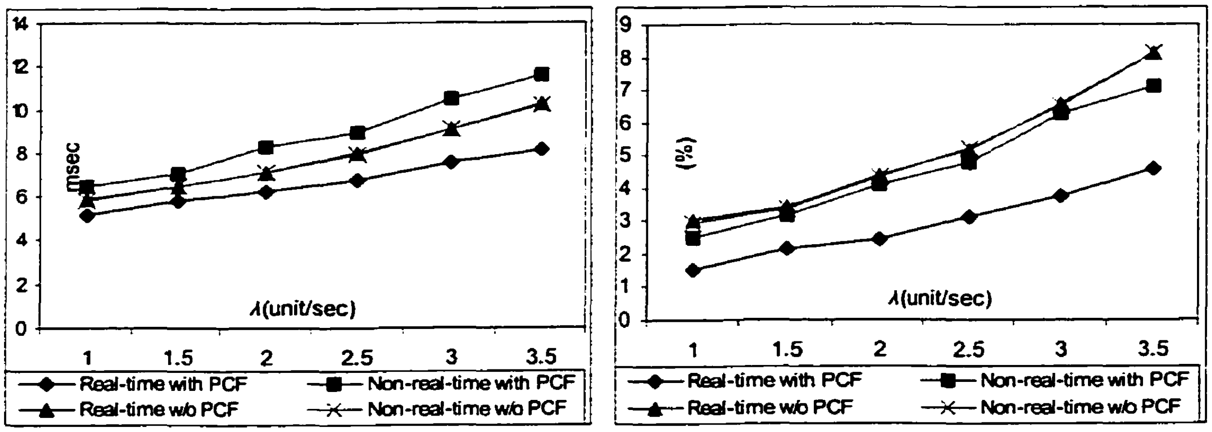
When we increased the MT's radio transmission range (R) to 200 units, this increases the density of the MTs per cell. The results of the experiment are shown in Figure 4.2B. When compared to Figure 4.2A, we can find that AD and DR both increase more sharply with the increase of λ. When R is 200 and λ is increased from 1 to 3.5 packets/sec, AD for real-time packets with the PCF increased by about 500% as opposed to only 57.6%, when R is 100. On the other hand, DR for real-time packets with the PCF increased by 870% using the same settings for R and λ, as opposed to only 200%, when R was 100 units.

When we increased the MT's moving speed from 1 unit/sec to 11 units/sec, the use of the PCF resulted in a larger performance loss for non-real-time packets than the performance gain for real-time packets. Comparing Figure 4.2C-I to Figure 4.2A-I, for example, when S is 11 and λ is 3.5 packets/sec, with the PCF the AD of real-time packets is decreased by 6.7% but that of non-real-time packets increases by 30.5%. Comparing Figure 4.2C-II to Figure 4.2A-II, we find that the performance gain for real–time traffic with the PCF is more apparent. We also note that the rate of increase of DR with increasing λ is much lower with higher mobility. When we increased the MT's pause time from 4 to 24 sec, the benefit to real-time packets with the PCF increased and the trade-off for non-real-time packets with the PCF shrunk. Using the PCF, AD for real-time packets decreased, and increased by about 17% for non-real-time traffic, as shown in Figure 4.2D-I. This beats the 6.7% gain (for real time traffic) and the 30.5% loss (for non-real-time data), as shown in Figure 4.2C-I.

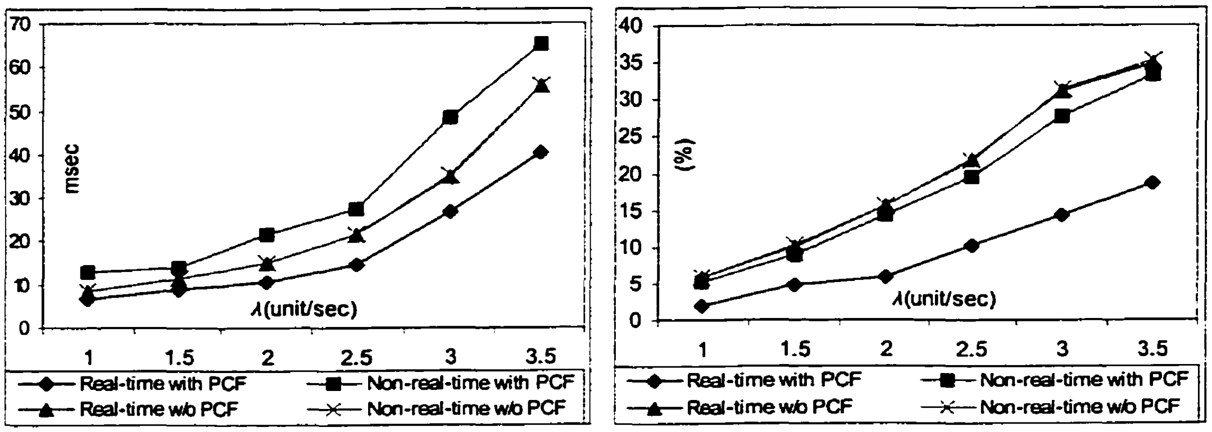
When we double the number of nodes from 40 to 80 (as shown in Figure 4.2E) the nodal density increases, a small increase in AD and DR results for small values of λ because the medium capacity is still much higher than the traffic load. However, with the increase of in λ, the increase of both AD and DR in the case of high nodal density is much sharper than in low nodal density, because the traffic load in high nodal density will increase dramatically when λ increases. For example, when λ reached 3.5 packets/sec, AD for real-time traffic using the PCF increased by 200% under high nodal density, as opposed to 60% under low nodal density, and DR for real-time with PCF also increased to twice as much as for a low nodal density network. The CR is higher in high nodal density network than low nodal density networks.

When we double the packet size (from 1000 to 2000 octets), as in Figure 5-2F, AD was also doubled when λ was small. The reason behind this is that when the λ is small, almost all the packets will be sent immediately without being delayed in the MAC queue. The MT needs more time to transmit the larger packets. When λ increases, both AD and DR increase in a sharp manner.

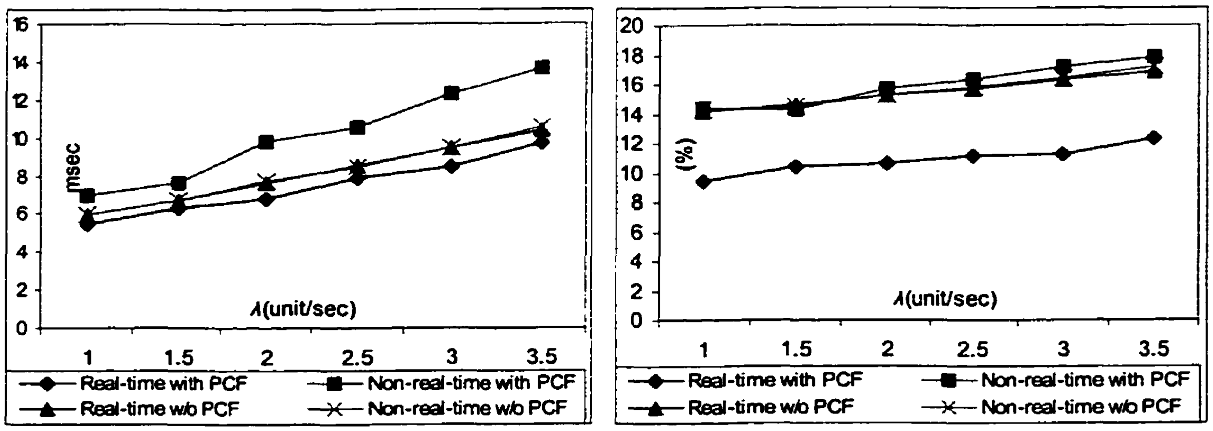
Column (I) - Average Delay Column (II) - Discard Ratio



(A) R=100, S=1, P=4, N=40, L=1000

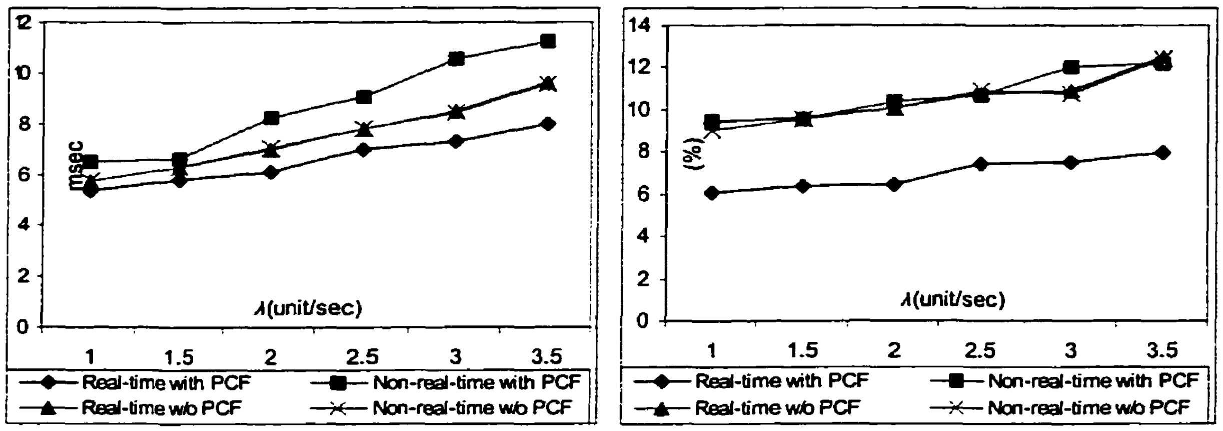


(B) R=**200**, S=1, P=4, N=40, L=1000

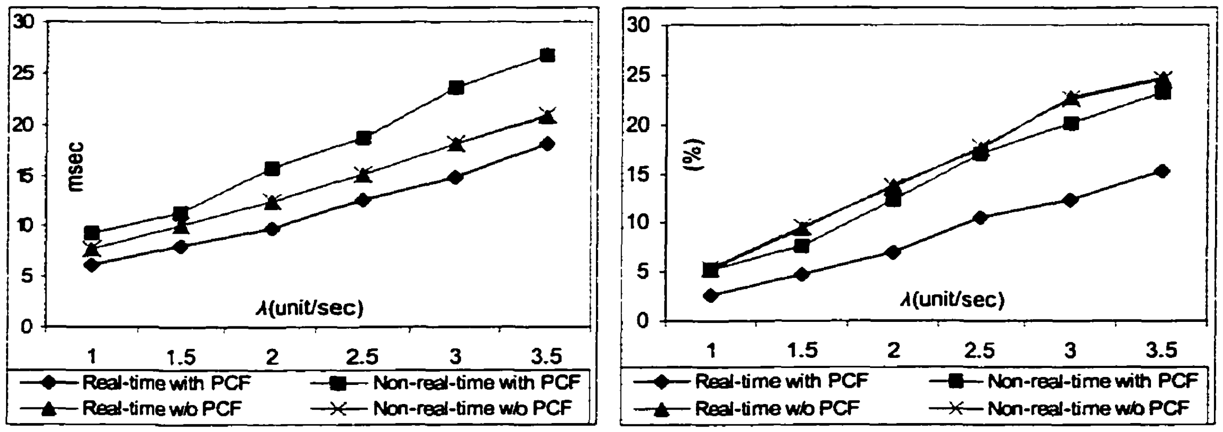


(C) R=100, S=**11**, P=4, N=40, L=1000

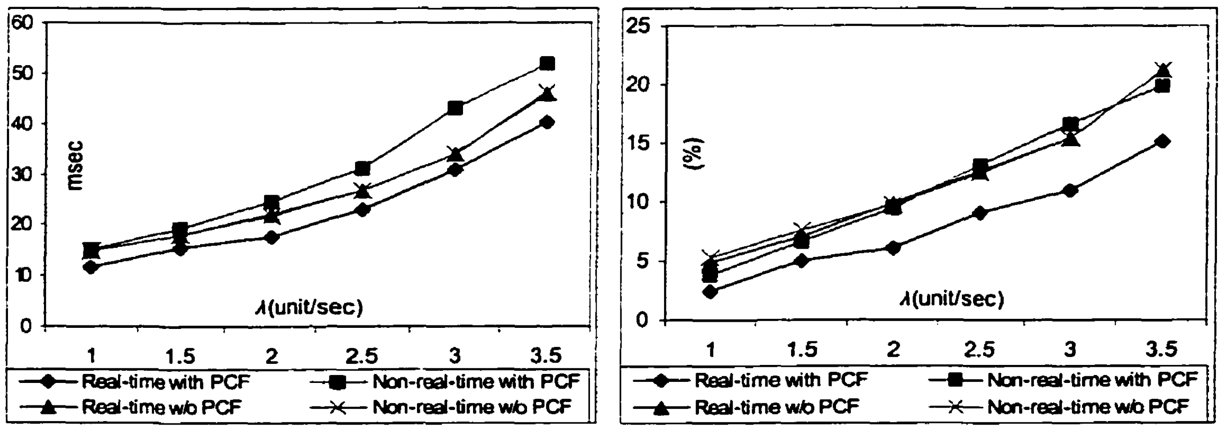
Column (I) - Average Delay Column (II) - Discard Ratio



(D) R=100, S=11, P=24, N=40, L=1000



(E) R=100, S=1, P=4, N=80, L=1000



(F) R=100, S=1, P=4, N=40, L=2000

Figure 4.2 Effect of packet arrival rate

4.2.2 The effect of the radio transmission range

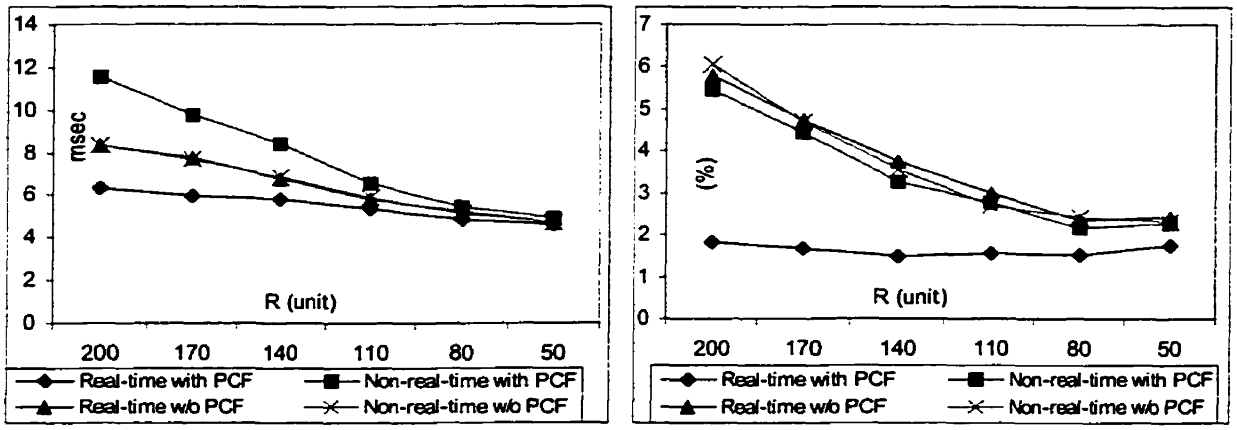
The experiments in this section study the effect of changing the radio transmission range (R) on the performance. In all the experiments, the use of the PCF benefited real-time packets by reducing AD and DR. In column I of Figure 4.3, we found that with the decrease of R, AD also decreases. If the traffic load is low enough, the MT will send the packet immediately, and the packet delay will be very small.

In Figure 4.3A, when R is large, the PCF decreased both AD and DR for real-time packets. When R was 200 units, using the PCF, AD for real-time packets decreased by 24%, and DR decreased by 69%. In Figure 4.2A-II, DR for real-time traffic using the PCF stayed very low, and with the decrease of R, there was almost no change in the value.

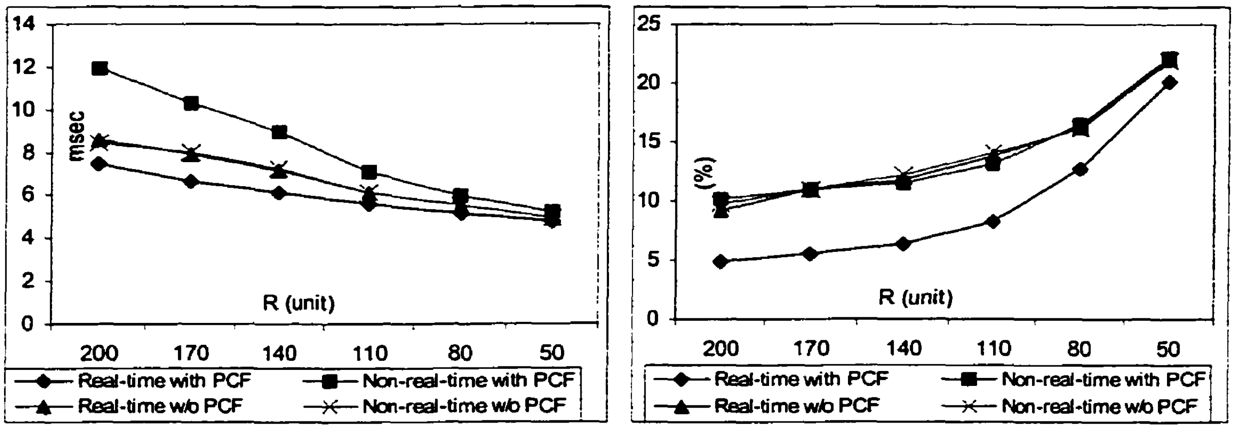
When we increased the average moving speed, from 1 to 11 units/sec, the Compensation Rate (CR) of AD increased. In Figure 4.3B-I, when R was 200, CR was 3.15 compared to 1.60 in Figure 4.3A-I. The pause time also affects CR. Comparing Figure 4.3C-I to Figure 4.3B-I, when we increased the pause time from 4 to 24 sec, CR was reduced to 2.26 from 3.15. The effect of nodal density is shown in Figure 4.3D and Figure 4.3E, which show simulation experiments for 80 nodes and 100 nodes, respectively. When R is 200 units, AD for real-time traffic with the PCF is 6.3, 13.9, and 32.3 ms when the number of nodes is 40, 80, and 100, respectively. With the decrease of R, AD for all the experiments decreases as well.

As can be concluded from column II in Figure 4.3 (A, B, C, D and E), there are two factors affecting DR. If R decreases, the number of competing MTs, and corresponding MAC collisions, also decreases. This is clearly shown in Figures 4.3 A-II, 4.3 DII, and 4.3 EII. The other factor is mobility, when R decreases, a moving MT would be more likely to move out of its communication zone. This would lead to the transmission of packets, for which no ACKs will be received. In Figure 4.3B-II and Figure 4.3C-II, the speed is much higher, and a decrease in R results in an increase in DR. The mobility factor affects the results more than the density factor does. Increasing pause time decreases the mobility factor. In Figure 4.3C-II, DR only increased by 9.3%, but it increased by 15.3% in Figure 4.3B-II, with a higher speed and a shorter pause time.

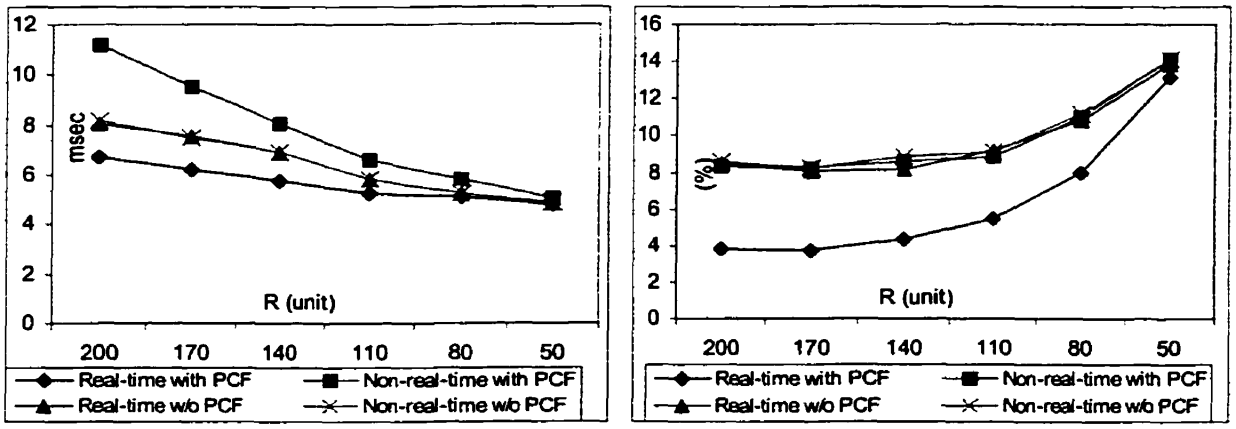
Column (I) - Average Delay Column (II) - Discard Ratio



(A) λ=1, S=1, P=4, N=40, L=1000

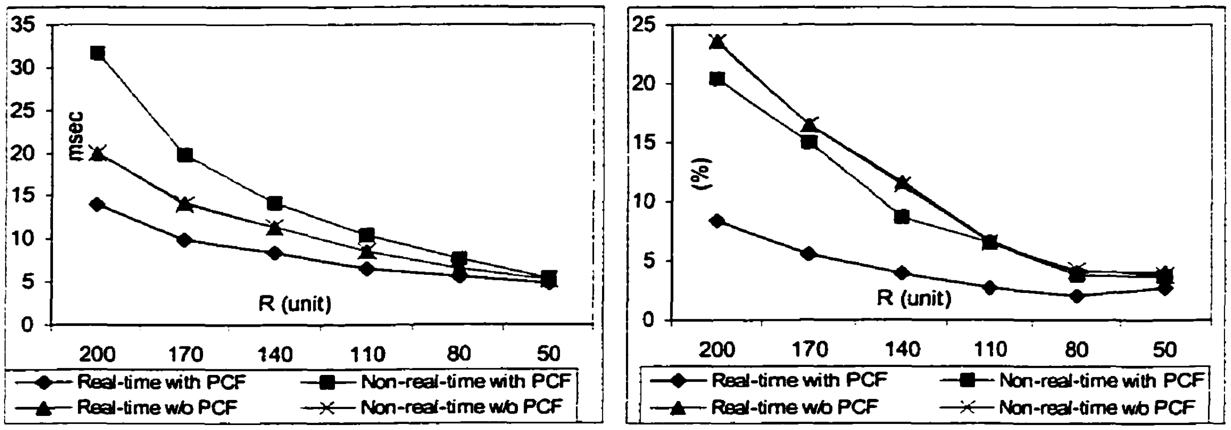


(B) λ=1, S=11, P=4, N=40, L=1000

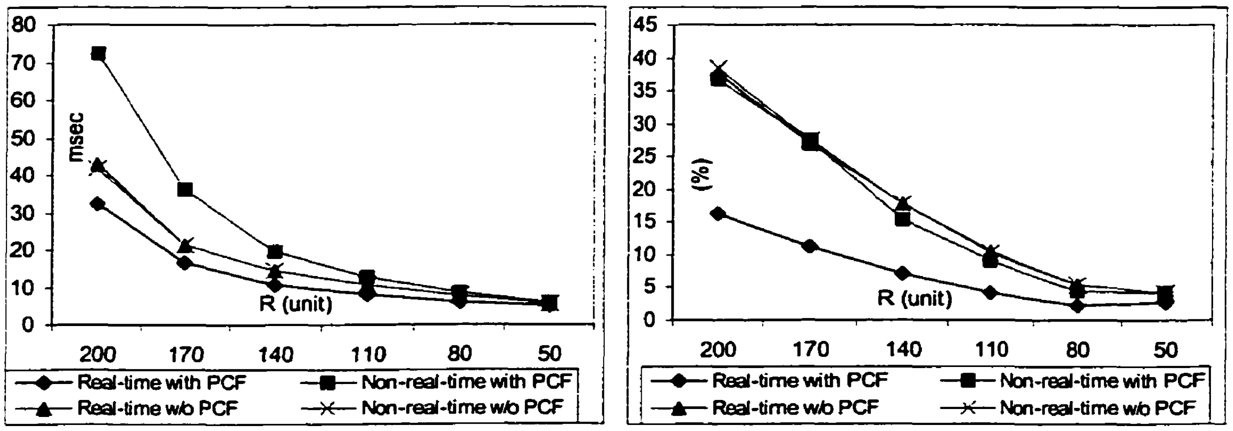


(C) λ=1, S=11, P=24, N=40, L=1000

Column (I) - Average Delay Column (II) - Discard Ratio



(D) $\lambda=1$, S=1, P=4, N=**80**, L=1000



(E) $\lambda=1$, S=1, P=4, N=**100**, L=1000

Figure 4.3 Effect of transmission range

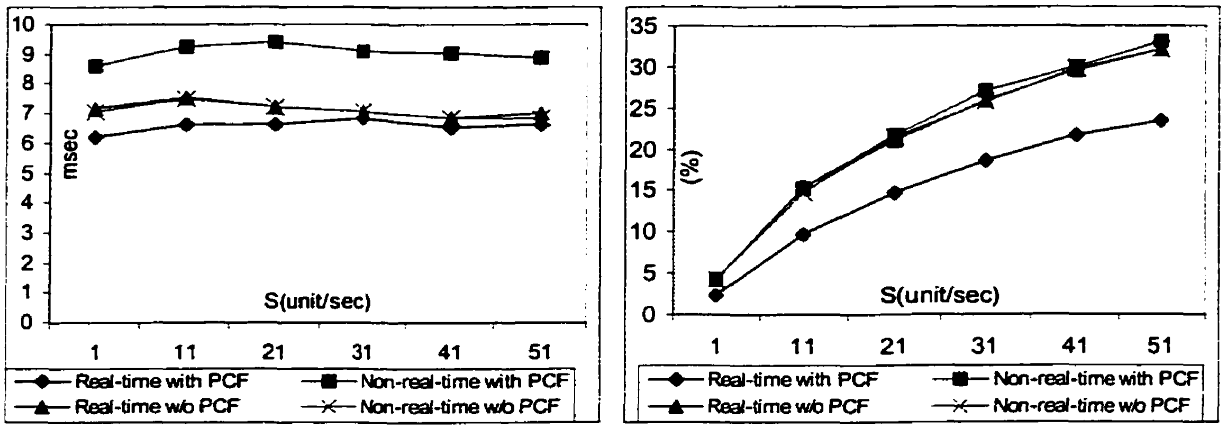
4.2.3. The effect of the mobile speed

The Experiments in this section study the effects of changing the speed (S) of MTs on the performance. In Figure 4.4, Column I, shows that the effect of mobility on the average delay is limited. However, the MT's speed had a greater effect on DR. For example, in Figure 4.4A-II, when S grew from 1 to 51 units/sec, DR for real-time packets with the PCF increased by 879%. When we doubled R from 100 units to 200 units, DR was initially at a higher level when S was 1 unit/sec, but grew less sharply with increasing S. For example, DR for real-time traffic with the PCF and R=200 was 6.6% when S was 1 unit/sec - much higher than DR in the R=100 case (2.4%). However, when S reached 51 units/sec, DR was 20% in the case of R=200 - lower than DR in the case of R=100, where it was equal to 24%. The reason is that when we increase the Radio Transmission Range, the nodal density radio coverage area increases, thus DR is increased when S is small. But, the large radio transmission range increases the ability to reduce the effect of mobility. When we double the number of nodes, the DR curve for real-time traffic with the PCF is parallel to the former, but is 5 points higher with the increase in S.

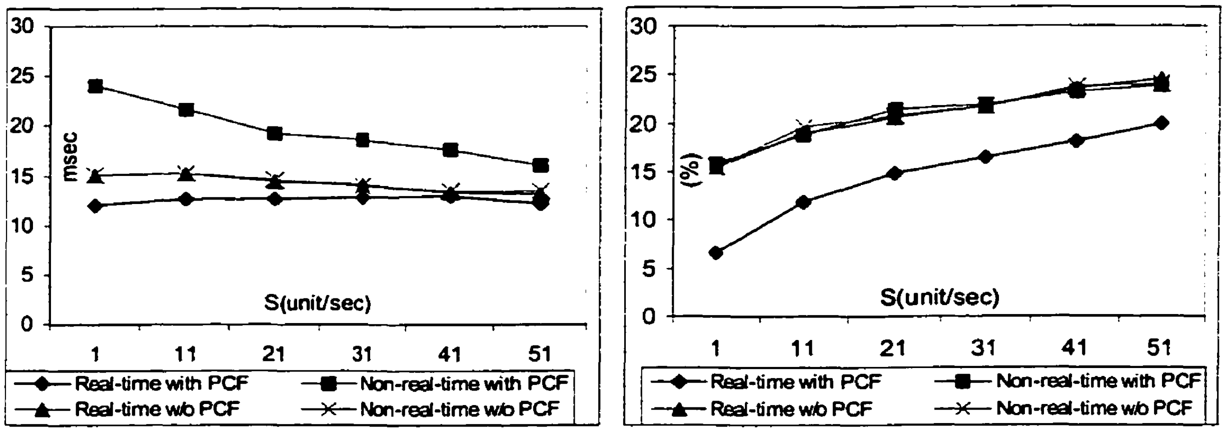
4.2.4. The effect of the pause time

The experiments in this section study the effects of changing the pause time (P) on performance. The results in Figure 4.5 show that both AD and DR are generally improved with increasing the pause time for the MTs. This is because with a higher pause time, packets can be usually delivered in fewer attempts.

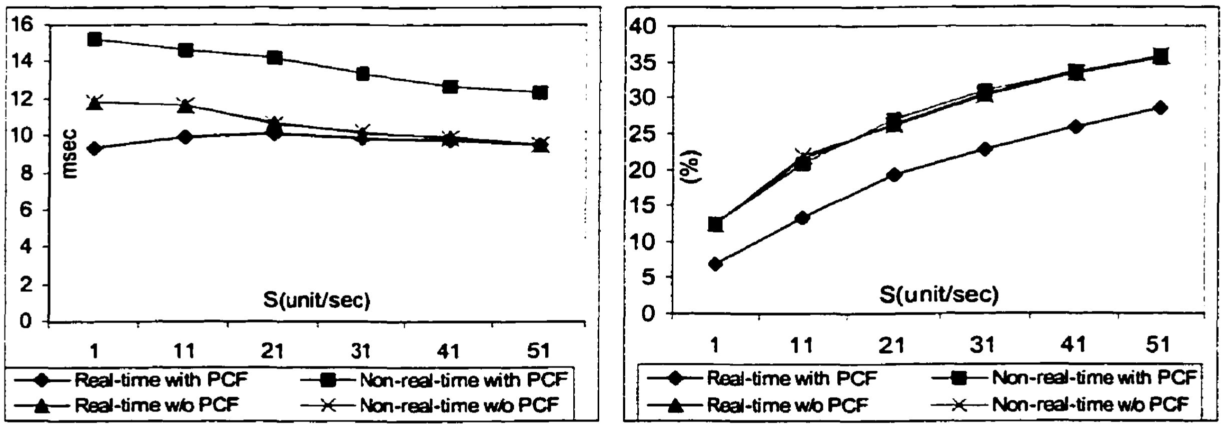
Column (I) - Average Delay Column (II) - Discard Ratio



(A) λ=2, R=100, P=4, N=40, L=1000



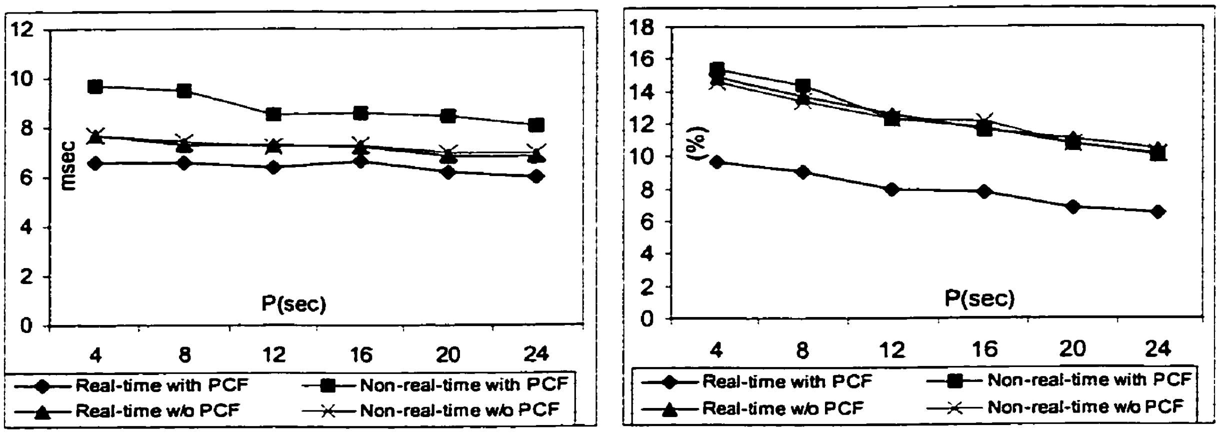
(B) λ=2, R=**200**, P=4, N=40, L=1000



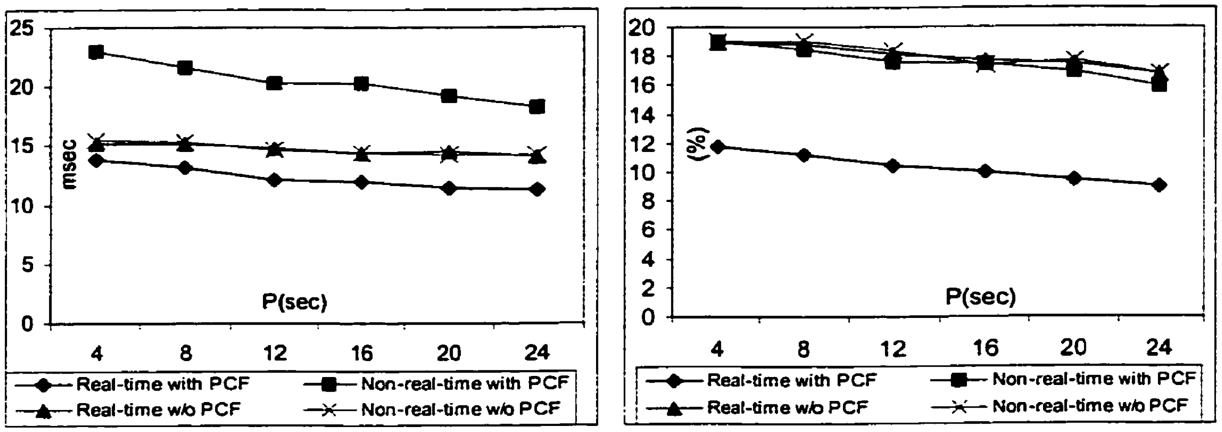
(C) λ=2, R=100, P=4, N=80, L=1000

Figure 4.4 Effect of mobile speed

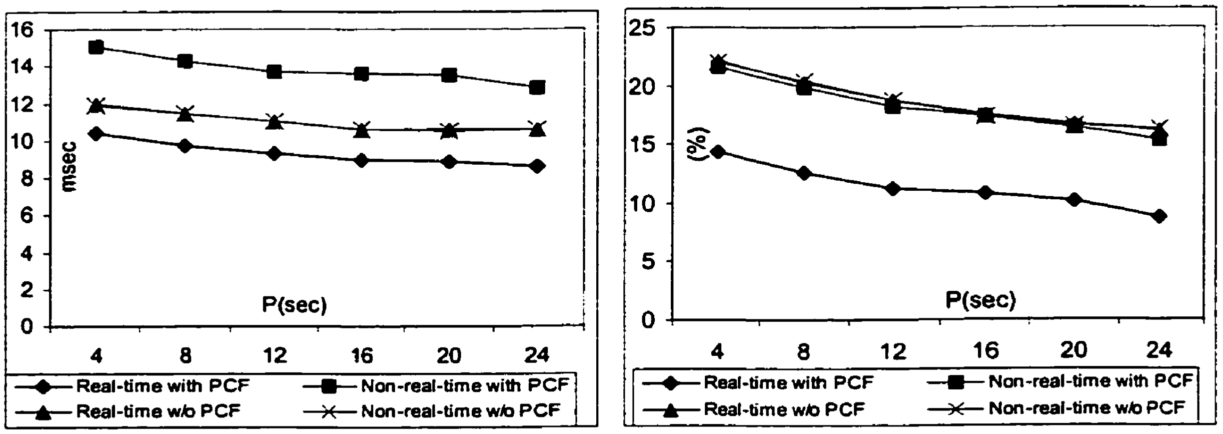
Column (I) - Average Delay Column (II) - Discard Ratio



(A) λ=2, R=100, S=11, N=40, L=1000



(B) λ=2, **R=200**, S=11, N=40, L=1000



(C) λ=2, R=100, S=11, **N=80**, L=1000

Figure 4.5 Effect of mobile pause time

4.2.5. The effect of the number of nodes

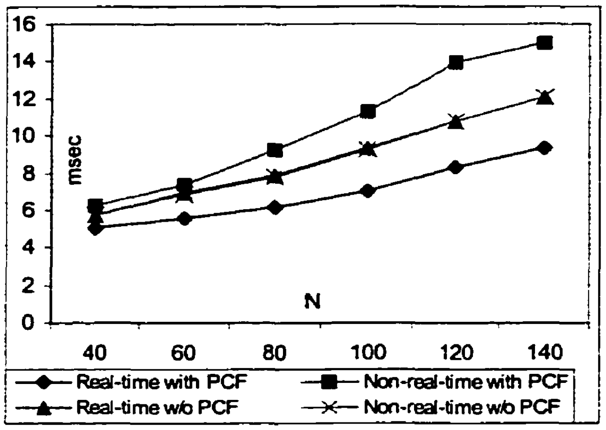
Experiments in this section study the effects of changing the number of MTs (N) in the simulation on the performance. Both AD and DR increase as the number of nodes increases. In Figure 4.6A-I, we see that with the increase in N, the performance advantage of the PCF for real-time packets is more apparent. When N was 40, with the PCF, AD for real-time packets decreased by 11.5%, as opposed to 23% for N=140. In Figure 4.6A-II, the use of the PCF decreased DR for real-time packets by at least 41%. DR for non-real-time packets also decreased.

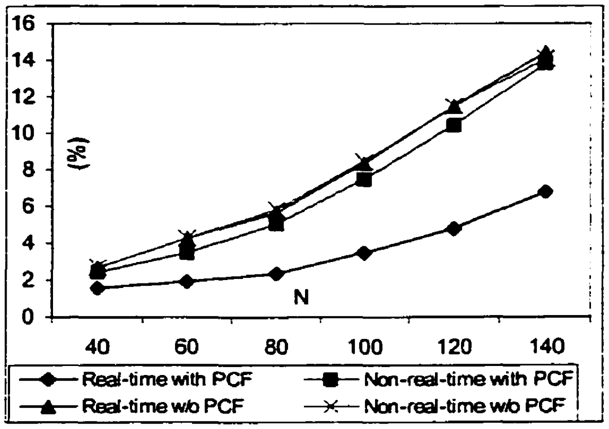
When we doubled λ from 1 packet/sec to 2 packets/sec, AD also increased, but the rate of increase is higher when N is higher. For example, when N was 40, AD for real-time traffic with the PCF increased from 5.1 msec to 6.0 msec, (a 17.6% increase). However, when N was 140, AD for real-time traffic with the PCF increased from 9.3 msec to 20.7 msec (a 123% increase). Also DR with $\lambda = 2$ packets/sec increased more sharply with the increase of N. When we doubled R from 100 to 200 units (see Figure 4.6C), it increased the density of the nodes in the each radio coverage area, thus the effect is the same as increasing λ (Figure 4.6B), but AD and DR grew more sharply than the increases of R.

When we increase the value of S to 11 units/sec (Figure 4.6D), the performance gain of using the PCF decreases. For example, when S=1 and N=40 (see Figure 4.6A-I), CR is 0.85, as opposed to 2.34 when S=11 (see Figure 4.6D-I). If the pause time is increased, the effect is a reduction of CR (see Figure 4.6E-I). The performance gain in DR is even more apparent when mobility is increased, especially for a large number of MTs.

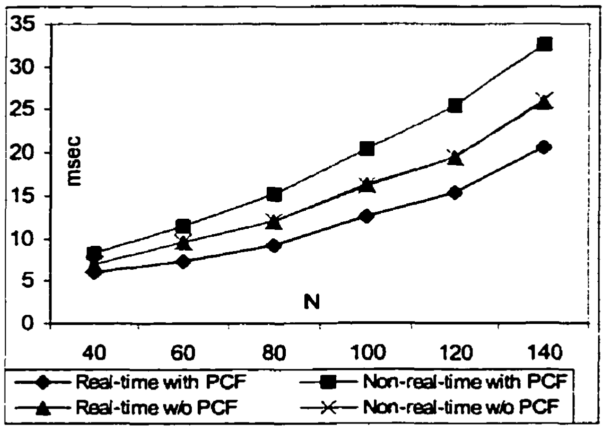
When we increased L from 1000 to 2000 octets (see Figure 4.6F), AD increased significantly. Comparing Figure 4.6F to Figure 4.6A, (with N growing from 40 to 140), AD for real-time traffic with the PCF increased by 4.3 msec and 23.8 msec, for small and large packets respectively. Also, DR for real-time traffic with the PCF increased from 5.2 (see Figure 4.6 A-II) to 17.7 (see Figure 4.6 F-II). Of more importance to us is the relative performance gain with the use of the PCF. CR for large size packets is 0.134 (when N=140 in Figure 4.6F-II), which is smaller than for small size packets (1.05 when N=140 in Figure 4.6A-II). This indicates that using the PCF brings in more benefit for real-time packets as the number of nodes and/or the packet size increase.

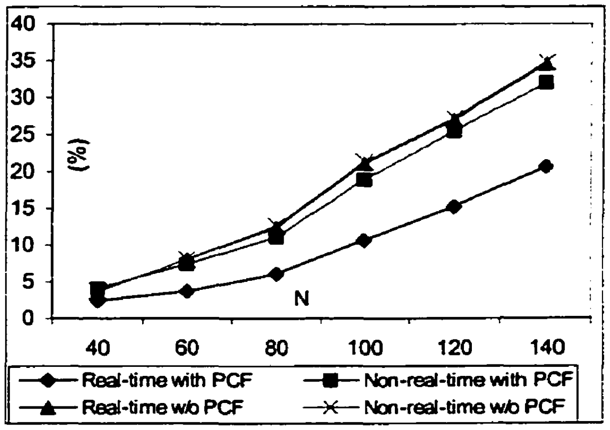
Column (I) - Average Delay Column (II) - Discard Ratio



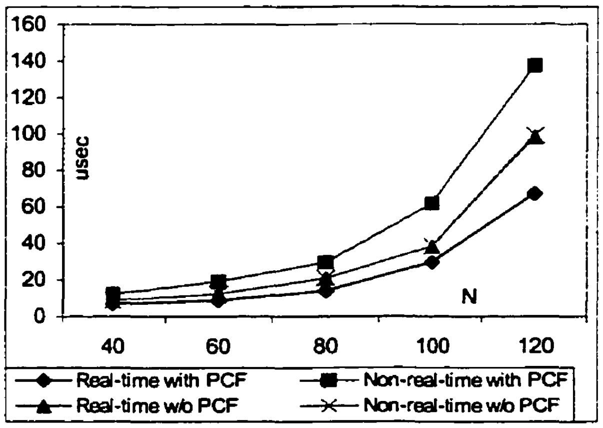


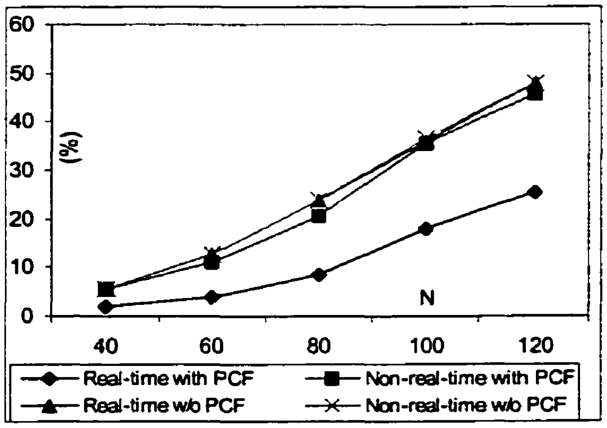
(A) λ=1, R=100, S=1, P=4, L=1000





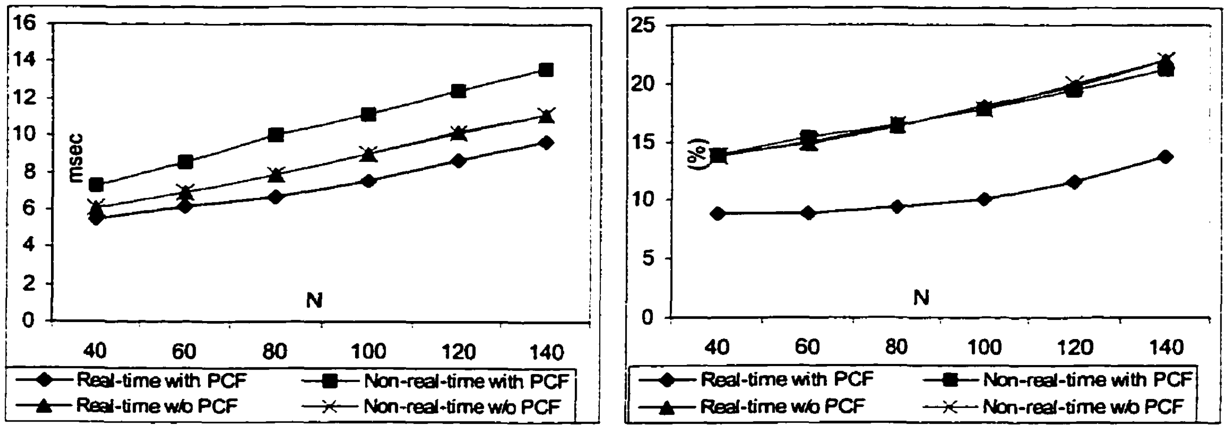
(B) λ=2, R=100, S=1, P=4, L=1000



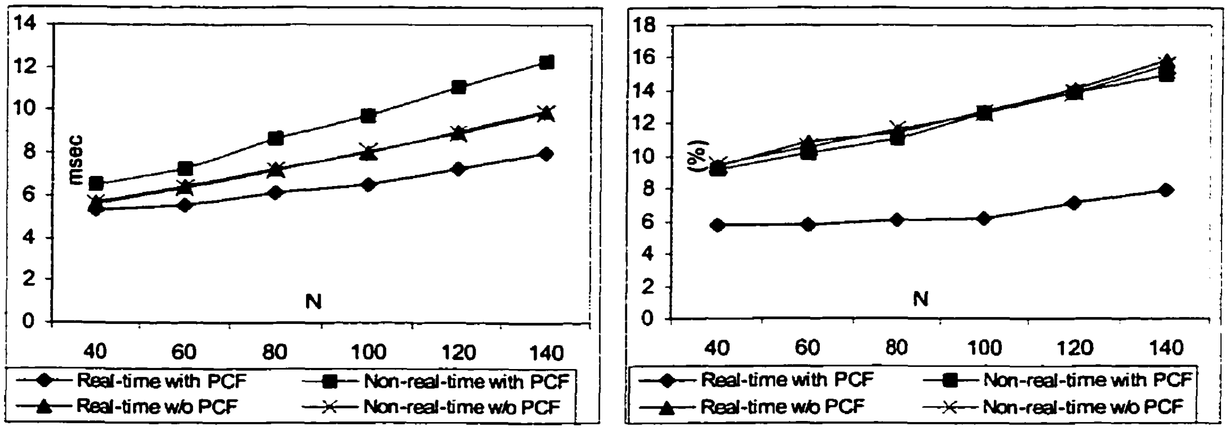


(C) λ=1, R=200, S=1, P=4, L=1000

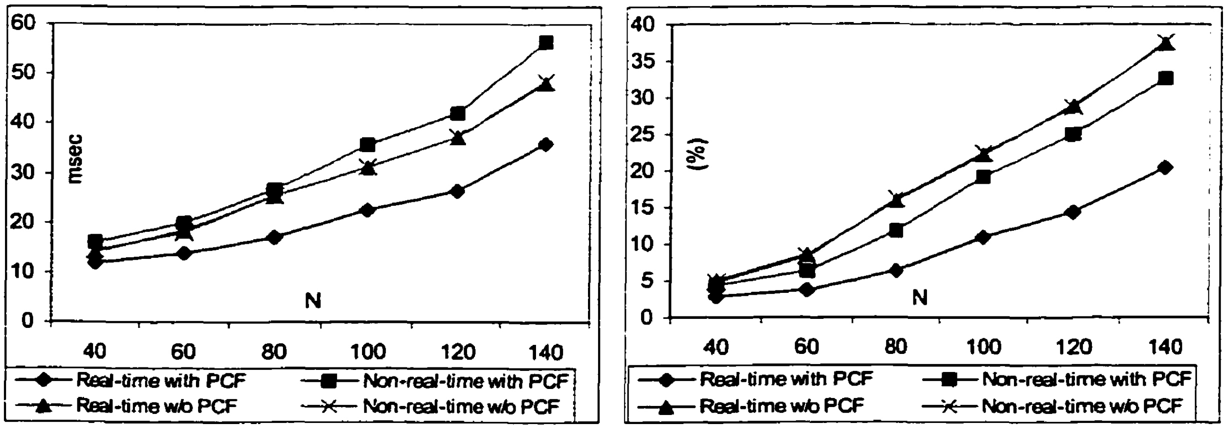
Column (I) - Average Delay Column (II) - Discard Ratio



(D) λ=1, R=100, S=11, P=4, L=1000



(E) λ=1, R=100, S=11, P=24, L=1000



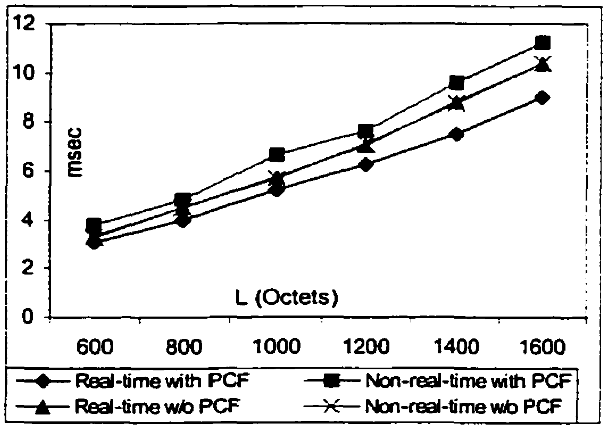
(F) λ=1, R=100, S=1, P=4, L=2000

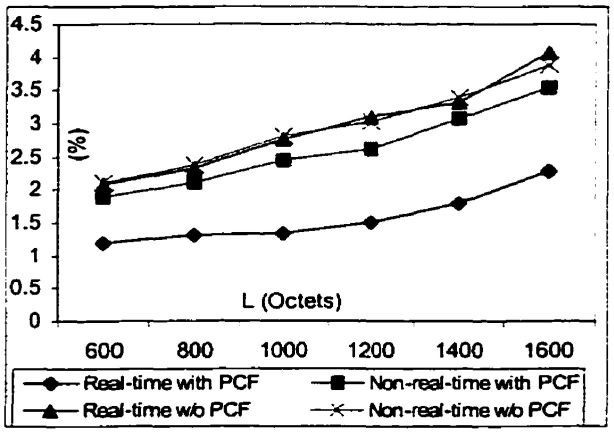
Figure 4.6 Effect of number of nodes

4.2.6. The effect of the packet length

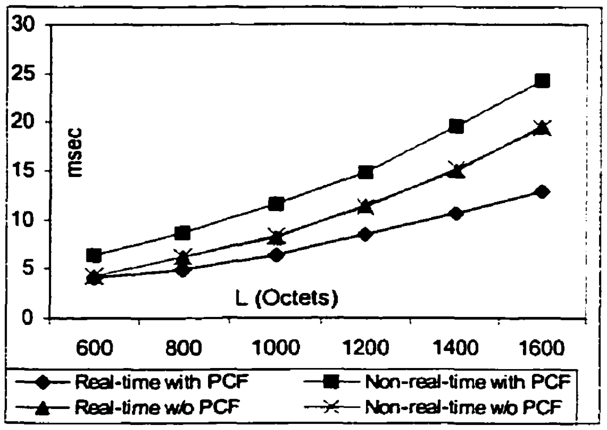
Experiments in this section study the effects of changing the packet size (L) on the performance. The results shown in Figure 4.7 indicate that both AD and DR increase as the packet size increases. However, the value of CR tends to decrease. For example, in Figure 4.7 A-I (with N=40, $\lambda=1$), when L=600, the CR was 2.00, but when the L=1600, the CR decreased to 0.62. This confirms the fact that the performance gain is more apparent for larger packet sizes.

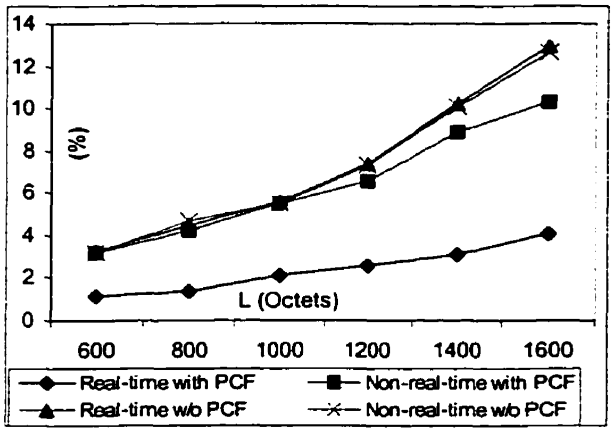
Column (I) - Average Delay Column (II) - Discard Ratio



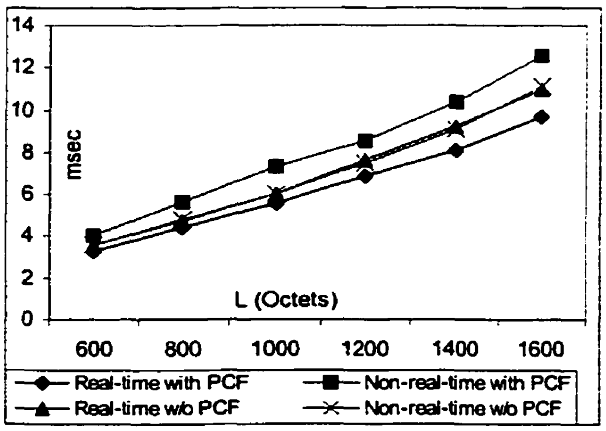


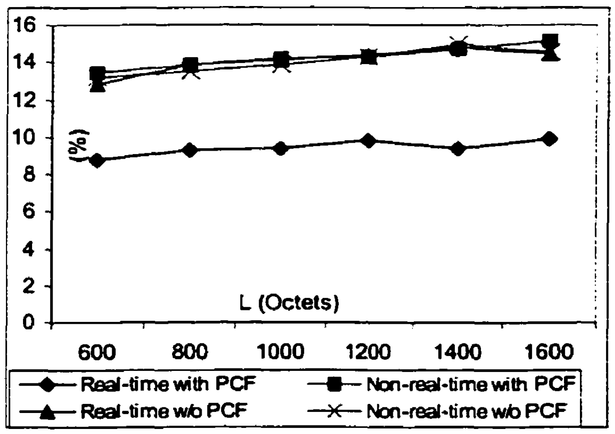
(A) λ=1, R=100, S=1, P=4, N=40





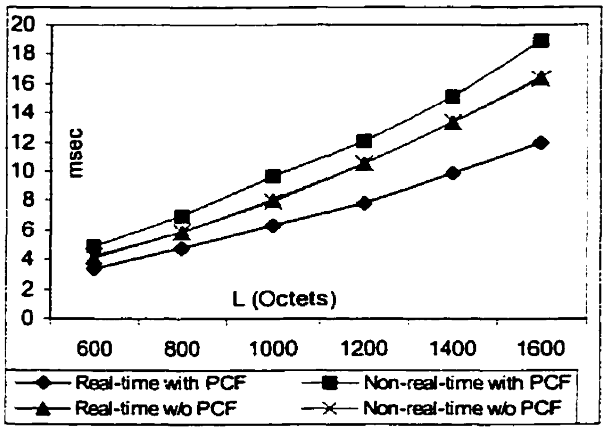
(B) λ=1, R=**200**, S=1, P=4, N=40

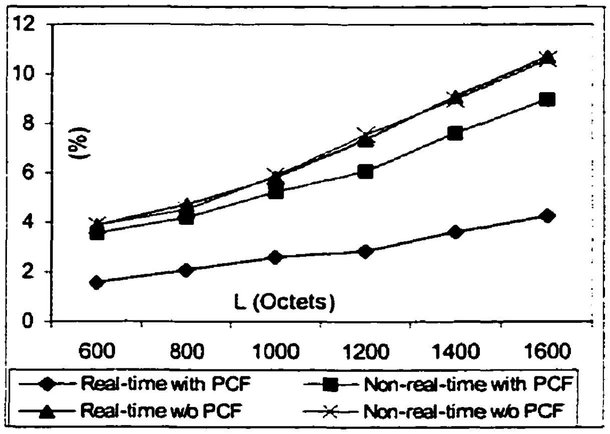




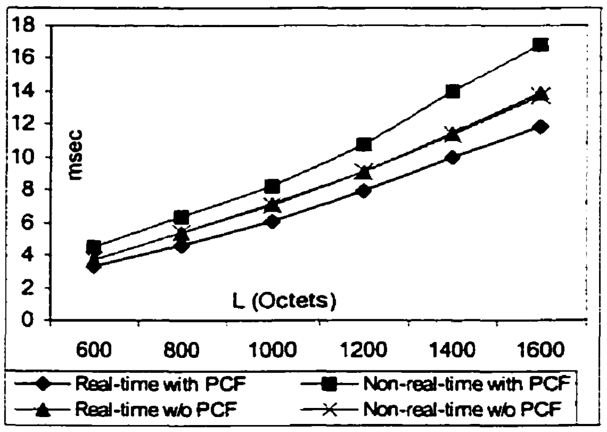
(C) λ=1, R=100, S=**11**, P=4, N=40

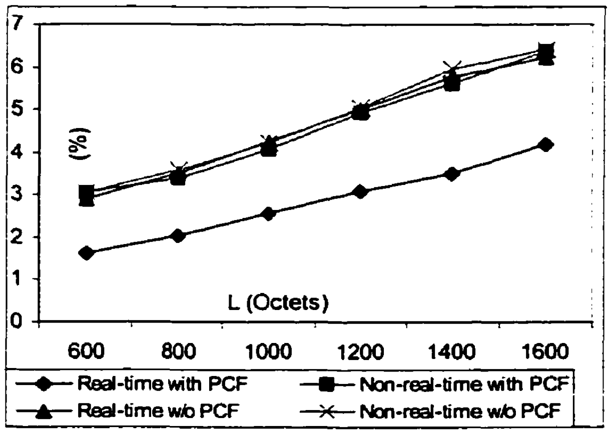
Column (I) - Average Delay Column (II) - Discard Ratio





(D) $\lambda=1$, R=100, S=1, P=4, N=**80**





(E) $\lambda=2$, R=100, S=1, P=4, N=40

Figure 4.7 Effect of packet size

4.3. Summary

This chapter evaluated the performance of our proposed MPC-MAC protocol with PCF control, through a comprehensive simulation model. The performance of MPC-MAC was compared to the IEEE 802.11 DCF-based MAC without MPC. Simulation experiments show that in all cases the use of PCF benefits real-time packets by decreasing the Average Delay (AD) and the Discard Ratio (DR). However, this may come at the expense of increasing the Average Delay for non-real-time data. The Discard Ratio for both real-time and non-real-time packets improves with the use of PCF. Therefore, our MPC-MAC outperforms the standard DCF IEEE 802.11 MAC protocol in multi-hop ad-hoc environments.

When packet arrival rate (λ) increases, both AD and DR increase. The size of data packets (L) affects AD directly, as more time is needed to transmit a larger size packet. DR also increases with the increase in L, but not as sharply as AD. With an increase in the number of nodes (N), the nodal density increases. This will also increase the traffic load, thus both AD and DR will increase. The radio transmission range (R) has more effect on the nodal density than N does. With the increase of R, the nodal density will also increase, causing AD and DR to increase dramatically when the node is in a low mobility state. The node's mobility mainly affects DR. Actually, it is the frequency of moving in and out of the neighboring MTs' transmission range that affects DR, and the higher the frequency, the higher DR.

CHAPTER 5

CONCLUSION

5.1. Summary

The IEEE 802.11 standard is the most popular MAC protocol for infrastructure-based wireless local area networks. However, in an ad-hoc environment, the Point Coordination Function (PCF), defined in the standard, cannot be readily used. This is due to the fact that there is no central authority to act as a Point Coordinator (PC). Peer-to-peer ad-hoc mode in the IEEE 802.11 standard only implements the Distributed Coordination Function (DCF). In this thesis, an efficient and on-the-fly infrastructure is created using our proposed Mobile Point Coordinators (MPC) protocol. Based on this protocol, we have also developed an efficient MAC protocol, namely MPC-MAC. Our MAC protocol extends the IEEE 802.11 standard for use in multi hop wireless ad-hoc networks implementing both the DCF and PCF modes of operation. The goal, and also the challenge, is to achieve QoS delivery and priority access for real-time traffic in ad-hoc wireless environments while maintaining backward compatibility with the IEEE 802.11 standard.

In our MPC infrastructure-creation protocol, some of the mobile nodes, based on an agreed-upon policy, are elected as MPCs and become in charge of all, or a subset, of the neighboring nodes located within their wireless communication range of association, known as the MPC range. With the "large communication range but small cluster range" concept in our MPC creation scheme, we can overcome, or at least ease, some of the problems caused by utilizing the MTs as BSs. In our proposed scheme, when two neighboring MPCs are more than one hop apart, both can initiate the PCF period at the same time without resulting in collisions. The neighboring MPCs will be farther away than the registered MT's MPC, and thus less inter-cluster interference results. We were also able to overcome the natural shortage of effective coordination methods between neighboring MPCs, which cannot conventionally communicate via any wired/wireless interconnection network, unlike BSs. Also, in a small cluster, when an MPC is shut down during the PCF period, the node that replaces it will be capable of communicating with all the cluster members. The cluster is small enough that any two nodes within the cluster can communicate directly. The small cluster range also eases the problems created by mobility: all the nodes registered to an MPC are located close to the MPC, and a registered MT has enough time to inform the MPC of its re-association before it moves out of its communication range.

We developed a packet-level simulator to test the performance of the MPC-MAC protocol in a wireless ad-hoc environment. The results were compared with the peer-to-peer mode of the IEEE 802-11 standard, which implements the CSMA/CA DCF. We have conducted experiments to observe the effects of mobility (including speed and pause period), packet arrival rate, packet length, and nodal density (including radio transmission range and number of nodes), on the performance of our protocol. We found that, our MPC-MAC has a positive effect on real-time packets by decreasing the average delay and discard rate. On the whole, the system performance of our MPC-MAC is improved compared to peer-to peer IEEE 802.11, especially when the mobility is greater. More importantly it achieves QoS performance for real-time traffic by enhancing their average delays and discard ratios.

5.2. Future work

The cluster-head protocol would facilitate the development of a comprehensive and promising framework for QoS management in wireless mobile ad-hoc networks once the proper integration of call admission control mechanisms and routing is done. The cluster-head improves the centralized management of ad-hoc networks, and makes centralized routing and call admission control possible. Consequently, high traffic loads become more manageable, and congestion can be avoided.

Our future research work will also include developing cluster-head-based call admission control and cluster-head-based routing. Our "small cluster" concept brings in one feature: neighboring cluster-heads may communicate directly within one hop. This makes direct communication between the cluster-heads without the help of "gateways" possible. Assigning channel bandwidth fairly and efficiently using cluster-heads is both challenging and interesting. Only cluster-heads need to maintain information about the whole network, while the member MTs only need to collect network topology information from their cluster-heads.

Enabling cooperation between cluster-heads and base stations is another interesting field of research. Today, base stations and low-earth-orbit (LEO) communication satellites cover large geographical areas. For example, in wireless LANs, cluster-heads can be used to carry Internet traffic to the nearest access point. Moreover, cluster-heads can be utilized to cover areas not covered by cellular base stations. They may act as complementary access points to both types of networks, cellular and ad-hoc, for which they will regulate efficient wireless access. Mobile nodes can then be guaranteed to roam seamlessly between the two types of networks.

Wireless ad-hoc routing is another hot research field. With the efficient MPC formation scheme we proposed in this thesis, routing may achieve considerable gains, since zone-MTs need not be tracked throughout the entire network. Only cluster-heads need to track zone-MTs during the re-association process. In cluster-head-based routing, when an MT wishes to send a data packet to another MT that is more than one hop away, it will only need to send the packet to its cluster-head which forwards the packet to the cluster-head in charge of the destination. Our MPC selection scheme will establish a smaller-MPC-range system that enables packet transmissions between MPCs. To achieve this, another cluster-head-based routing protocol should be designed. Using our MPC-MAC protocol and our efficient access control mechanism, we can soon witness the first generation of wireless mobile multi-hop ad-hoc networks with total conformance to existing wireless LANs.

BIBLIOGRAPHY

[1] B. Leiner, D. Nielson, and F. A. Tobagi, Eds., *Proceedings of IEEE GLOBECOM, Special issue on packet radio networks*, vol. 75, Jan. 1987

[2] I. Chlamtac, A. Farago, A.D. Myers, V.R. Syrotiuk, and G. Zaruba, "ADAPT: A Dynamically Self-Adjusting Media Access Control Protocol for Ad Hoc Networks," *Proceedings of IEEE GLOBECOM, 1999*

[3] I. F. Akyildiz and J. McNair, "Medium Access Control Protocols for Multimedia Traffic in Wireless Networks", *IEEE Network*, July/August 1999, pp.39-47.

[4] L. Kleinrock and F. A. Tobagi, "Packet switching in radio channels, part I – Carrier sense multiple-access modes and their throughput-delay characteristics," *IEEE Trans. Commun.*, vol. COM-23, Dec. 1975, pp. 1400-1416.

[5] C. R. Lin and M. Gerla, "A Distributed Architecture for Multimedia in Dynamic Wireless Networks," *Proceedings of IEEE GLOBECOM*, New York 1995, pp. 1468-1472.

[6] A. K. Parekh, "Selecting routers in ad-hoc wireless networks", ITS, 1994.

[7] M. Gerla and J. T.C. Tsai, "Multicluster, Mobile, Multimedia Radio Network", *Wireless Networks*, 1(3) 1995, pp. 255-265.

[8] C.-H.R. Lin and M. Gerla, "A Distributed Control Scheme in Multi-hop Packet Radio Networks for Voice/Data Traffic Support", *Proceedings of IEEE GLOBECOM*, 1995, pp. 1238-1242.

[9] C.-H. R. Lin and M. Gerla, "A Distributed Architecture for Multimedia in Dynamic Wireless Networks", *Proceedings of IEEE GLOBECOM*, 1995, pp. 1468-1472.

[10] D.J. Baker and A. Ephremides, ''The architectural organization of a mobile radio network via a distributed algorithm,'' *IEEE Transactions on Communications*, Nov. 1981, pp. 1694-1701.

[11] D.J. Baker, J. Wieselthier and A. Ephremides, ''A distributed algorithm for scheduling the activation of links in a self-organizing, mobile, radio network,'' *IEEE ICC'82*, pp. 2F.6.1-2F.6.5.

[12] M. Gerla and J.T.-C. Tsai, ''Multicluster, mobile, multimedia radio network,'' *ACM-Baltzer Journal of Wireless Networks,* Vol. 1, No. 3, 1995, pp. 255-265.

[13] H. Hassanein and A. Safwat, "Virtual base stations for wireless mobile ad hoc communications: an infrastructure for the infrastructure-less," To appear, *The International Journal of Communication Systems*, Vol.14, pp. 763-782.

[14] A. Safwat and H. Hassanein, "Infrastructure-based Routing in Wireless Mobile Ad hoc Networks," To appear, *The Journal of Computer Communications,* 2002.

[15] A. Safwat and H. Hassanein, "Structured routing in wireless mobile ad hoc networks," *Proceedings of the 6th IEEE Symposium on Computers and Communications (ISCC) 2001*, July 2001, pp. 332-337.

[16] S. Basagni, I. Chlamtac, and A. Farago, "A Generalized Clustering Algorithm for Peer-to-Peer Networks", *Workshop on Algorithmic Aspects of Communication (satellite workshop of ICALP)*, Bologna, Italy, July 1997.

[17] S. Basagni, "Distributed Clustering for Ad Hoc Networks", *International Symposiun on Parallel Architectures, Algorithms and Networks'*, Perth, June 1999, pp. 310-315.

[18] A. Safwat, H. Hassanein, and H. Mouftah, "Power-aware fair infrastructure formation for wireless mobile ad hoc communications," *Proceedings of IEEE GLOBECOM 2001*, vol. 5, November 2001, pp. 2832-2836.

[19] A. Safwat, H. Hassanein, and H. Mouftah, "Energy-efficient infrastructure formation in MANETs," *Proceedings of IEEE Conference on Local Computer Networks (LCN) 2001*, November 2001, pp. 542-549.

[20] D. J. Goodman and S. X. Wei, "Efficiency of packet reservation multiple access," *IEEE Trans*, *Vehic. Tech*., vol. 40, Feb. 1991, pp. 170-176

[21] N. Amitay, "Resource auction multiple access: Efficient method for fast resource assignment in decentralized wireless PCS, " *Elect. Leff*., vol. 28, Apr. 1992, pp.799-801.

[22] J. DeVile, "A reservation based multiple access scheme for a future universal mobile telecommunication system," *Proc. 7ty IEE Conf. Mobile and Pers. Commun.*, Brighton, U.K., Dec. 1993, pp.210-15

[23] M. J. Karol, Z.Liu, and K. Eng, "An efficient demand assignment multiple access protocol for wireless packet (ATM0 networks," *ACM/Baltzer J. Wireless Networks*, vol. 1, Oct. 1995, pp.267-79.

[24] F .D. Priscoli, "medium access control for the median system, " *Proc. ACTS Mobile Summit '96*, Granada,Spain, Nov.1996, pp. 1-8

[25] P. Smulders, and C. Blondia, "A MAC protocol for ATM-based indoor radio network", *rep. For European Cooperation in the Field of Scientific and Technical Research (EUCO-COST), COST 231*, TD (94),055, Apr. 1994.

[26] L. Tan and Q.Zhang, "A reservation random-access protocol for voice/data integrated spread-spectrum multiple-access systems, " *IEEE JSAC*, vol. 14, Dec. 1996, pp1717-1727

[27] G. Anastasi, D. Grillo, and L. Lenzini, "An access protocol speech/data/video integration in TDMA-based advanced mobile system," *IEEE JSAC*, vol. 15, Oct. 1997, pp. 1498-1510

[28] X. Qui, V.Li, and J.H.Ju, "A multiple access scheme for multimedia traffic in wireless ATM", *J.Mobile Networks and Apps.*,vol. 1, Dec. 1996, pp. 259-272.

[29] N. D. Wilson et al., "Packet CDMA versus dynamic TDMA for multiple access in an integrated voice/data PCN, " *IEEE JSAC*, vol. 11 Aug. 1993, pp.870-884

[30] Z. Zhang and Y.J. Liu, "Performance analysis of multiple access protocols for CDMA cellular and personal communication services," *Proceedings of IEEE INFOCOM* '93,vol.3, 1993, pp. 1214-1221

[31] *IEEE Standard for Wireless LAN LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, IEEE Standard 802.11*, 1997

[32] The Institute of Electrical and Electronics Engineers, Inc. *IEEE Std 802.11 - Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications*, 1999 edition.

[33] J.T.Geier, J.Geier "Wireless LANs: Implementing Interoperable Networks" 1st edition, pp129-157

[34] Benny Bing, "High-Speed Wireless ATM and LANs" by Artech House publishers

[35] William Stallings "Data and Computer Communications" fifth edition, pp. 393-398.

[36] F.A. Tobagi and L.Kleinrock, Packet Switching in Radio Channels: Part II – The Hidden Terminal Problem in CSMA and Busy-Tone Solution, *IEEE Trans. On Communications COM–23*, December 1975, pp. 1417 – 1433.

[37] G.Anastasi, L.Lenzini, and E.Mingozzi, "Stability and Performance Analysis of HIPERLAN", *Proceedings of IEEE GLOBECOM*, 1998

[38] T. Wilkinson, T.G.C.Phipps, Stephen K.Barton; A report on HIPERLAN Standardization", *International Journal of wireless Information Networks*; Vol.2; No 2; 1995

[39] J. L. Sobrinho and A. S. Krishnakumar, "Real-time traffic over the IEEE 802.11 medium access control layer, " *Bell Labs Tech. J.*, vol. 1, Autumn 1996, pp. 172-187.

[40] J. L. Sobrinho and A. S. Krishnakumar. Quality-of-Service in ad hoc carrier sense multiple access networks. *IEEE Journal on Selected Areas in Communications*, 17(8), August 1999, pp. 1353–1368.

[41] Z. Haas and J. Deng, "Dual Busy Tone Multiple Access (DBTMA): a new Medium Access Control for Packet Radio Networks," *IEEE International Conference on Universal Personal Communications*, October 1998

[42] Z. Haas and J.Deng, "A collision-free medium access control scheme for ad-hoc networks," *WCNC'99*, September 1999

[43] F.Talucci, M.Gerla and L.Fratta, "MACABI (MACA By Invitation): A Receiver Oriented Access Protocol for Wireless Multiple Networks," in *PIMRC '97*, Helsinki, Finland, September 1-4, 1997

[44] C.Zhu and M.Corson, "A five-phase reservation protocol (FPRP) for mobile ad hoc networks," *Proceedings of IEEE INFOCOM '98*, San Francisco, CA, Apr. 1998, pp. 322-331.

[45] A. Muir and J.J. Garcia-Luna-Aceves, "A channel access protocol for multi-hop wireless networks with multiple channels, " *Proc. ICC '98*, Atlanta, GA, June 1998.

[46] S.Singh and C. S. Raghavendra, "PAMAS- power aware multi-access protocol with signaling for ad-hoc networks, " *ACM Comp. Commun. Rev.*, vol. 28, July 1998

APPENDIX A

MPC-CREATION ALGORITHMS

This appendix contains the pseudo code for the algorithms executed by the Mobile Terminal (MT), which runs the MPC infrastructure-creation protocol. The hello_from() routine is called by an MT periodically when it is time to send a "hello" message. The hello_receipt_from() routine is called by an MT when it receives a "hello" message from a neighbor. After the MT has been turned on for the time exceeding the "observing period", if the MT finds the "best" MPC candidate, the merge_request_from() routine will be called to send a "merge request" message. An MT receiving a "merge request" message, will call the merge_request_receipt_from() routine which will send back "merge response" message. Upon receiving a "merge accept" message, the sender of the "merge request" message, if originally registered to another MPC, will call the disjoin_from() routine to send a "disjoin" message to it. The original MPC will then call disjoin_receipt_from() routine to update its topology parameters. The registration process may last for some time due to MAC contention. During the registration process, the MT may possibly find another MPC candidate that is better than the one it is trying to associate with. To guarantee that an MT would not try to re-associate with more than one node at the same time, only one instance of the merge_request_from() and the disjoin_from() routines is allowed at the any time.

hello_from()

The hello_from() routine is called periodically (see Figure A.1). Line 2 shows the creation of a new "hello" message with the values of using the five topology-parameters. If the MT is an MPC, it should contain the list of MTs that registered to it. In lines 3-7, the "hello" message is inserted into the first position of the "data sending" queue. This implies that the "hello" message will have higher priority than other non-real-time data packets.

The "hello" message is only sent during the DCF period with medium priority. After the "hello" message has been created (line 2), the routine will check the first element in the "data sending" queue to see whether it is a "hello" message or not. If it is not a "hello" message (because the last "hello" message has been sent out successfully), then the routine will insert the new "hello" frame in the first position of the "data sending" queue (line 4). If the first element of the "data sending" queue is a "hello" message, then this means that the last "hello" message has been delayed till now. The routine will delete the old "hello" message from the "data sending" queue (line 6) and insert the new "hello" message to the beginning of the queue (line 7).

```
hello_from(My_ID, My_sequence_num, My_reg_MT_num, My_MPC, My_neighboring_MPC_num, poll_MT_list)
    X = new hello_message(int my_ID, my_sequence_num, My_reg_MT_num, my_MPC,My_neighboring_MPC_num,poll_MT_list);
    if (DataSendingQueue.firstElement() is not "hello" message){ //last "hello" message has sent
        DataSendingQueue.insertAtFirst (X);
    else
        DataSendingQueue.dequeue();
        DataSendingQueue.insertAtFirst (X);
```

Figure A.1 hello_from()

hello_receipt_from()

When the MT receives a "hello" message, it will record the receiving time, and find out the signal strength. It will then pass the value of the topology parameters contained in the "hello" message with the received_time and the received_signal_strength on to the hello_receipt_from() routine. Figure A.2 shows the pseudo-code of the hello_receipt_from() routine. If the sender is the receiver's MPC, the receiver will update its "poll MT list" with the update information contained in the "hello" message (lines 2-3). If the sender is a new MPC or a new free MT, My_neighboring_MPC_num and my_sequence_num will both be increased by 1 (lines 4-6). If the sender changed roles from an MPC or a free MT, to a zone-MT, or vice versa, My_neighboring_MPC_num will be incremented or decremented by 1, respectively, and my_sequence_num will increase by 1 (lines 7-12). If the receiver is a free MT (lines 14-17), the routine will find the best MPC candidate. When the most suitable MPC candidate is found, provided that it is not itself nor the original one, the “merge_request_from()” routine will be called to send a “merge request” message to the destination (lines 16-17). If the MT is a zone-MT, it will not only merge with the new MPC (lines 18-23), but also disassociate from the original MPC by calling the “disjoin_from()” subroutine (lines 24-25). If the zone-MT finds out that its MPC is out of its MPC range (line 26), it will change its role to a free MT (line 27). After that, it will find the best MPC and will try to register with it by calling the merge_request_from() subroutine (lines 29-30). Whether the MT re-associates with another MPC or not, it will disassociate from its original MPC - the sender of “hello” message (line 31).

Best_MPC()

Figure A.3 shows the pseudo-code of the bestMPC() subroutine used to find the best MPC either by searching the entire "MT list" or by comparing the original MPC to the MTs that pass to it. In the group of candidates, there is one or none best MPC could be selected as the result of process part or all selection criteria.

```
hello_receipt_from ( Object hello_message,  received_signal_strength , received_time)
    if (sender is my MPC)
        update the poll_MT_list;
    if (sender is a new MPC or a new free node)
        My_neighboring_MPC_num ++;
        my_sequence_num ++;
    else if (sender changes roles from an MPC or a free MT to a zone-MT)
        My_neighboring_MPC_num --;
        my_sequence_num ++;
    else if (sender changed roles from a zone-MT to an MPC or a free MT)
        My_neighboring_MPC_num ++;
        my_sequence_num ++;
    if (I overpassed the observing period)
        if (I'm a free MT)
            bestMPC = bestMPC();
            if (bestMPC != myself)
                success = merge_request_from(my_ID, bestMPC.my_ID)
        else if (I'm  zone-MT)
            //check whether the sender is a better MPC candidate
            bestMPC =bestMPC (my original MPC, sender);
            if (bestMPC == my original MPC)
                //do nothing;
            else if (bestMPC ==sender)
                success = merge_request_from(my_ID, sender)
                if (success == ture)
                    success = disjoin_from(my_ID,  my original MPC);
                    if(my_MPC==sender and signal_strength < threshold)
                        my_MPC=0; //change role to free MT temporarily
                        bestMPC = bestMPC();
                        if (bestMPC is not myself) {
                        success = merge_request_from(my_ID, my_MPC.ID)
                        success = disjoin_from(my_ID, sender);
```

Figure A.2 hello_receipt_from()

```
1 bestMPC (x)
2     initial best MPC to be my original MPC or myself;
3     for any MPC candidate in x who is non-zone-MT and is in my MPC range
4         if the MPC candidate's My_reg_MT_num is larger than best MPC's
5             replace best MPC with this MPC candidate;
6         else if the MPC candidate's My_reg_MT_num is equal to best MPC's
7             if the MPC candidate's My_neighboring_MPC_num is larger than best MPC's
8                 replace best MPC with this MPC candidate;
9             else if the MPC candidate's My_neighboring_MPC_num is equal to best MPC's
10                if the MPC candidate's ID is lower than best MPC's
11                    replace best MPC with this MPC candidate;
12        return best MPC;
```

Figure A.3 bestMPC()

merge_request_from()

Figure A.4 shows the pseudo code of the merge_request_from() routine. When the MT finds a best MPC candidate that is neither itself nor its original MPC, the routine will be called. It will create a new "merge request" message, which contains the information of the source MT. The "merge request" message will be passed on to the MAC layer. If the MT does not receive a "merge response" message, or receives the message with the "disagree tag" (lines 3-5), it will mark the destination MT as "unfit MPC". If the MT receives the destination MT's "merge response" message with the "agree tag" (lines 6-10), the MT will change the value of the "my_MPC" variable to the new MPC's ID number, and increase "my_sequence_num" by 1. It also updates the "polling MT list".

```
1  merge_request_from(sender's ID, receiver's ID, "hello" message)
2          new merge_request_message(sender's ID new_MPC's ID, "hello" message);
3          if(merge_response with "disagree" information)
4                  Mark the destination MT as "unfit MPC";
5                  return false;
6          else if(merge_response with "agree" information)
7                  My_MPC = receiver's ID;
8                  Update the polling MT list;
9                  My_sequence_num ++;
10                 return true;
```

Figure A.4 merge_request_from()

disjoin_from()

As the Figure A.5 shows, the routine disjoin_from() is called to send a "disjoin" message to the MT's original MPC when it wants to disassociate. Like the "merge_request_from()" routine, the "disjoin_from()" routine first creates a "disjoin" message (line 2), then passes it to the MAC layer. If the destination MT returns back an "ACK" frame, the sender of the "disjoin" message will realize that the destination is aware of the disassociation. Otherwise, the sender will have to update its "topology parameters" accordingly.

```
disjoin_from(){
        new disjoin-message (sender's ID reciever's ID, "hello" message)
         success=MAC_send(disjoin-message); //keep sending till get ACK
        if (! success)
                MT.delete(destination MT);
                check effect on the "topology parameters";
                return false;
        else
                return true;
```

Figure A.5 disjoin_from()

merge_request_receipt_from()

If the receiver is the destination of the "merge request" message, it will call the merge_request_receipt_from() routine shown in Figure A.6. If the MT is the destination of the "merge request" message, it will reply with a "merge response" message (lines 4-12). However, before sending the "merge response" message, the MT will check the association condition. If both wireless nodes satisfy the requirement, the receiver will create the new "merge response" message with an "agree tag" (line 5) – otherwise, it will be created with a "disagree" tag (line 11). In case of a successful association, if the sender was originally a free MT or an MPC, the My_neighboring_MPC_num will be decremented by 1 (lines 8-9).

```
1  merge_request_receipt_from(sender's ID, receiver's ID, "hello" message){
2      if (my_ID ==receiver's ID )
3          search (MT_list, sender's ID);
4          if (I accept the sender's registration request)
5              new merge_response_message(accept, "hello");
6              My_reg_MT_num ++;
7              my_sequence_num ++;
8              if (hello.my_MPC==0)
9                  My_neighboring_MPC_num --;
10         else if (I reject the sender's request)
11             new merge_resposed_message(reject, "hello");
12         success=MAC_send (merge_response_message);
13         if (! success)
14             delete (destination MT);
15             check effect on the "topology parameters";
```

Figure A.6 merge_request_receipt_from()

disjoin_receipt_from()

When the MT receives a "disjoin" message, it will find out whether it is the destination of the "disjoin" message or not. If it is not the destination, it will discard the message. Otherwise, it will call the disjoin_receipt_from() routine. Figure A.7 shows the pseudo code of the disjoin_receipt_from() routine. The destinationof the "disjoin" decreases the number of MTs which are registered to it by 1 (line 4), and its my_sequence_num will be incremented by 1 to indicate the topology change (line 5). If the information in the "disjoin" message shows the sender has not registered to another MPC yet, the number of free MTs and MPCs will increase by 1 (lines 6-7).

```
disjoin_receipt_from(sender's ID, receiver's ID, hello)
    if (sender found)
        MT.updatewith(hello, received_signal_strength, received_time);
        My_reg_MT_num --;
        my_sequence_num ++;
        if (hello.my_MPC==my_ID or hello.my_MPC==0)
            My_neighboring_MPC_num++;
    else
        // do nothing
```

Figure A.7 disjoin_receipt_from()

APPENDIX B

MPC-MAC PROTOCOL

This section contains the pseudo code for the algorithms executed by the MT to send and receive data packets. If the data buffer ("data sending" queue or the "Control frame" stack) contains a data packet, the operation system will trigger the DCF_send() routine. This routine will try to send the packets using DCF defined in the IEEE 802.11 standard till the data buffer becomes empty again. When the DCF_send() routine is processing, if the packet size of the data packet is small, it will send the packet directly, otherwise, it will trigger the virtual sense mechanism, (send the "RTS packet" first and suspend activity there waiting for the "CTS packet"). After successfully receiving the "CTS", the routine then goes to the next step to send the data packet. When receiving the "RTS packet", the surrounding MTs will call the RTS_receipt_from() routine, which will reserve the medium by setting the value in the NAV. If the MT is the destination of the "RTS packet", it will send back the "CTS packet". When the surrounding MTs receive the "CTS packet" they react by setting a value in NAV the same occurs when they receive the "RTS packet". All the MTs will call PCF_initiate() periodically (once per super-frame). Only the MPC or free MT can send out "beacon signal" (which contains the PCF duration parameter) successfully to initiate a PCF period. Then, this routine will call the PCF_send() subroutine to communicate in the PCF period. In the PCF period, the PC will send a real-time packet or poll its member MTs to send real-time packets. After all real-time packets have been sent or PCF duration time expired, the PC will send the "CF_end signal" to announce the end of PCF. When the MTs around the PC receive the "beacon signal" (no matter whether they are registered to it or not), they will call the PCF_beacon_receipt_from() routine to set their NAV to a value contained in the beacon signal. If the sender is a receiver's MPC, the receiver will set "my_MPC_is_in_PCF" to the value "true". The DCF_send() routine will be suspended except for the sending of the function packets and the PCF_send() routine will be activated to supervise the existence of the MPC. If the MPC is shut down in this PCF period, certain member MTs will take over the original MPC's job and finish the rest of the PCF. When the MT receives a poll token, it will call poll_receipt_from() routine to send a real-time packet or an empty frame packet if the receiver is the destination of the "poll token". Other group members will estimate the time when they are to be polled. The MT receiving the "CF-end" signal will call the CF_end_receipt_from() routine to reset the corresponding value in its NAV to "0", which will also set the value of "my_MPC is_in_PCF" to "false" if the sender is the receiver's MPC. In the PCF, only real-time packets could be sent. When the MT plans to send a packet, it will first check the characteristics of the packet, such as the accessibility of the packet's destination. Because of the mobility of the MTs, an MT may no longer be able to reach the packet's destination - however, the packet may be coming from last relay MT that does not know the latest network state of this sending MT.

DCF_send()

When there are packets in the buffer, the DCF_send() routine will be called. Figure B.1 shows the pseudo-code of it. The DCF_send() routine will first check and try to send packets in the "control frame" stack. (lines 2-5). When sending the function packet, the routine will suspend itself until the medium remains clear for the SIFS period (line 3), then it pops one packet and sends it anyway (lines 4-5) until the stack becomes empty. If the stack is empty the routine will try to send the first packet in the "data sending" queue (lines 6-33). If the NAV is not "0" or the MT is at the PCF mode, the routine will suspend itself until the NAV reach "0" and MT go back to DCF mode (line 7). After sensing the medium idle for DIFS, it will wait for an extra time of SDN. The value of SDN is created at random or is completed from value of the last lost competition (lines 9-10). If the medium remains idle for a period equal to SDN, the routine then steps into the process of sending packets (lines 12-33); otherwise the DCF loses this round of competition. If the data packet is fairly small, the routine will send the data packet directly and suspend activity there while waiting for the ACK (lines 13-15). If the size of the data packets is larger than the RTSThreshold, then instead of sending the data packet directly, it will send the RTS packet and also suspends itself while waiting for the CTS packet (lines 17-18). If it receives the CTS, it will send the data packet (line 21) after sensing the medium for SIFS without the slot defer extension (line 20) then suspends itself waiting for the ACK (line 22). If the medium is not clear during this SIFS, this means the virtual sense method did not complete properly, and the routine will quit and restart from beginning. If the MT waits for a suspected long time without receiving the

ACK or CTS, it will write a bad record to this processing packet's send history and then quit and restart from the very beginning (lines 23-24). If the MT receives the ACK from the destination MT in time (the data packet finally completes transmission), it will delete the first packet from the sending queue (line 33). The routine will go into the process of modifying the CW depending on the collision record and a successful sending credit (lines 26-32). If the final successfully sent data has a collision record (line 26), then the CW will be doubled (line 27). If the MT has sent out several packets without experiencing a collision, the CW will halve till reaching the minimum limit (lines 29-32). During the waiting time, if the routine detects that the medium is reserved or the "Control frame" stack is non-empty, it will stop executing further steps and start from the beginning.

```
DCF_send()
    if (control_frame_stack  is not empty)
        suspend until (medium clear for SIFS);
        processing_data= control_frame_stack .pop();
        send processing_data;
    else if (control_frame_stack  is empty but "data sending" queue is not empty)
        suspend until ((NAV==0)  and my_MPC_is_in_PCF==false;
        suspend till((medium clear for DIFS)
        if(SDN =0)  //win last DCF contention
            SDN = ramdom(0, CW); //need to create new by ramdom from 0 to CW
        consume the SDN;
        if(SDN ==0)
            if((processing_data.size() <= RTSThreshold)
                sending(processing_data);
                suspend unitil ((receive ACK) or (waiting-time expire)
            }else if(processing_data.size() > RTSThreshold)
                sending(RTS_packet);
                suspend until ((receive CTS) or (waiting-time expire)
                if (receive CTS of my sent RTS)
                    suspend until the medium clear for SIFS
                    sending(processing_data);
                    suspend until ((receive ACK)or(waiting-time expire)
                    if (waiting-time expire){
                        processing_data.send_history++;
                    else if  (receive ACK of my sent packet)
                        if (processing_data.send_history>0)
                            CW = CW *2;
                            credit_for_CW_decrease =0;
                        else{
                            credit_for_CW_decrease ++;
                            if (credit_for_CW_decrease >threshold)
                                CW reduce;
                        sending_queue.dequeue();
```

Figure B.1 DCF_send()

RTS_receipt_from()

When the MT receives the RTS-packet frame, it will call the RTS_receipt_from() routine. Figure B.2 shows the pseudo-code of the RTS_ receipt_from() routine. First it will set a value in the NAV to reserve the medium. After that, it will check whether the MT is the destination of the RTS packet or not. It also needs to check the value in NAV to see whether the medium has been reserved or not. If the MT is the destination of the RTS and the medium has not been reserved for any purposes (PCF or RTS), the MT will create a new CTS packet with a modified duration time (line 5), and piggyback it to the ACK packet in the stack if there were any (line 7). It will also delete any other CTS packets in the stack if there were any (line 6).

```
1 RTS_receipt_from(sender, destination, duration_time)
2       set NAV
3       if(my_ID == destination){
4               if(NAV==0){//medium is not reserved yet
5                       CTS = new CTS(my_ID, duration_time); //create CTS
6                       control_frame_stack .delete(CTS);
7                       control_frame_stack .insertAtEnd(CTS);
```

Figure B.2 RTS_receipt_from()

CTS_receipt_from()

When MT receives the CTS-packet, it will call the CTS_receipt_from() routine, which just sets the NAV to the value contained in the CTS-packet.

```
CTS_receipt_from(CTS_frame)
    set_NAV_
```

Figure B.3 CTS_receipt_from()

packet_receipt_from()

When the MT receives the any data packet, it will call the packet_receipt_from() routine to create a new ACK and send it by pushing it on the top of "Control frame" stack and send it using the DCF_send() routine. Before inserting, it will delete any old duplicate ACK. Figure B.4 shows the pseudo-code of this routine.

```
packet_receipt_from(sender, destination){
    if(my_ID == destination){
        ACK=new ACK(my_ID, sender); //create the ACK packet
        control_frame_stack .deleteReplicate;
        control_frame_stack .push(ACK);
```

Figure B.4 Packet_receipt_from()

PCF_initiate()

MPCs initiate the PCF_initiate() routine periodically see Figure B.5. When the PCF_initiate() routine is called, it first updates its unfinished_polling_list by copying from the polling_list, and goes to compete for the medium (lines 2 -10). When it senses the medium idle for PIFS (line 3), it waits for several slots equal to its SDN to accommodate the neighboring MPCs currently competing for the medium. If the medium remains idle, then the PCF initiator will broadcast the beacon signal (line 7). The zone MT will call the PCF_Beacon_receipt_from() routine (see Figure B.7) when it receives the beacon signal. If the beacon signal is transmitted without collision, the CF period then starts. The MPC will call the PCF_send() routine to poll for real-time packets. The MT will lock its DCF and waits for the polling. If the MT detects that its MPC has shut down, its PCF_send() routine will be activated and finish the uncompleted PCF period.

```
PCF_initiate()
1     update unfinished_polling_list;
2     while (true)
3           suspend until (medium is clear for PIFS);
4           calculate the backoff_num by count the beacon record
5           consume the backoff_num;
6           if(backoff_num ==0)
7                 sending (beacon_signal);
8                 if the beacon does not suffer collision
9                       PCF_send();
10                      break from while loop; //win the medium
```

Figure B.5 PCF_initiate()

PCF_send()

In the PCF period, the MPC will call the PCF_send() (see Figure B.6) to cooperate the PCF in its group. When the PCF_send() routine is called, it will keep on polling until it finishes all real-time packets in its group or the reservation of medium for the PCF period is expired (line 1). Every node in the group will maintain the unfinished_polling_list to reflect the current waiting MT to be polled. If the MPC senses the medium idle for PIFS, it sends the poll token to specific MTs in the unfinished_polling_list (lines 3-4). The destination MT will call the PCF_poll_receipt_from() routine (see Figure B.8) to send a real-time packet if it has any or just send an empty packet frame. The MPC and other zone MT will update its unfinished polling list depending on the polled MT's response (line 6). If the polled MT sends a real-time packet to other destinations than the MPC, the MPC will wait for ACK+SIFS time to let the destination of the real-time packet confirm it receiving by sending back ACK (line 8). After one round of polling, the MPC itself also have one turn to send one real-time packet if it has any (line 9). After all the MTs with real-time packets in the group have sent their packets or the PCF period times out, the MPC will send the CF_end signal to announce the end of the PCF period (lines 10-11).

```
PCF_send(){
1       while (unfinished_polling_list is not empty and PCF duration did not expire)
2               for every destination in the unfinished_polling_list do
3                       suspend until (medium clear for PIFS);
4                       sending (poll token);
5                       suspend until ((receive respond_packet) or (waiting-time expire));
6                       refresh the unfinished_polling_list
7                       if (respond_packet is real-time packet not to me)
8                               sleep for ACK+SIFS;//respond from the destination of the real time
9               polling myself interior;
10      suspend until (medium clear for PIFS);
11      sending (CF_end signal);
```

Figure B.6 PCF_send()

PCF_Beacon_receipt_from()

Figure B.7 shows the algorithm when an MT receives a beacon signal. Every node will record the time of the beacon signal from its neighboring MPC. Once it becomes an MPC, it can predict when its neighboring MPCs will broadcast the next beacon and thus avoid the possible collision (line 1). The MTs around it, whether registered to it or not, will set a value in their NAV (line 2). If the sender of the beacon is my MPC, the my_MPC_is_in_PCF parameter will be set to "true" and PCF will start. The unfinished polling list should be updated at the very first stage and my_next_PCF_poll should be initialized and kept non-negative (lines 5-6).

```
PCF_Beacon_receipt_from(source, beacon_signal){
1       refresh the last beacon time to the correspondent MPC;
2       set in NAV the PCF duration time contained in beacon signal;
3       if I'm zone-MT of this MPC then
4               my_MPC_is_in_PCF =true; //prepare to be poll
5               refresh unfinished_polling_list;
6               initial my_next_PCF_poll;
```

Figure B.7 PCF_Beacon_receipt_from()

poll_receipt_from()

Figure B.8 shows the algorithm when the MT is polled. If the sender of the poll token is not the MPC, and is an MT on the polling list, then the sender become my MPC. If the sender is the MPC, and the MT is the destination of the poll token, it will send a real-time packet from the "data sending" queue. If the MT is not the destination of the poll token, then it will wait for the response of the polled MT to update the unfinished polling list (line 9). Finally, the my_next_PCF_poll should be synchronized (line 10).

```
poll_receipt_from(source)
1     if I am in PCF period and (source is my group-mate or my MPC)
2           if source is a group-mate then
3                 replace the MPC with this group-mate and refresh other networks parameter;
4           if(my_ID == destination)
5                 suspend until(medium clear for PIFS)
6                 send real-time or empty frame packet;
7           else if (my_ID != destination)
8                 suspend until ((hear the respond_packet) or (hear message from my MPC));
9                 refresh unfinished_polling_list depend on the respond-type;
10          refresh my_next_PCF_poll;
```

Figure B.8 poll_receipt_from()

CF_end_receipt_from()

When MT receives the CF_end signal, it just releases the value in the NAV and unlocks the DCF by setting the my_MPC_is_in_PCF parameter to "false" (see Figure B.9).

```
CF_end_receipt_from(source)
1     set in my_NAV the correspondence value to 0;
2     if(my_MPC==source)
3           my_MPC_is_in_PCF=false
```

Figure B.9 CF_end_receipt_from()

APPENDIX C

CONFIDENCE INTERVALS

Normally, confidence intervals placed on the mean values of simulation results can be used to describe the accuracy of the simulation results. Consider the results of N statistically independent simulation runs for the same experiment: X_1, X_2, ..., X_N. The sample mean, $\overline{X}$ is given as:

$$\overline{X} = \frac{\sum_{i=1}^{N} X_i}{N}$$

The variance of the distribution of the sample values, S_x^2 is:

$$S_x^2 = \frac{\sum_{i=1}^{N} (X_i - \overline{X})^2}{N-1}$$

The standard derivation of the sample mean is given by: $\frac{S_x}{\sqrt{N}}$.

Under the assumption of independence and normality, the sample mean is distributed in accordance to the T-Distribution, which means the sample mean of the simulation runs fall in the interval $\pm\varepsilon$ within the actual mean with a certain probability drawn from the T-Distribution.

$$\varepsilon = \frac{S_x t_{\alpha/2,N-1}}{\sqrt{N}}$$

where $t_{\alpha/2,N-1}$ is the value of the T-distribution with N-1 degrees of freedom with probability $\alpha/2$.

The upper and lower limits of the confidence interval regarding the simulation results are:

Lower Limit = $\overline{X} - \frac{S_x t_{\alpha/2,N-1}}{\sqrt{N}}$

Upper Limit = $\overline{X} + \frac{S_x t_{\alpha/2,N-1}}{\sqrt{N}}$

APPENDIX D

THE SIMULATION SOFTWARE STRUCTURE

The simulator used in this study uses discrete event driven simulation to simulate both real-time and non-real-time packet creation. Packets are sent by PCF and DCF. The simulation software was developed using the Java programming language. The simulation software package consists of 18 classes and subclasses. The major classes are described as follows:

Class MovingModel

The Class MovingModel simulates an MT's mobile model creator. For every MT, the MovingModel will create a series of time-place pairs, which could represent the MT's whole mobility life. For example, related to MT "a", the MovingModel creates a series of time-place pairs, that is: 1, (3,5): 5, (3,5): 9, (9,8) ~ 101, (4,5). This means MT "a" is located at (3,5) when turned on at time 1, pauses there till time 5, and begins to move evenly at time 5, reaches new location (9,8) at time 9, ... and finally turns off at location (4,5) at time 101.

Class MT

Class MT will simulate the behavior of an individual MT. Class MT is a mother thread. When an MT turns on power (the MT thread initializes), it will create and run 8 children threads that list underneath. The thread must be in one of the activate states or suspend states when it is running.

(1) **Subclass Receiver** – a thread that simulates a receiver of packets that are directed to it or broadcast. Normally, this thread stays suspended. However, the receiver will be activated to deal with any packets that are received.

(2) **Subclass PCF_initiator** – a thread which stays suspended but activates periodically by a wake-up-event sent by the **EventController** (a class introduced later). When the PCF_initiator is being activated, it will try to initialize a PCF period by sending a beacon signal with PIFS. After it finishes initializing the PCF period, it activates the PCF_send thread, pushing the next wake-up-event into the queue in the EventController and suspends itself again.

(3) **Subclass PCF_sender** – a thread which simulates PCF. This thread is activated by the thread PCF_initiator in the MPC, or the thread Receiver in a zone-MT when it receives a beacon signal. The PCF_sender thread will poll and send real-time packets during the PCF period

(4) **Subclass DCF_sender** – a thread used to simulate the DCF of IEEE 802.11. It will activate when the data-sending queue is not empty and sleeps for the time that is set in NAV if it is non-zero.

(5) **Subclass CommonPacket_creator** – a thread which simulates the arrival of non-real-time packets. When the thread is activated, it will create a non-real-time packet and push the packet into the data-sending queue. Before suspending itself, it will push a wake-up event to the queue in the EventController with "owner" set to this CommonPacket_creator and "time" set to a value that reflects the non-real-time packet arrival rate.

(6) **Subclass RealtimePacket_creator** – a thread like **CommonPacket_creator** but which simulates the arrival of real-time packets.

(7) **Subclass Hello_creator** – a thread which activates periodically to create a hello message and pushes it into first position of the data sending queue.

(8) **Subclass DCF_function_packet_sender** – a thread which simulates the sending of control packets through the DCF of IEEE 802.11.

Class Event

Class Event is an object that has field: (1) Owner: Event executor (2) Time: The time this event occurs. When the Event pushes into the queue in EventController, it must be sorted by time.

Class EventController

Class EventController simulates an event process center. The most important queue in the EventController is the time-sorted-events queue, which is used to queue the events that are sorted by time. Any steps of MT activity are events, which will be pushed into the events queue. The processing of the current event may modify certain events in the queue or create new events. EventController processes events one by one. MTs could interact with each other through interacting with events that MTs created. For example, when processing a sending event of an MT, it will create several receiving events that are to MTs in the communication range of the sending MT and push and sort the new created events into the queue. When the EventController begins to process the receiving event, the thread "Receiver" can now receive a packet from the sender.

Figure D.1 shows the system flow chart. When the simulation program starts, it will first initialize the class EventController, then run the class MovingModel. The MovingModel will create a table that schedules every MT's mobile data, including the MT start time and end time. It will also create MTs' start & end Events depending on the MTs' start & end time, then sort-insert the Events into the event queue of EventController.

After running the Class MovingModel, the simulator begins to process the first Event in the event queue. If the Event is an MT Start Event, the program will create a new MT thread and begin to sort-insert some of this MT's action Events into the event queue. If the MT Start Event is the last one of MT Start Events, the simulator program will schedule itself to start counting the data after 5 seconds warm-up. The results will be shown at 35 seconds and 65 seconds by means of sort-insert related Events into event queue. If the processing Event is an MT End Event, the simulation program will kill the MT thread and delete all the Events that relate to this killed MT in the event queue.

If the processing Event is Count start Event, the simulation program will initialize and set to 0 the count result related variable like Total Delay (of all packets), sent successfully (packets) number, and (packets) discarded by collision number. After this moment, the transmission delay of every successfully sent packet will be added up to Total Delay, and the related sent successfully packets' number will increase. The number of packets discarded would also be counted after the result count start. If the processing Event is the Show result Event, the simulation program will show the result of the Average Delay, and the Discard Ratio of this moment to the user. If this show result Event is not the last one, the program will record the result to the temporary variable for future comparison. Otherwise, the program will compare the current AD and DR to the previous values. If the difference is high, the program will discard the value and quit, otherwise, the last values will be collected.

If the processing Event is not any one of Event types above, it must be an MT related action Event. Processing this kind of event must lead to modification of the contents in the event queue or other variable. The Event processing will be completed by deleting it from the event queue. The program will go into the next round by processing the next Event, which is currently located at the front of the event queue. During the running of the simulation program, the event queue is never empty; otherwise, the running of the program will break.

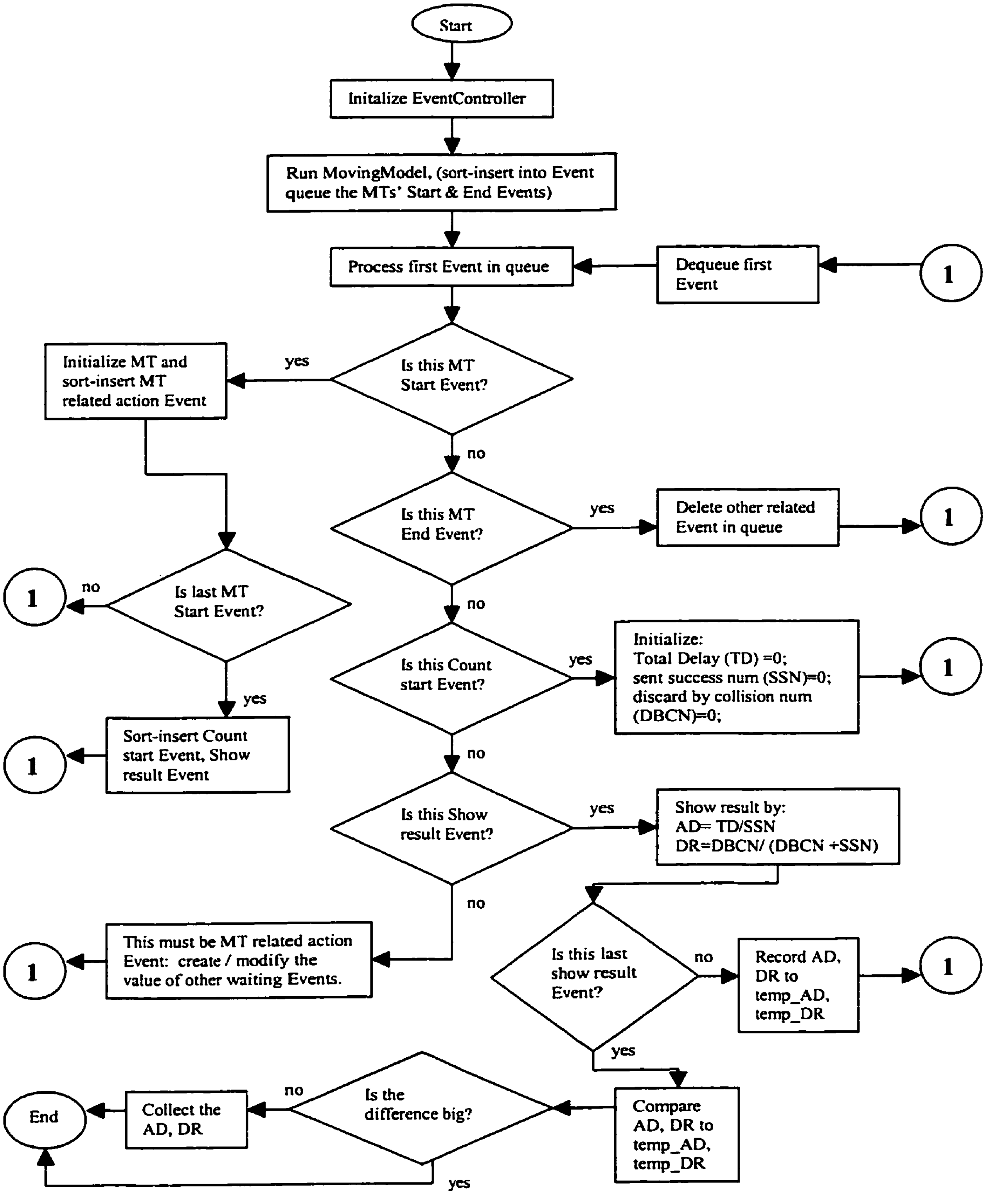

Start
Initalize EventController
Run MovingModel, (sort-insert into Event queue the MTs' Start & End Events)
Process first Event in queue
Dequeue first Event
1
Is this MT Start Event?
yes
Initialize MT and sort-insert MT related action Event
no
Is this MT End Event?
yes
Delete other related Event in queue
Is last MT Start Event?
no
yes
Sort-insert Count start Event, Show result Event
Is this Count start Event?
Initialize: Total Delay (TD) =0; sent success num (SSN)=0; discard by collision num (DBCN)=0;
Is this Show result Event?
Show result by: AD= TD/SSN DR=DBCN/ (DBCN +SSN)
This must be MT related action Event: create / modify the value of other waiting Events.
Is this last show result Event?
Record AD, DR to temp_AD, temp_DR
Compare AD, DR to temp_AD, temp_DR
Is the difference big?
Collect the AD, DR
End

Figure D.1 System flow chart