IEEE 802.11
Wireless Access Method and Physical Layer Specifications

Title: A Complete Description of Frame Prioritization in a CSMA/CA MAC Protocol

Authors:

Rick White
Motorola, Inc.
Wireless Data Group
50 E. Commerce Drive
Schaumburg, IL 60173
Tel: 1-708-576-7878
Fax: 1-708-576-7907
Email: rick_white@wes.mot.com

Mark Demange
Motorola, Inc.
Wireless Data Group
50 E. Commerce Drive
Schaumburg, IL 60173
Tel: 1-708-576-7913
Fax: 1-708-538-5152
Email: mark_demange@wes.mot.com

Kevin Doss
Motorola, Inc.
Wireless Data Group
50 E. Commerce Drive
Schaumburg, IL 60173
Tel: 1-708-576-9918
Fax: 1-708-538-5152
Email: kevin_doss@wes.mot.com

Fred Vook
Motorola, Inc.
Wireless Data Group
50 E. Commerce Drive
Schaumburg, IL 60173
Tel: 1-708-576-7939
Fax: 1-708-538-5152
Email: fred_vook@wes.mot.com

Abstract: This submission describes, in more detail, a method of prioritizing frames using a CSMA/CA Media Access Control Protocol. It also addresses the issue of contention and back-off.

Introduction

This submission describes, in more detail, prioritizing frames using a CSMA/CA Media Access Control Protocol that was initially presented in [1]. Flow charts and timing diagrams are provided to aid in the description. The issues of contention resolution and backoff are also discussed in great detail.

CSMA Protocol Specification

The basic channel access method is P-persistent CSMA with a MAC-level acknowledgment. A CSMA protocol requires the devices to "listen before talk." Before initiating a transmission, devices must first determine that the channel is idle (i.e., no other devices are transmitting). If the channel is busy, the devices must wait until the channel becomes idle. The protocol also provides an additional feature which allows some devices priority to initiate the next transmission after the channel becomes idle. The components of the protocol are discussed below.

"Listen-before-talk"

A device is permitted to transmit only when the channel appears idle. When a device is awake,it must monitor the radio channel and keep track of the channel state (idle or busy). When a device has a frame to transmit, it will first check the channel state. If the channel is idle, the device will immediately transmit the frame. If the channel is busy, the device will wait until the channel becomes idle. When the channel becomes idle, the priority mechanism described below will determine precisely when the device transmits.

A device that is asleep is unable to transmit or monitor the channel. Hence, when a sleeping device wakes up and wishes to transmit, the device must first determine the channel state (idle or busy). After determining the channel state, the device that just woke up will then follow the same procedure as the awake devices.

Packet detection is used to determine whether the channel is busy or idle. Packet detection means that a device determines the channel state by listening for the beginning of a packet. If a device synchronizes on the preamble (i.e., the start-packet indicator) of a packet, then the device knows that the channel is busy. When the device decodes the packet header, the device will read the frame length field to determine how long the frame will last. The device must set a timer based on the frame length field that will indicate when the packet has ended. When the timer expires, the device knows that the channel has become idle. If the frame header is received but fails CRC, the device will not know how long the frame will last. If a frame header fails CRC, the device will continue to monitor the time following the header with the failed CRC. If no other frame header is received in an amount of time equal to the variable *max_payload*, the device will assume the channel has become idle. The variable *max_payload* contains the maximum time duration (i.e., bytes) of the payload of a data frame.

When a sleeping device wakes up to monitor the channel, it decides whether the channel is busy or idle as follows. As soon as a frame header is received, the channel is then assumed to be busy. If, after waking up, a frame header has not been received before an amount of time equal to the largest RF frame size, then the channel is assumed to be idle.

Figure 1 below illustrates the process by which a device decides whether the channel is busy or idle.

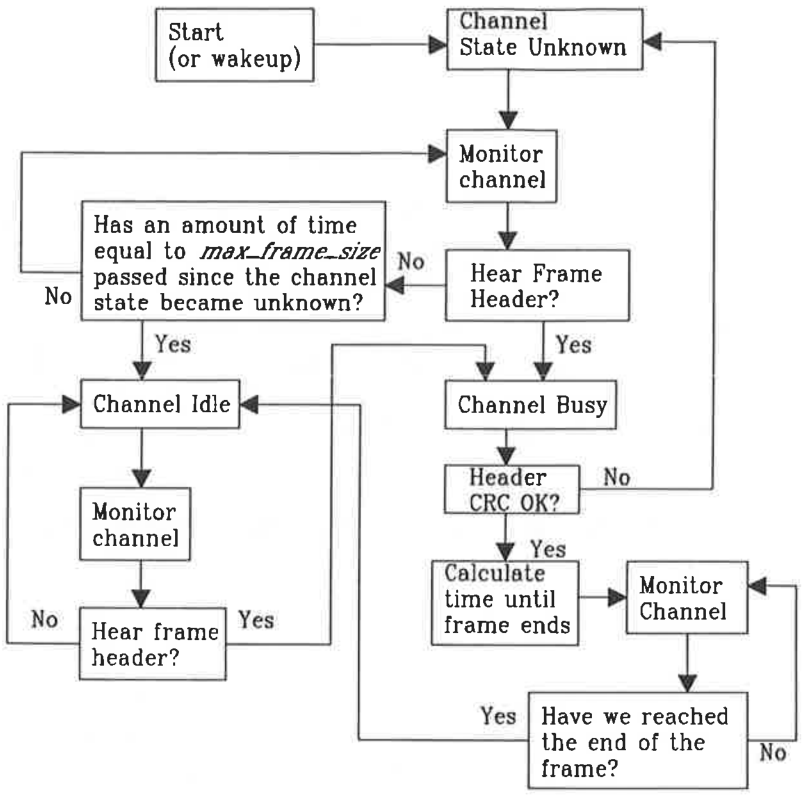


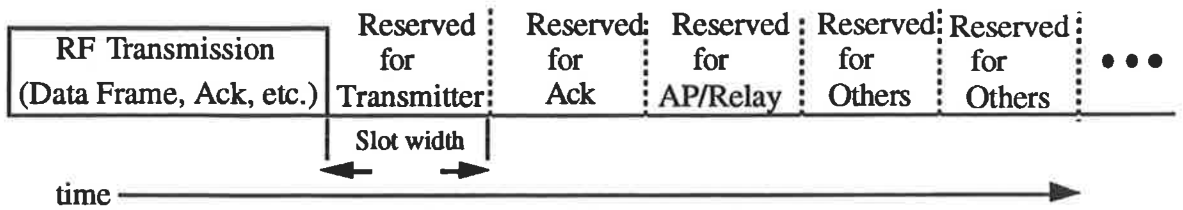
Figure 1: Determining the Channel State via Packet Detection

Priority mechanism when a busy channel becomes idle.

When a transmission is completed, the time following the transmission will be slotted. The purpose of the time slots is to allow some devices priority over other devices in initiating the next transmission. Each time slot is reserved for a different device to initiate a transmission. Before a device can initiate a transmission in its assigned slot, it must first determine if another (higher priority) device has already initiated a transmission in an earlier slot. If another device has initiated a transmission, the device must defer his transmission until the channel becomes idle again. The length of each slot is specified below.

The slots following a transmission are reserved as follows. The first slot is reserved for the transmitting device to begin transmitting another fragment of the same data packet. The second slot is reserved for the receiving device (of the just-finished transmission) to begin transmitting an acknowledgment back to the source device. The third slot is reserved for the Access Point to begin transmitting any packets (control packets, data packets, etc.) that it may need to send. (The Access Point can also use the third slot to begin relaying data frames between devices within the microcell who cannot communicate directly.) The fourth slot and beyond is used by everyone else who wishes to transmit to perform the persistence algorithm (described below).

As soon as a device initiates a transmission in its assigned slot, other devices who were at a lower priority must wait until their assigned slot shows up again following the transmission. The rules for which devices can grab a particular slot are further discussed below. The following figure depicts this channel priority mechanism.



Slots give some devices priority over other devices to initiate the next transmission.
A device can initiate a transmission in its assigned slot only if channel remains idle.
Devices wishing to transmit in the "others" slot execute P persistence algorithm.

Figure 2: Channel Priority Mechanism Following a Transmission

The length of the priority slots

There are several factors which determine the length of the priority slots. At the start of a time slot, a device will determine whether the channel is busy or idle. If the channel is idle, the device must decide whether it will transmit in that slot. If the device does decide to transmit, it must turn its radio from receive to transmit. By the end of the time slot, all other devices must be able to detect the fact that a transmission has started.

The length of a priority slot is denoted by *priority_slot_length* and is made up of three components:

1) When a device detects a clear channel at the start of a time slot, the device must decide if it has priority to transmit in that slot. The time required for this decision is related to the processing speed of the wireless device. Let the variable *decide_to_transmit* denote the amount of time required for a device to determine whether it has priority to transmit in a slot. (How the device makes the decision to transmit is discussed below.)

2) If a device does decide to transmit in a slot, there is a finite amount of time required for the device to turn its radio from receive to transmit. Let the variable *rf_turnaround* denote the time required to turn the radio from either receive to transmit or from transmit to receive.

3) When a device begins transmitting in a slot, there is a finite amount of time required for the other devices to detect that device's transmission. Let the variable *detect_start_frame* denote the time that elapses between the instant a device begins transmitting and the instant that other devices are notified that a transmission has started.

In other words, *priority_slot_length = decide_to_transmit + rf_turnaround +* detect_start_frame.

When a device stops transmitting, the first time slot is considered to start as soon as all other devices have detected the end of the transmission. The amount of time required for other devices to detect the fact that another device has stopped transmitting is called the *detect_end_frame* time. The device that finishes the transmission must wait an amount of time equal to *detect_end_frame* and then note that the first time slot has just started.

Figures 3 and 4, below, illustrate the length of the priority time slots. Figure 3 shows an example where a device seizes the first time slot following another device's transmission. Figure 4 shows an example where a device seizes the second time slot.

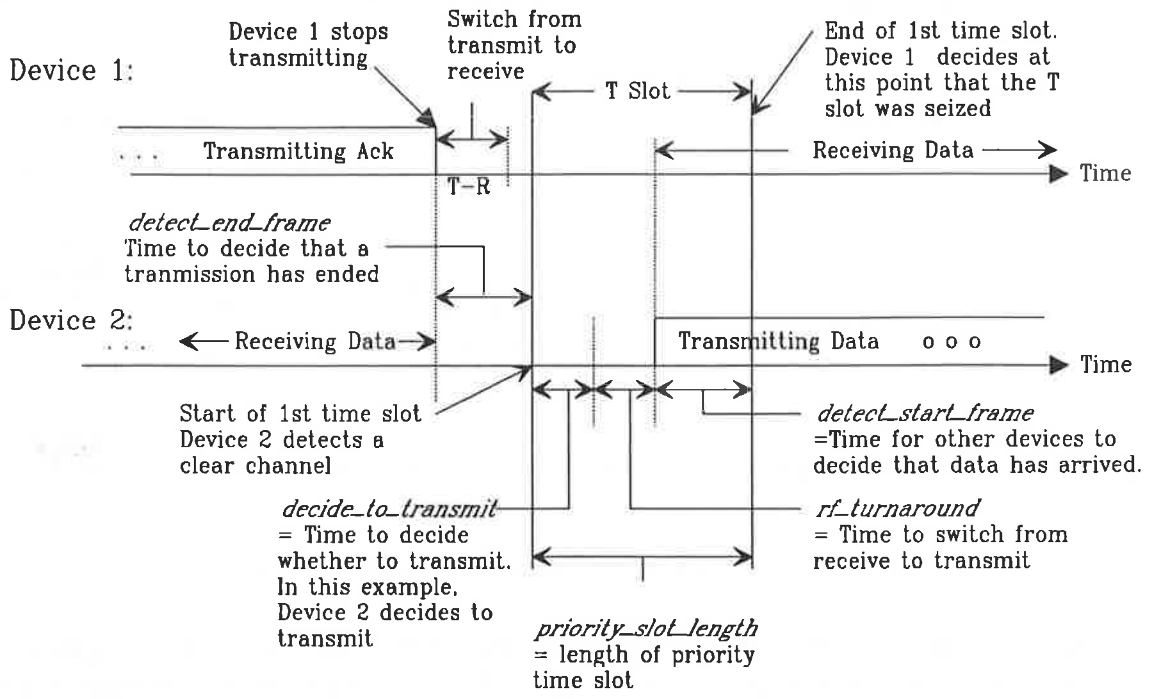

Device 1:
Device 1 stops transmitting
Switch from transmit to receive
T Slot
End of 1st time slot. Device 1 decides at this point that the T slot was seized
. . . Transmitting Ack
Receiving Data
Time
T–R
detect_end_frame
Time to decide that a tranmission has ended
Device 2:
. . . Receiving Data
Transmitting Data o o o
Time
Start of 1st time slot Device 2 detects a clear channel
detect_start_frame
=Time for other devices to decide that data has arrived.
decide_to_transmit
= Time to decide whether to transmit. In this example, Device 2 decides to transmit
rf_turnaround
= Time to switch from receive to transmit
priority_slot_length
= length of priority time slot

Figure 3: Length of the Priority Time Slots: a Device Seizes the First Time Slot

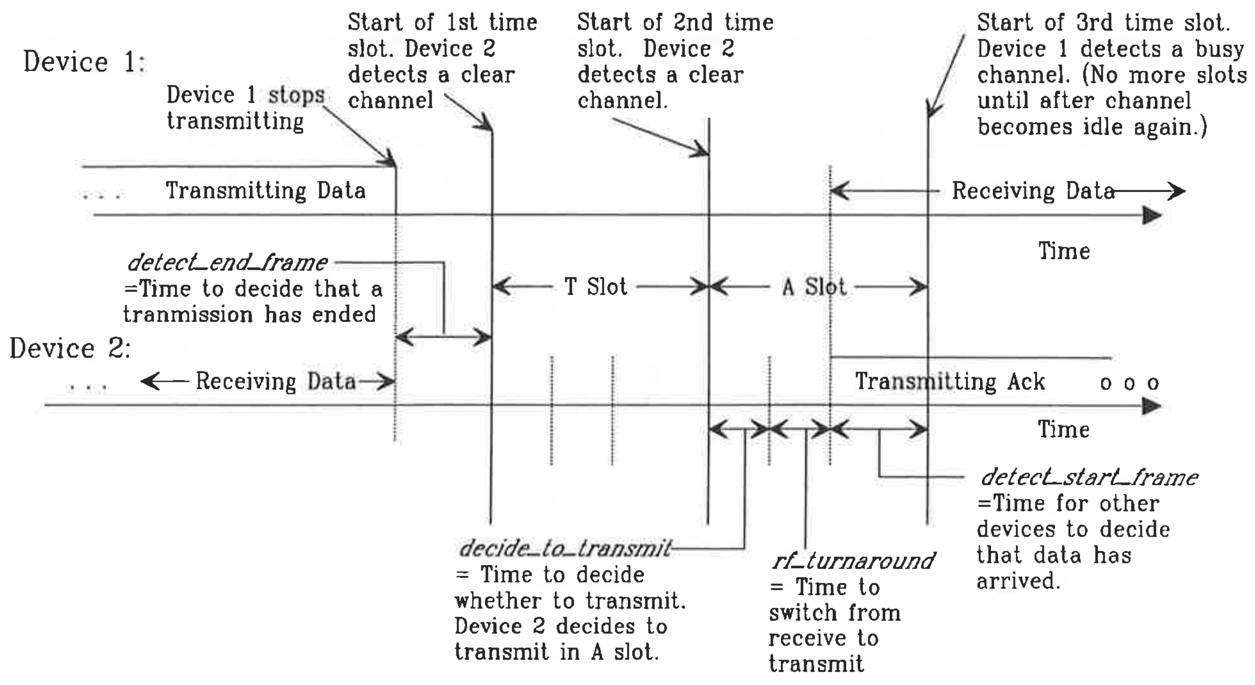

Device 1:
Device 1 stops transmitting
Start of 1st time slot. Device 2 detects a clear channel
Start of 2nd time slot. Device 2 detects a clear channel.
Start of 3rd time slot. Device 1 detects a busy channel. (No more slots until after channel becomes idle again.)
. . . Transmitting Data
Receiving Data
Time
detect_end_frame =Time to decide that a tranmission has ended
T Slot
A Slot
Device 2:
. . . Receiving Data
Transmitting Ack o o o
Time
detect_start_frame =Time for other devices to decide that data has arrived.
decide_to_transmit = Time to decide whether to transmit. Device 2 decides to transmit in A slot.
rf_turnaround = Time to switch from receive to transmit

Figure 4: Length of the Priority Time Slots: a Device Seizes the Second Time Slot

The "transmitter" and "acknowledgment" priority slots.

The purpose of the "transmitter" and "acknowledgment" slots is to provide a device the ability to maintain control of the channel until the device finishes transmitting all fragments of its packet. The "acknowledgment" slot gives the destination device priority to transmit an acknowledgment back to the source device should the source device not seize the "transmitter" slot to send more fragments. The purpose here is to prevent other devices from seizing the channel away from the device that must send the acknowledgment.

After a device has transmitted a data fragment, it should have priority to seize the T slot and transmit more fragments of the same data packet. The device shall transmit some number (the fragment window size) of data fragments and then release the T slot and wait for an acknowledgment. The variable *fragment_window_size* shall determine the maximum number of fragments (of the same packet) a device may transmit before releasing the T slot.

A device must release the T slot before reaching the fragment window size if no more fragments of the data packet need transmitting. The device should release the T slot to allow an acknowledgment to be transmitted.

When a source device releases the T slot following its data fragments, it will immediately turn its radio around and monitor the channel for an acknowledgment frame from the destination device. When the destination device hears an idle transmitter slot, it must then transmit an acknowledgment frame to the source device. The destination device will seize the acknowledgment slot and start transmitting the acknowledgment. Figure 4, above, shows an example of this situation. In Figure

4, device 1 is sending a data frame to device 2. When device 2 sees an idle T slot, device 2 sends an acknowledgment in the A slot.

When the destination device has finished sending the acknowledgment, the transmitter slot following the acknowledgment is then reserved for the source device to continue (if necessary) with more fragments of the same packet. The device sending the acknowledgment does not have permission to grab the "transmitter" slot that immediately follows the acknowledgment. Figure 2, above, shows an example of this situation. In Figure 2, device 1 is sending an acknowledgment to device 2, and device 2 seizes the T slot following device 2's ack.

If the source device does not receive an acknowledgment frame, the source device will attempt to retransmit the data frame(s) at a later time (according to the random exponential backoff algorithm described below). When the time arrives to retransmit the data frame(s), the source device will compete for access in the "others" slots, described below.

To summarize the above discussion, a device may grab the T slot only when the following conditions are all satisfied:

- Either the device has just finished transmitting a data fragment or the device has just received an acknowledgment of its own data fragment.
- The device has previously transmitted at least one fragment of a data packet and more fragments of the same packet require transmission.
- The number of consecutive T slots the device has grabbed (without waiting for an acknowledgment) has not exceeded the *fragment_window_size*.

The following guidelines also apply.

- A transmitting device with permission to grab the T slot following its own transmission need not turn its radio around to sense the channel at the start of the T slot. The device can simply continue in transmit mode and start transmitting the next fragment as soon as the current fragment is finished. At the end of the T slot, other devices will hear a busy T slot as required.
- When a device has transmitted a frame other than a data frame, it does not have priority to grab the T slot following that frame.
- When a data packet has been successfully delivered, the device does not have priority to grab the T slot following the last acknowledgment of the last fragment.

A device will grab the A slot only when the following conditions are all satisfied:

- The device has just received a frame that requires acknowledging.
- The T slot is detected idle.

Note that none of the above conditions require the receiving device to be concerned with the value of *fragment_window_size*.

If a multi-fragment packet does not require an acknowledgment (for example, a broadcast/multicast packet transmitted by the Access Point), the sending device will transmit all fragments of the packet without releasing the T slot. The sending device will ignore the fragment window rule and continue with all fragments of the packet until the packet is complete.

The "AP / Relay" slot.

The AP/Relay slot serves multiple purposes. First, this time slot is used by the Access Point to intervene in the channel and transmit any data packets or control packets that it may need to send. The types of control packets that the Access Point may need to transmit in this slot will be described in future submissions. The Access Point can also seize the relay slot to initiate the transmission of a packet from the wireline network to a wireless device.

The second purpose of the slot is to allow the Access Point the ability to seize the channel to perform relay between two devices that are out of range of each other. The Access Point will relay packets between the source and destination. The Access Point can use the third priority slot (the "relay" slot) to initiate this relay once it has received the entire packet from the source device.

Figure 5 below illustrates the transmitter, acknowledgment, and relay slots. At the top of Figure 5 is a two device network with an Access Point. A line connecting two devices indicates that the two devices can hear each other over the RF channel. The three horizontal lines labeled 1, 2, and AP represent a timeline for each device. The blocks labeled Frag X, Ack, Control, represent transmissions initiated by the device. In this figure, device 1 has a packet that has been divided into four smaller fragments for transmission over the RF. Device 1 transmits two fragments and then waits for an acknowledgment. While device 1 is transmitting, the Access Point is waiting for an opportunity to transmit (for example) a control packet.

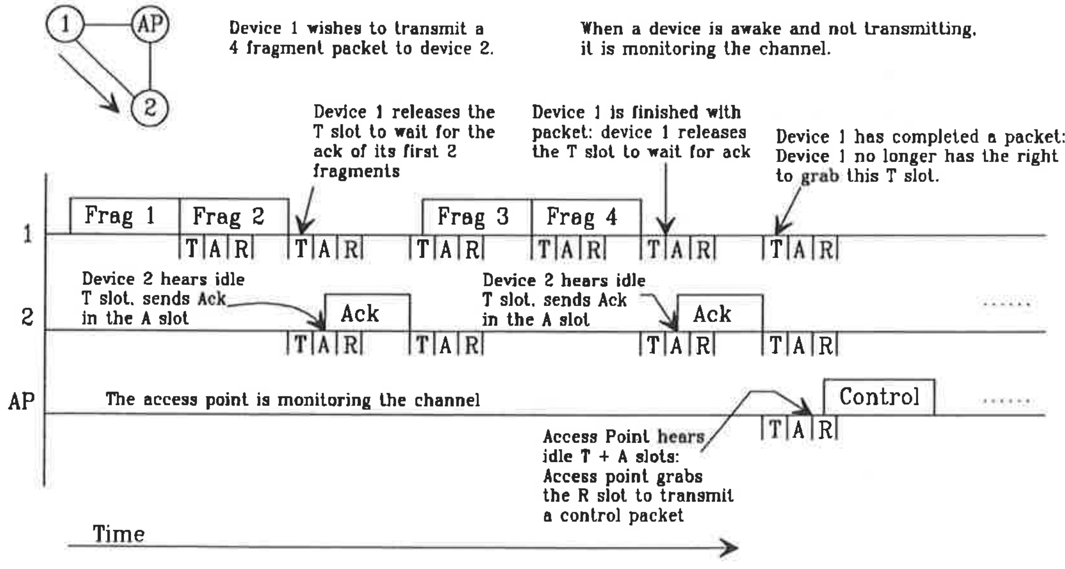

1
AP
2
Device 1 wishes to transmit a 4 fragment packet to device 2.
When a device is awake and not transmitting, it is monitoring the channel.
Device 1 releases the T slot to wait for the ack of its first 2 fragments
Device 1 is finished with packet: device 1 releases the T slot to wait for ack
Device 1 has completed a packet: Device 1 no longer has the right to grab this T slot.
Frag 1
Frag 2
Frag 3
Frag 4
T A R
Device 2 hears idle T slot, sends Ack in the A slot
Ack
Control
The access point is monitoring the channel
Access Point hears idle T + A slots: Access point grabs the R slot to transmit a control packet
Time

Figure 5: Transmitter, Acknowledgment, and Relay slots

The "Others" Time Slot and the P-Persistence Algorithm.

When the "transmitter", "ack", and "relay" slots have elapsed without a higher priority device (or AP) initiating a transmission, a device can then transmit its data frame in the "others" slots according to the persistence algorithm. The persistence algorithm is as follows. A device wishing to transmit in a slot will initiate its transmission with probability equal to *p_persistence*. With probability = (1-*p_persistence*) the device will continue to monitor the channel for another period of time equal to the slot length (*priority_slot_length*). After waiting until the next slot, the device will again monitor the channel. If the channel is still idle, the device will transmit with probability *p_persistence* and defer until the next slot with probability (1-*p_persistence*). If another device is transmitting, then the device will wait until the channel goes idle, wait until the "others" slot appears, and then repeat the above steps. The value of *p_persistence* can range from *p_persistence*>0 to *p_persistence*=1.

In some cases, a device may have successfully delivered some fragments of a larger packet and may be competing for access in the O slots to transmit more fragments of the same packet. If the device's last data fragment was acknowledged, then a device in this situation is said to have a *packet in progress*. Such a device has lost the permission to the transmitter slot following the acknowledgment of its last fragment and is competing with all other devices for access in the O slots.

Devices with a packet in progress shall use a higher transmission probability than devices who do not have packets in progress. This strategy gives an advantage to devices that lost permission to the T slot following an acknowledgment due to an impending dwell boundary. Devices with packets in progress shall use a transmission probability of *p_persistence_pip*. The value of *p_persistence_pip* must be greater than *p_persistence*.

There shall be some configurable number *(num_o_slots)* of "others" slots following a transmission. That is, when a device wishes to transmit a fragment and a number greater than *num_o_slots* time slots have passed since the last transmission, the device will not need to execute the P-persistence algorithm.

Figure 6 below illustrates the method for accessing the channel in the "others" slot via the P-Persistence algorithm:

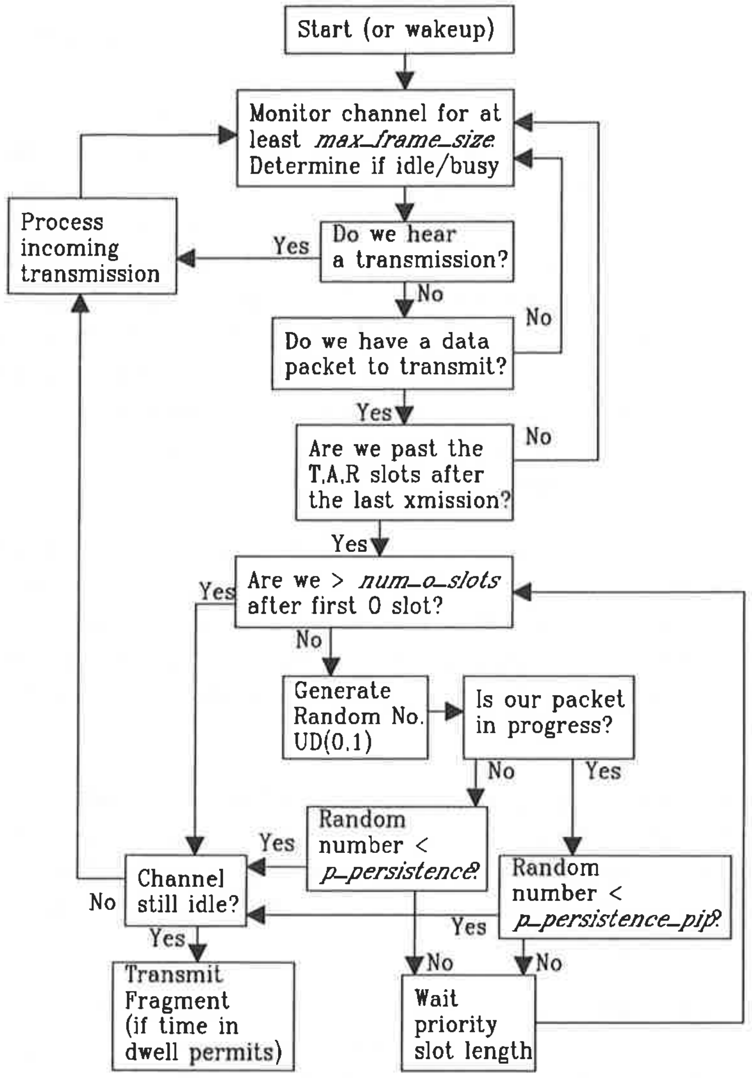

Start (or wakeup)
Monitor channel for at least *max_frame_size*. Determine if idle/busy
Do we hear a transmission?
Yes
Process incoming transmission
No
Do we have a data packet to transmit?
No
Yes
Are we past the T,A,R slots after the last xmission?
No
Yes
Are we > *num_o_slots* after first O slot?
Yes
No
Generate Random No. UD(0,1)
Is our packet in progress?
No
Yes
Random number < *p_persistence*?
Yes
Random number < *p_persistence_pip*?
Yes
Channel still idle?
No
Yes
Transmit Fragment (if time in dwell permits)
No
No
Wait priority slot length

Figure 6: P-Persistence Algorithm

Figure 7, below, contains a timeline showing how a device (without a packet in progress) accesses the channel via the p-persistence algorithm. Devices 3 and 4 generate packets while the channel is occupied by device 1. Devices 3 and 4 must wait until the channel goes idle and execute the p-persistence algorithm. When hearing idle T, A, and R slots, devices 3 and 4 flip their coins to decide whether to transmit in the first O slot. With probability P, a device will transmit in the first O slot. The coins of both devices 3 and 4 say do not transmit in the first O slot. Since the channel is still idle after the first O slot, devices 3 and 4 flip their coins again. This time, device 3 gets the go-ahead to transmit, but device 4's coin says do not transmit. After the second slot, device 4 hears a busy channel and must wait until the next set of O slots occur before attempting to transmit again. After transmitting, device 3 then has permission to grab the T slots after its first transmission. Meanwhile, the Access Point is monitoring the channel, but does not seize the R slots since it has no packets to transmit.

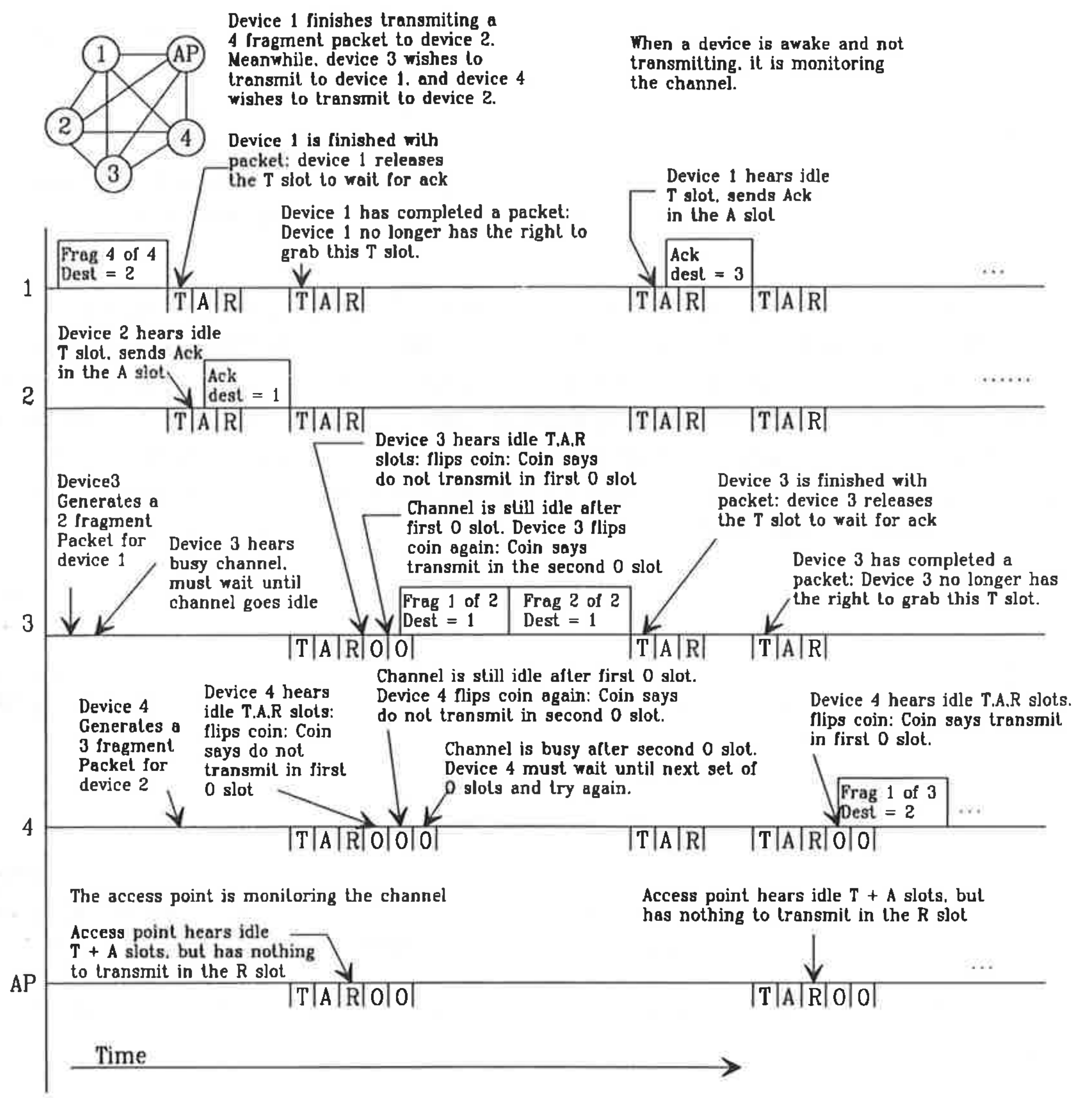

1
AP
2
4
3
Device 1 finishes transmiting a 4 fragment packet to device 2. Meanwhile, device 3 wishes to transmit to device 1, and device 4 wishes to transmit to device 2.
When a device is awake and not transmitting, it is monitoring the channel.
Device 1 is finished with packet: device 1 releases the T slot to wait for ack
Device 1 has completed a packet: Device 1 no longer has the right to grab this T slot.
Device 1 hears idle T slot, sends Ack in the A slot
Frag 4 of 4 Dest = 2
Ack dest = 3
T A R
Device 2 hears idle T slot, sends Ack in the A slot
Ack dest = 1
Device 3 hears idle T,A,R slots: flips coin: Coin says do not transmit in first 0 slot
Device3 Generates a 2 fragment Packet for device 1
Device 3 hears busy channel, must wait until channel goes idle
Channel is still idle after first 0 slot. Device 3 flips coin again: Coin says transmit in the second 0 slot
Device 3 is finished with packet: device 3 releases the T slot to wait for ack
Device 3 has completed a packet: Device 3 no longer has the right to grab this T slot.
Frag 1 of 2 Dest = 1
Frag 2 of 2 Dest = 1
T A R 0 0
Channel is still idle after first 0 slot. Device 4 flips coin again: Coin says do not transmit in second 0 slot.
Device 4 Generates a 3 fragment Packet for device 2
Device 4 hears idle T.A.R slots: flips coin: Coin says do not transmit in first 0 slot
Channel is busy after second 0 slot. Device 4 must wait until next set of 0 slots and try again.
Device 4 hears idle T.A.R slots. flips coin: Coin says transmit in first 0 slot.
Frag 1 of 3 Dest = 2
T A R 0 0 0
The access point is monitoring the channel
Access point hears idle T + A slots, but has nothing to transmit in the R slot
Access point hears idle T + A slots, but has nothing to transmit in the R slot
Time

Figure 7: The "Others" Slots and P-Persistence

Backoff Algorithm

There are two cases in which the wireless MAC must refrain from transmitting. The first case is when a user data frame arrives (from the local host) and the radio channel is sensed to be busy. In this case, the wireless MAC implements the priority and P-persistent algorithms described earlier. The second case is when the wireless MAC transmits a data frame but fails to receive an acknowledgment frame.

There are three possible causes for missing an acknowledgment:

1) The data frame failed to reach its intended receiver.

2) The acknowledgment frame was not received by the source device.

3) A header CRC failure on a short fragment causes the receiving device to monitor the channel for *max_payload*, an amount of time that happens to be much longer than the actual frame size. Meanwhile, another device hears an idle channel following the end of the frame and grabs an O slot before the receiving device can send its acknowledgment. In this case, the device who lost its priority to send the acknowledgment must contend for access in the next set of O slots to send its acknowledgment.

When no acknowledgment is received in the acknowledgment slot, the wireless MAC will implement a "binary exponential" backoff algorithm. The term "exponential" means that the amount of time that the wireless MAC waits before retransmitting a lost data frame will be exponentially distributed with a mean time *mean_backoff_time*. The term "binary" means the average backoff time (*mean_backoff_time*) for each successive retry will double compared to the previous retry. As an example, if the mean backoff time for the first retry were 10 msec, the mean for the second retry would be 20 msec, the third retry 40 msec, etc.

The binary exponential backoff time can be calculated in the following manner. Assume a configurable system parameter, *mean_backoff_time*, which is the mean backoff time for the first retry.

1) Generate a random variable uniformly distributed between 0 and 1 (U(0,1)).

2) Generate the backoff time for the i^{th} retry with the following formula:

   backoff = $-\textit{mean\_backoff\_time}*\ln(U(0,1))*2^{(i-1)}$

This formula represents a well known technique for converting a uniformly distributed random variable into a exponentially distributed random variable.

When a device fails to hear an acknowledgment, the device will execute this backoff. The main purpose of backoff is to resolve the contention between two or more devices that transmitted at the same time. When the device's backoff timer expires, the device will attempt to retransmit the fragment window. If two devices had attempted to transmit at the same time and are now backing off, the two devices will likely choose different backoff times. If the two devices pick backoff times that differ by more than a slot width, one device will gain access to the channel while the other must defer.

While a device is backing off, the device will continue to monitor the channel in case the destination device lost its priority and eventually does transmit the acknowledgment. When a device receives the acknowledgment the device must cancel its backoff and will have priority to seize the T slot following the acknowledgment.

There will be a maximum amount of time that a sending device is allowed to use in transmitting a packet. This amount of time is called the *packet_lifetime*. When a device generates a packet to transmit, the device starts incrementing the packet lifetime timer. If the final acknowledgment indicating successful delivery is not received before the packet lifetime timer reaches *packet_lifetime*, the device will give up on transmitting the packet and throw out the packet.

When the device gives up on transmitting a packet (the *packet_lifetime* timer expires), the device should throw out all other packets destined for the same destination (broadcast destination excluded).

If the channel is busy when the backoff timer expires, the device must wait until the channel goes idle before attempting transmission in an "others" slot using the P-persistence algorithm.

Figure 8 illustrates the backoff algorithm. This flo·v chart assumes a configurable parameter that sets the maximum time allowed for attempting to transmit a packet before the sending device gives up on the packet.

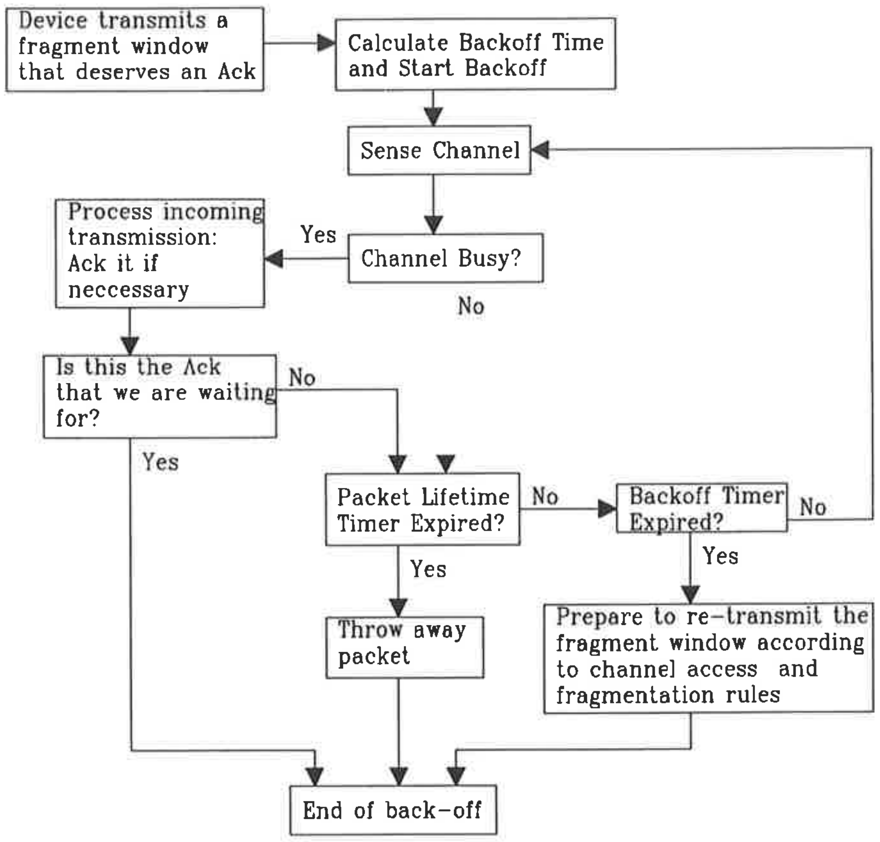

Device transmits a fragment window that deserves an Ack
Calculate Backoff Time and Start Backoff
Sense Channel
Process incoming transmission: Ack it if neccessary
Yes
Channel Busy?
No
Is this the Ack that we are waiting for?
No
Yes
Packet Lifetime Timer Expired?
No
Backoff Timer Expired?
No
Yes
Yes
Throw away packet
Prepare to re-transmit the fragment window according to channel access and fragmentation rules
End of back-off

Figure 8: Backoff Algorithm

Conclusion

This submission provides a foundation for future submissions that will address topics such as operation in a frequency hopping environment, synchronization, association, link transfer, and power management.

References

[1] Rick White Motorola, "Frame Prioritization in a CSMA/CA Media Access Control Protocol", Doc IEEE P802.11-93/159 Sept 1993.