Specification and Analysis of the DCF and PCF Protocols in the 802.11 Standard Using Systems of Communicating Machines

Moustafa A. Youssef*, Arunchandar Vasan*, Raymond E. Miller
{moustafa, arun, miller}@cs.umd.edu
Department of Computer Science
University of Maryland
College Park, MD 20742

Abstract

We propose formal models for coordination functions of the 802.11 MAC layer using systems of communicating machines and analyze them. We model the basic DCF (CSMA/CA) protocol, the DCF protocol with RTS/CTS (MACA), and the PCF protocol. Analyses show the following safety results: the CSMA/CA protocol is free from deadlocks and non-executable transitions, the MACA has a potential livelock, but is free from deadlocks and non-executable transitions, and the PCF protocol is free from deadlocks and non-executable transitions. We also show that liveness is guaranteed in the PCF protocol.

1 Introduction

802.11 [1] is an emerging IEEE standard for broadband Wireless LANs (WLANs) The main issue addressed is sharing the wireless communication medium. The protocols used for sharing the medium are called coordination functions. They are the basic DCF (Distributed Coordiantion Function), DCF with RTS/CTS (Request to Send/Clear to Send), and PCF (Point Coordination Function).

We formally specify the basic DCF protocol (CSMA/CA), the DCF with RTS/CTS (MACA), and the PCF protocol using systems of communicating machines (SCM). The former two are modeled in the context of the *ad-hoc* mode, while the latter is modeled in the infrastructure mode in the presence of AP (Access Point). We model the DCF operation in the ad-hoc mode as the AP behaves like an ordinary machine, i.e., it does not do anything central in DCF.

We demonstrate that the CSMA/CA and the MACA protocols are free from deadlocks and non-executable transitions. The MACA protocol is shown to have a potential

---

livelock, which may not be seen in practice. The PCF protocol is shown to be free from deadlocks and non-executable transitions. In addition, we prove a liveness result for the PCF protocol.

We briefly review some 802.11 terminology in section 2. We follow this with a review of the Systems of Communicating Machines model in section 3. Section 4 formally specifies and analyses CSMA/CA. Sections 5 and 6 deal with the specification and analysis of MACA and PCF respectively. We conclude in section 7.

2 The 802.11 MAC layer protocols

We review some of the terminology used by the 802.11 specification in this section. Details can be found in [1]. A basic service set (BSS) is a set of wireless stations that communicate with each other and outside the set using an AP, which is possibly connected to a wired LAN. An Independent Basic Service Set (IBSS) is a set of stations that communicate directly without making use of an AP. A BSS operates in infrastructure mode while an IBSS operates in ad-hoc mode.

A coordination function shares the medium among the stations, i.e., decides when a station can send data. The DCF is a distributed algorithm, where all stations run the algorithm. The PCF achieves coordination by a centralized algorithm, where the AP runs the algorithm.

2.1 Basic DCF

The basic DCF is Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA). Carrier sensing is done through physical and virtual mechanisms. A station senses the medium to check if another station is transmitting in the physical mechanism. The virtual carrier-sense is done by distributing reservation information with RTS/CTS Exchanges in MACA (described later).

---

*Student authors in reverse alphabetical order

1

If the WM is not busy, the station waits for a DIFS (DCF Interframe Spacing) duration. If the medium were to remain free for a DIFS, the station transmits the DATA frame. If the medium becomes busy during the DIFS time interval, the backoff procedure is enabled. Similarly, backoff is enabled if the medium were found busy when sensing for the first time.

When the backoff is invoked, the station waits till the current transmission is over. After the end of the current transmission, the station waits for a DIFS. Once the medium was detected to be idle for a DIFS, the station performs an additional backoff wait before actual transmission. The backoff timer is set to some randomly chosen value. If the medium is free throughout the backoff interval and the timer expires, the frame is transmitted. If the medium becomes busy during this interval, the timer is *frozen* at its current value but *not reset*. The station waits for the medium to become free, waits for an DIFS, and performs the backoff wait again, reducing the timer. This process continues till the backoff timer finally expires and the station transmits the frame.

When a station receives a DATA frame, it waits for a SIFS (Short IFS) duration and transmits an ACK to the sender. There is no carrier sensing or backing off for ACK frames. The sender retransmits if no ACK is received within a specified duration.

2.2 DCF with RTS/CTS: MACA

A station which needs to send data sends an RTS (Request to Send) frame in the normal CSMA/CA style. The receiver when it receives the RTS frame, sends a CTS frame after waiting for an SIFS. The sender sends its data frame after an SIFS after it gets the CTS. Likewise, on receiving a data frame, a station waits for an SIFS and sends an ACK.

The RTS/CTS frames have two octets, which specify the time for which the medium is reserved. If a machine receives an RTS/CTS of duration τ at time t, it marks the medium as busy in its NAV (Network Allocation Vector) for the time interval $[t, t + \tau]$. A machine does not transmit any DATA or CTS when its NAV is busy. This is the virtual carrier sensing. RTS/CTS reduces the number of collisions and solves the *hidden* and *exposed* terminal problems [1],[2].

2.3 PCF Operation

Point Coordination Function (PCF) is a centralized, polling-based access mechanism which requires the presence of an AP that acts as Point Coordinator (PC). In the PCF mode, time is divided into superframes. Each superframe consists of a contention period where DCF is used and a contention-free period (CFP) where PCF is used. The CFP is started by a beacon frame sent by the PC using DCF.
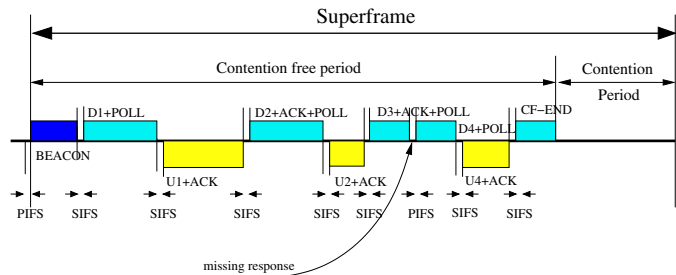

Figure 1. Structure of superframe

The CFP may vary from superframe to superframe, as the base station has to contend for the medium. Once the CFP starts, the PC polls each station in its polling list (the high priority stations) when they can access the medium. To ensure that no DCF stations are able to interrupt this mode of operation, the inter frame space between PCF data frames (PIFS) is shorter than the DIFS. To prevent starvation of stations that are not allowed to send during the CFP, there must always be room for at least one maximum length frame to be sent during the contention period.

The PC polls the stations in a round-robin fashion . A polled station always responds to a poll. If there is no pending transmission, the response is a null frame containing no payload. If the CFP terminates before all stations have been polled, the polling list is resumed at the next station in the following CFP cycle. A typical medium access sequence during PCF is shown in Figure 1. A station being polled is allowed to transmit a data frame. In case of an unsuccessful transmission, the station retransmits the frame after being repolled or during the next Contention Period.

3 Systems of Communicating Machines

We review the basic formalism used in systems of communicating machines. The reader is referred to [3], [4] for more details.

All sets are finite in the following description. A system of communicating machines is an ordered pair $C = (M, V)$ where M is a set of machines, and V is a set of shared variables. For each machine i, there exist two subsets of V namely R_i and W_i which contain the variables which machine i has read access and write access respectively.

Each machine $m \in M$ is a 5-tuple given by (S, s, L, N, τ) where:

- S is the set of states in the machine.
- s is a specially designated state called the starting state.
- L is the set of variables local to the machine.
- N is a set of names, each of which is associated with a unique pair (p, a) where p is a predicate on the variables of $L \cup R_i$, and a is an action on the variables of $L \cup R_i \cup W_i$. An action is a partial function from the

values of the local and read action variables and write action variables.

- $\tau : S \times N \to S$ is a transition function which is a partial function from the states and names to the set of states.

The system state tuple is the tuple of all the machine states. The global state is the system state tuple taken along with the values of all the variables. The system state is the system state tuple taken along with all the enabled transitions. A transition is enabled when the predicate associated with it evaluates to be true.

All machines start in their respective start states. Intuitively, when a transition is enabled, the associated action changes the variables it can modify, i.e., those in W_i, as an effect. These changed variables will now enable other transitions, and the machines move between their states.

4 Formal Specification of CSMA/CA

We model only unicast traffic from station to station. In our model, DCF operation handles errors by retransmissions, while we assume no transmission errors in PCF operation. In DCF with RTS/CTS, i.e., MACA, all machines are assumed to implement MACA or all machines implement just basic DCF.

We generalize the timer machine introduced in [5]. The predicate action table for the generalized timer machine is shown in table 1, and the state description is given in figure 2. The timer machine starts in state 0. It has two variables: *st* and *val*. The *st* variable, standing for status, can be one of ACT, INACT, and EXP. ACT stands for the timer being active, i.e., ticking. INACT stands for the timer being suspended, while EXP stands for the expiry of the timer. The *val* variable stands for the remaining time before the timer expires. These two variables are shared between a timer machine and the station machine which owns the timer machine.

Suspend stops the machine from counting down, and *Resume* allows a suspended machine to resume counting. Each machine modeling a station has a number of timer machines for various events in the protocol. It can alter the state of its timer machines by starting, suspending, or resuming them using the shared variables *st* and *val*.

4.1 External Variables and Assumptions

A boolean variable called *HaveData* is set true by a higher layer when there is data to be sent. In addition, the destination address of the current frame is available in *Address*. PCF_PERIOD (duration of the PCF period), SIFS, PIFS, ONE_POLL_T (time needed to send a poll and elicit a response), BEACON_PERIOD(time between beacons) are

Transition	Predicate	Action
Start	st=ACT	
Expired	val=0	st=EXP
Suspend	st=INACT	
Resume	st=ACT	
Decrement	st=ACT $\wedge$ val > 0	val=val - 1

Table 1. Predicate-Action Table for Generalized Timer

Abbreviation	Meaning
ACT	Active
EXP	Expired
INACT	Inactive
B-Timer	Backoff Timer
T-Timer	Timeout Timer
I-Timer	DIFS Timer
S-Timer	Short IFS Timer
R-Timer	Reservation Timer
STD	Value given by standard
BBusy	Busy during backoff

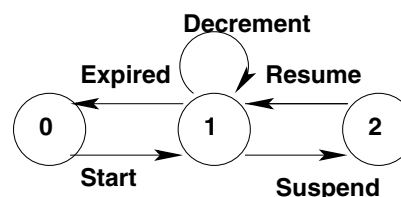**Table 2. Abbreviations used and their meanings**

Figure 2. Generalized Timer Machine

Figure 3. Shared variables

other variables which are set by entities outside of our specification. We model the medium as a shared variable with the following fields.

- Duration: In DCF, stores the remaining time in the current reservation. In PCF, it stores the remaining time in the PCF period.
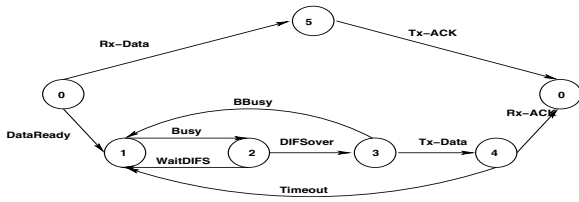- SA: stores the source address of the message on the

Figure 4. DCF State description

medium.

- DA: stores the destination address of the message on the medium.
- Beacon: set to 1 if the message on the medium is a *Beacon* message.
- RTS: set to 1 if the message on the medium is a RTS messages.
- CTS: set to 1 if the message on the medium is a CTS messages.

Transition	Predicate	Action
DataReady	HaveData = true	B-Timer.val := rand()
WaitDIFS	Medium = 0 ∨ Backoff = false	I-Timer.st := ACT I-Timer.val := STD
Busy	Medium ≠ 0	I-Timer.st := INACT Backoff := true
DIFS-Over	I-Timer.st = EXP	B-Timer.st := ACT
BBusy	medium ≠ 0	B-Timer.st := INACT Backoff=true
Tx-Data	B-Timer.st = EXP ∨ (medium = 0 ∧ Backoff=false)	Medium.DA = Address Medium.NullData = 0 T-Timer.val := STD T-Timer.st := ACT
Timeout	T-Timer.st = EXP	B-Timer.val := rand() Backoff := true
Rx-ACK	Medium.DA = i ∧ Medium.ACK = 1	Backoff := false
Rx-Data	Medium.DA=i	S-Timer.val := STD S.Timer.st := ACT
Tx-ACK	S-Timer.st = EXP	medium.ACK := 1 Medium.DA := Address Medium.SA := i

Table 3. Predicate-Action Table for Basic DCF

Variable	Range	Initial Value	Purpose
Backoff	boolean	false	True if backoff is enabled
HaveData	boolean	false	True if data is to be sent Set by external entity.
Medium		Empty	Wireless medium

Table 4. Variables in the DCF specification

- ToDS: set to 1 if the data message is from a station, set to zero otherwise.
- End: set to 1 if the message on the medium is an *CF_END* message, i.e. end of contention free period.
- Poll: set to 1 if the AP wants to poll a station.
- NullData: set to 1 if the message on the medium contains no data.
- Ack: set to 1 if the message on the medium contains an acknowledgment for a previously sent data message.

Some messages are broadcast messages, e.g., the *Beacon* messages. These messages have a destination address set to *BROADCAST*. We make the AP ensure that all entities hear the broadcast before it clears the medium. This is accomplished by using a shared array *AllHeard[0..NSTA- 1]* that has an entry for each station. The entry for station *i* is set to *true* by the station if this station heard the last broadcast by the AP. When all the entries of the *AllHeard* array are *true*, the AP clears the medium and the operation of the protocol continues. Figure 3 shows the *Medium and AllHeard* shared variables.

4.2 DCF State description

The DCF state machine at station i is specified in figure 4. The transitions for this state machine are outlined in table 3. Each station machine has the following timers local to it: I-Timer, R-Timer, S-Timer, and B-Timer. The meanings of these timers are explained in table 2. Each of these timer machines has two variables *st* and *val* as explained before. These are shared with the station machine which uses these timers. All variables are specified in table 4.

The machine starts in state 0. The *Rx-Data* transition is enabled to get to state 5 when a frame is received. The machine performs the SIFS wait in state 5. Once this timer expires, the *Tx-ACK* is enabled and the machine transmits the ACK frame and gets to state 0. When a machine has a DATA frame to send, *Dataready* is enabled, and it gets to state 1. The *WaitDIFS* is enabled in state 1 if backoff is false (which holds the first time it does carrier sensing) or if the medium is free, and the machine gets to state 2. If the medium were busy any time, *Busy* is enabled, *Backoff* is set true, and the machine reaches state 1. If the medium were free for a DIFS duration, *DIFS-over* is enabled, and the machine reaches state 3. Once in state 3, it performs the additional backoff wait if *Backoff* is true. The backoff timer is reduced for the time the medium was free. If the medium becomes busy, the backoff timer is frozen, and the machine gets back to state 1 using *BBusy*- standing for Busy during Backoff. When the backoff timer eventually expires, *Tx-Data* is enabled, and the machine gets to state 4. In state 4, the *Timeout* or the *Rx-ACK* transitions are enabled if there is a timeout or an ACK respectively.

The following points should be noted:

- The number of retransmissions is limited. Once this limit is exceeded, the data frame is discarded, and the machine returns to state 0. We do not show this transition in the state machine.
- The reception of an ACK for a frame being currently re-transmitted due to a timeout causes the system to declare the frame to be a success, and the machine returns to state 0. We also do not show these transitions in the state machine.
- The reception of a corrupt frame does not enable *Rx-data*. Therefore, the sending station eventually timesout and retransmits the frame for proper reception.

We now state two lemmas and subsequently theorems on freedom from deadlocks and non-executable transitions. All proofs have been removed for want of space and can be found in [6].

Lemma 1 *If a machine x gets to state 1, it gets to state 0 eventually.*

Lemma 2 *If a machine x gets to state 5, it gets to state 0 eventually.*

Theorem 1 *The CSMA/CA protocol as specified is free from deadlocks.*

Theorem 2 *The CSMA/CA protocol as specified is free from non-executable transitions.*

There is no *liveness* in this protocol. If the random backoff timers of two machines choose the same value, the two machines transmit at the same time causing collision. Even after backing off subsequent to the collision, the machines could transmit again at the same time. The collisions could potentially continue indefinitely, thereby implying lack of liveness in the protocol.

5   Formal Specification of MACA

We model each station implementing MACA as a state machine. The state machine for station i is shown in figures 5, 6, and 7. The enabling predicates are shown in table 5. The variables used are the same as in table 4. In addition, we have an R-Timer which counts when the medium has been reserved by another RTS, CTS, or data frame observed by this station.

Each machine starts at state 0. If the machine overhears an RTS/CTS/Data frame not meant for it, i.e., one that reserves the medium, the *Reserve* transition is enabled. It sets the *R-Timer* to the duration specified in the field of the frame, and enters state 8. Once this timer expires, the machine gets back to state 0. If it receives any further reservation information, it updates the waiting time and stays in state 8.
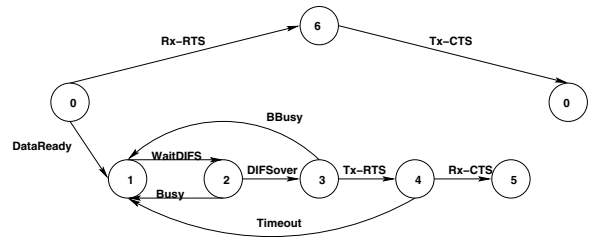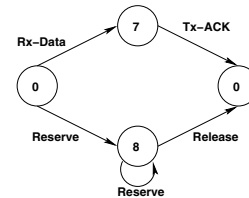

Figure 5. State machine for station i contd

Figure 6. State machine for station i contd.

If a station needs to send data, it needs to send the RTS frame first. The station gets to state 1. It performs the DIFS wait in states 1 and 2. It invokes the backoff procedure in state 3 if necessary. Eventually, it transmits the RTS frame and waits for an acknowledging CTS frame in state 4. If it receives the CTS, it can move to state 5, where it performs the SIFS wait. Once the SIFS is over, the station transmits the DATA frame and moves to state 9, where it waits for an ACK. The transitions *Timeout* and *ACKed* are enabled appropriately.

When a station receives data, *Rx-DATA* is enabled, and the machine gets to state 7. In state 7, it waits for an SIFS duration. At the end of the SIFS, *Tx-ACK* is enabled, and the machine gets to state 0. Similarly, when a machine receives an RTS meant for itself, it waits for an SIFS and proceeds to transmit the CTS.

We now proceed to state some lemmas and subsequently theorems on safety of the MACA protocol. All proofs have been removed for want of space and can be found in [6].

Lemma 3 *If a machine x gets to state 6, it gets to state 0 eventually.*

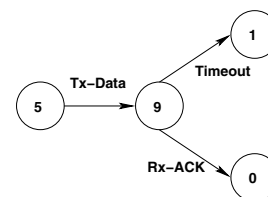**Lemma 4** *If a machine x gets to state 1, it gets to state 5*

Figure 7. State machine for station i

Transition	Predicate	Action
DataReady	HaveData = true	B-Timer.val = rand() Backoff=false
WaitDIFS	Medium = 0 ∨ Backoff=false	I-Timer.st=ACT I-Timer.val = STD
Busy	Medium ≠ 0	I-Timer.st=INACT
DIFS-Over	I-Timer.st = EXP	B-Timer.st=ACT
BBusy	Medium ≠ 0	B-Timer.st =INACT Backoff = true
Tx-RTS	B-Timer.st = EXP ∨ (Medium = 0 ∧ Backoff=false)	Medium.RTS = 1 Medium.DA = Address T-Timer.val = STD T-Timer.st = ACT
Timeout	T-Timer = EXP	B-Timer.val = rand() Backoff=true
Rx-CTS	Medium.DA = i ∧ Medium.CTS = 1	S-Timer.st=ACT
Rx-RTS	Medium.DA = i ∧ Medium.RTS=1	S-Timer.st=ACT
Tx-CTS	S-Timer.st=EXP	Medium.CTS = 1 Medium.DA = Address
Rx-Data	Medium.DA = i	S-Timer.st=ACT
Tx-ACK	S-Timer.st=EXP	Medium.ACK = 1 Medium.DA = Address
Reserve	Medium.DA != i ∧ (Medium.RTS=1 ∨ Medium.CTS =1 ∨ Medium.NullData =0)	R-timer.st= ACT R-Timer.val = max(R-Timer.val, Medium.Duration)
Release	R-Timer.st=EXP	
Tx-Data	S-Timer.st=EXP	Medium.DA = Address Medium.NullData = 0 T-Timer=ACT
Timeout	T-Timer.st=EXP	backoff=true
Rx-ACK	Medium.DA = i ∧ Medium.ACK=1	

Table 5. Predicate-Action Table for DCF with RTS/CTS: MACA

or state 0 eventually.

Lemma 5 *If a machine x gets to state 5, it gets to state 0 eventually.*

Lemma 6 *The MACA protocol has a potential livelock in state 8.*

Lemma 7 *If a machine gets to state 7, it gets to state 0 eventually.*

Theorem 3 *The MACA protocol as specified is free from deadlocks modulo lemma 6.*

Theorem 4 *The MACA protocol as specified is free from any non-executable transitions.*

6 Formal Specification of the PCF Protocol

There are three entities in the specification of the PCF protocol: the AP, the pollable stations, and the non-pollable stations. We have a total of *NSTA* stations of which *NPoll*, *NPoll* ≤ *NSTA*, are pollable stations and the remaining *NSTA- NPoll* stations are non-pollable. Without loss of generality, we assume the the pollable stations have the first *NPoll* addresses. The specifications of the PCF protocol consists of the global variables (Figure 3), local and timer variables for the AP (Table 6), the state machine and the predicate-action table for the AP (Figure 8 and Table 8), the local and timer variables for stations (Table 7), the state machine and the predicate-action table for the pollable stations (Figure 9 and Table 9), the state machine and the predicate-action table for the non-pollable stations (Figure 10 and Table 10), the state machines and predicate-action tables for the timers (Figure 11 and Table 11), and the initial state.

6.1 State description

For the AP, states 0 and 1 send a *Beacon*, and make sure that all stations heard it before the AP can proceed. In state 2, the AP loops through the set of pollable stations in a round robin fashion.

For the current pollable station, denoted by the variable *CurPSTA*, the AP sets the *Poll* field of the medium to 1 and checks if it has data to this station, by consulting the array *PData*. If this is the case, the AP sets the *NullData* flag in the medium to 0. If there is a pending acknowledgment, the variable *AckNeeded* is *true*, the AP sets the *Ack* field of the medium to 1. This acknowledgment is piggybacked and is not necessarily acknowledging a data packet for the polled station (making use of the fact that all stations will hear the message due to the shared medium). In all cases, the AP moves to state 4 waiting for a reply from the polled station. Note that the AP cannot, however, send a poll if the remaining time in the current contention free period is not enough for the polled station to send a data message with minimum length.

If the AP receives data in reply to its poll, it sets the *AckNeeded* flag to indicate that an acknowledgment is required. If the data message received is destined to a non-pollable station, it sets the *DataNPoll* to the address of the non-pollable station. If the data message received is destined to another pollable station, the AP sets the *PData* array entry for the pollable station to true, indicating the availability
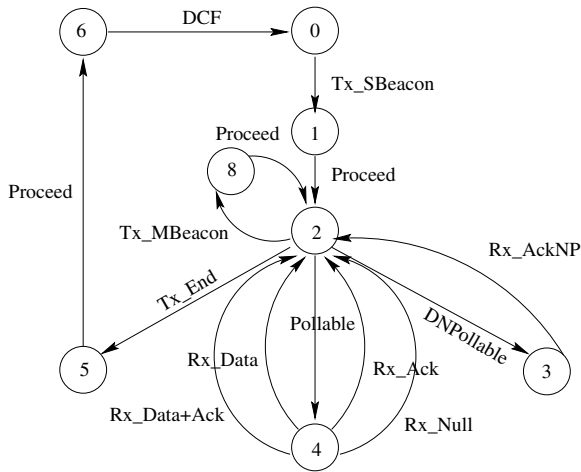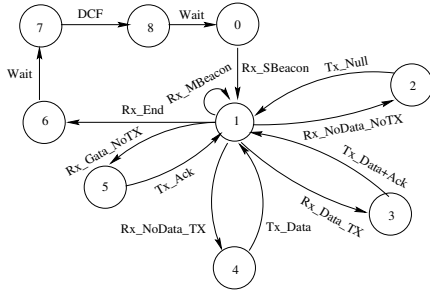
Figure 8. Specification for the AP

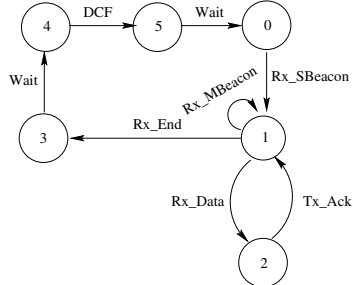**Figure 9. Specification for Pollable Station** i

Figure 10. Specification for non-Pollable Station i

Figure 11. Specification for a timer

Variable name	Range	Initial value	Purpose
STAType	array [0..NSTA- 1] of {pollable, non-pollable}		StationType[i] stores the type of the ith station
CurPSTA	0..NPoll- 1	0	current pollable station
PData	array [0..NPoll- 1] of integer	0	PData[i] is the number of messages buffered in the AP to the ith pollable station
DataNPoll	-1.. NSTA- 1	-1	address for a non-pollable station for which the AP has data to send, -1 if none
AckNeeded	boolean	false	true if the AP received a data packet that needs acknowledgment
IFS_T	integer	PIFS	Controls the IFS (Shared with the timer)
Remain_T	integer	PCF_PERIOD	Controls the superframe duration (Shared with the timer)
Beacon_T	integer	BEACON_T	Controls the beacon interval (Shared with the timer)

Table 6. Local, global, and timer variables specification for the access point

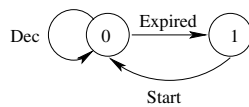| Variable name	Range	Initial value	Purpose
NAV	integer		Holds the remaining time of the current PCF period
HaveData	boolean		True if the station has data to send (set by an external entity)
Address	0.. NSTA- 1		Holds the destination address of the data to be transmitted (set by an external entity)
IFS_T	integer	PIFS	Controls the IFS (Shared with the timer)

Table 7. Local, global, and timer variables specification for stations

of data for this station. In all cases, the AP returns to state 2 after incrementing the *CurPSTA* variable using modulo *NPoll* arithmetic.

If the AP has data to a non-pollable station, it will send it to the station address, stored in the *DataNPoll* variable, and move to state 3 waiting for a reply before it continues

its normal operations.

Periodically, the AP sends *Beacon* messages. This helps stations in sleep mode to know the remaining time in the current contention free period.

At the end of the contention free period, the AP sends a *CF_END* message and moves to state 5 waiting for all stations to hear the *CF_END* message. After that, the DCF period starts and after it finishes, a new PCF period starts.

The operation of a pollable station is a response to the actions taken by the AP. When a station receives a *Beacon* message from the AP, it updates its *NAV* to the *Duration* value in the message on the medium. If the station has data to send (states 3 and 4) it sends the data and sets the acknowledgment field of the medium to 1 if the message received from the AP contained data. If the station does not have data to send and the message received from the AP contained data, then the station sends an acknowledgment back to the AP (state 5). If no data was received from the AP and the station does not have data to send, the station must send a *Null* message. A polled station can only send one data packet per poll.

If the station receives a *Beacon* message, it updates its *NAV* accordingly. When the station receives an *End* message from the AP, it moves to state 6 and waits there until all other stations have heard the same message. After that a DCF period starts and after it finishes, the station returns to state 0 waiting for the start of a new contention free period.

The operation of a non-pollable station is similar to the operation of a pollable station and can be found in [7].

The AP uses three types of timers:

- IFS_T: used to determine the period for the AP to wait before sending a message.
- Remain_T: used to determine the length of the PCF and DCF periods.
- Beacon_T: used to determine the beacon interval.

All three timers have identical state machines (shown in Figure 11). The counter is continuously decremented in state 0 until it reaches 0. In this case, the counter moves to state 1 where it waits for the AP to reset it again to return to the start state. The predicate-action table for the three timers is very similar. We include only one of them (Table 11) for space constraints.

The stations use the first 2 timers and their specifications are the same as those of the AP.

The initial state of the system is that all entities are in state 0, the medium is idle and the initial value for the other global and local variables are as shown in tables 6 and 7.

In the balance of this section, we show that the PCF protocol, as specified, is free from deadlocks and non-executable transitions. After stating the safety properties in section 6.2, liveness properties are discussed in section 6.3.

6.2 Safety properties

Lemma 8 *If the AP in in state* 2 *then all the other stations must be in state* 1.

Lemma 9 *If the AP is in state* 2, *then the medium must be idle.*

Lemma 10 *If the system is in the initial state, it will go to the state* $(1, 1, ..., 1, 2)$, *where the last entry represents the state of the AP and the remaining $NSTA$ entries of the tuple represent the state of the stations.*

Lemma 11 *If the system is in a state of the form*

$$(x_0, x_1, ..., x_{NPoll-1}, y_0, y_1, ..., y_{NSTA-NPoll-1}, 2)$$

where the protocol has $NPoll$ pollable stations, represented by x_i, and $NSTA - NPoll$ non-pollable stations, represented by y_i, and an AP, represented as the last element in the tuple, then the system will eventually leave this form but will return to this form in a finite number of steps.

Theorem 5 (Safety) *The PCF protocol as specified is free from deadlock states.*

Corollary 1 *The PCF protocol as specified is free from non-executable transitions.*

The corollary states that every transition is executable, i.e. there are no unreachable 'lines of code'. The corollary can be proved by working through the proofs of the lemmas, and noting that all transitions are executable at some point.

6.3 Liveness properties

Theorem 6 (Liveness) *Each pollable station will be polled in a finite number of superframes.*

All proofs can be found in [7] and have been removed for space constraints.

The liveness proof shows that if a station has data to send, it will get a chance to transmit after at most *NPoll* superframes. However, the length of both the PCF and DCF periods are variable, as shown in Figure 1 and the DCF period length may be unbounded due to collisions. The use of backoff timers and different inter-frame spacing priorities lessens these effects.

7 Conclusions and future work

In the DCF, the freedom from deadlocks is guaranteed by the wait-state. The MACA protocol has a potential live-lock in the ad-hoc mode as one station could potentially be

Transition	Enabling predicate	Action
Tx_SBeacon	$IFS\_T = 0 \land Medium = 0$	IFS_T= SIFS; Remain_T= PCF_PERIOD; Beacon_T= BEACON_PERIOD; Medium.(Beacon, ToDS, End, Poll, NullData, Ack, Duration, DA) ← (1, 0, 0, 0, 0, 0, Remain_T, BROADCAST);
Tx_End	Remain_T= 0 ∧ IFS_T= 0	Remain_T= DCF_PERIOD; Medium.(Beacon, ToDS, End, Poll, NullData, Ack, DA) ← (0, 0, 1, 1, 1, AckNeeded? 1: 0, BROADCAST); AckNeeded= false;
DCF	Remain_T= 0 ∧ Medium= 0 ∧ AllHeard[0..NSTA- 1]= true	IFS_T= PIFS; AllHeard[0..NSTA- 1]= false
Pollable	IFS_T= 0 ∧ (Remain_T- ONE_POLL_T) > 0 ∧ Beacon_T > 0 ∧ DataNPoll= -1	Medium.(Beacon, ToDS, End, Poll, NullData, Ack, DA) ← (0, 0, 0, 1, (PData[CurPSTA]=0)? 1: 0, AckNeeded? 1: 0, CurPSTA); AckNeeded= false; if (PData[CurPSTA] > 0) → PData[CurPSTA]- -;
Rx_Data+Ack	Medium.(ToDS, NullData, Ack)= (1, 0, 1)	AckNeeded= true; if STAType(DA)= pollable → PData[DA]++; else DataN-Poll= DA; CurPSTA= CurPSTA ⊕ 1; Medium= 0; IFS_T= SIFS;
Rx_Data	Medium.(ToDS, NullData, Ack)= (1, 0, 0)	AckNeeded= true; if STAType(DA)= pollable → PData[DA]++; else DataN-Poll= DA ; CurPSTA= CurPSTA ⊕ 1; Medium= 0; IFS_T= SIFS;
Rx_Ack	Medium.(ToDS, NullData, Ack)= (1, 1, 1)	CurPSTA= CurPSTA ⊕ 1; Medium= 0; IFS_T= SIFS;
Rx_Null	Medium.(ToDS, NullData, Ack)= (1, 1, 0)	CurPSTA= CurPSTA ⊕ 1; Medium= 0; IFS_T= SIFS;
DNPollable	IFS_T= 0 ∧ DataNPoll ≠ -1 ∧ Beacon_T > 0 ∧ (Remain_T- ONE_POLL_T) > 0	Medium.(Beacon, ToDS, End, Poll, NullData, Ack, DA) ← (0, 0, 0, 0, 0, AckNeeded? 1: 0, DNPollable); (DNPollable, AckNeeded)= (-1, 0)
Rx_AckNP	$Medium.(ToDS, Ack) = (1, 1)$	DataNPoll= -1; Medium= 0; IFS_T= SIFS;
Tx_MBeacon	$Beacon\_T = 0 \land IFS\_T = 0$	Beacon_T= BEACON_PERIOD ∧ Remain_T ≠ 0; Medium.(Beacon, ToDS, End, Poll, NullData, Ack, Duration, DA) ← (1, 0, 0, 0, 0, 0, Remain_T, BROADCAST); IFS_T= SIFS;
Proceed	AllHeard[0..NSTA- 1]= true	Medium= 0; AllHeard[0..NSTA- 1]= false;

Table 8. Predicate action table for the AP

starved. Liveness is not guaranteed in either due to possibility of continued collisions.

PCF deadlock freedom is due to the synchronous operation betweent the AP and the stations. The liveness property is guaranteed because of the round robin scheduling mechanism at the AP and the minimum length of the contention free period. However, the length of both the PCF and DCF periods are variable, as shown in Figure 1 and the DCF period length may be unbounded due to collisions.

Directions for future work include relaxing some of our assumptions, association/dissociation protocols in the MAC, and modeling an entire ESS, among others.

References

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

[2] P.Karn. MACA: A New Channel Access Method for Packet Radio. *Proceedings of the ARRL/CRRL Amateur Radio 9th Computer Networking Conference*, September 1990.

[3] G. M. Lundy. *Systems of Communicating Machines: A Model for Communicatioon Protocols*. Ph.D. dissertation, School of Information and Computer Science, Georgia Institute of Technology, Atlanta, GA, 1988.

[4] G. M. Lundy and Raymond E. Miller. Analyzing a CSMA/CD Protocol through a Systems of Communicating Machines Specification. *IEEE Transactions on Communications*, 41(3):447–449, March 1993.

[5] G.M.Lundy and I.F.Akyildiz. Specification and analysis of the FDDI MAC protocol using systems of communicating machines. *Computer Communications*, 15, No.5:285–294, 1992.

[6] Arunchandar Vasan and Raymond E. Miller. Specification and Analysis of the DCF Protocol in the 802.11 Standard using Systems of Communicating Machines. Technical Report UMIACS-TR 2002-37 and CS-TR 4358, University of Maryland, College Park, May 2002. http://www.cs.umd.edu/Library/TRs/.

[7] Moustafa A. Youssef and Raymond E. Miller. Analyzing the Point Coordination Function of the IEEE 802.11 WLAN Protocol using a Systems of Communicating

Transition	Enabling predicate	Action
Rx_SBeacon	$Medium.Beacon = 1$	NAV= Medium.Duration; AllHeard[i]= true
Rx_End	$Medium.End = 1$	Remain_T= DCF_PERIOD; NAV= 0; AllHeard[i]= true
DCF	Remain_T= 0 ∧ Medium= 0	AllHeard[i]= true
Rx_Data_TX	Medium.(ToDS, Poll, NullData, DA) = (0, 1, 0, i) ∧ HaveData	IFS_T= SIFS
Tx_Data+Ack	IFS_T= 0	Medium.(Beacon, ToDS, End, Poll, NullData, Ack)= (0, 1, 0, 0, 0, 1); HaveData= false
Rx_NoData_TX	Medium.(ToDS, Poll, NullData, DA) = (0, 1, 1, i) ∧ HaveData	IFS_T= SIFS
Tx_Data	IFS_T= 0	Medium.(Beacon, ToDS, End, Poll, NullData, Ack, DA)= (0, 1, 0, 0, 0, 0, Address); HaveData= false;
Rx_Data_NoTX	Medium.(ToDS, Poll, NullData, DA) = (0, 0, 0, i) ∨ (Medium.(ToDS, Poll, NullData, DA) = (0, 1, 0, i) ∧ not HaveData)	IFS_T= SIFS
Tx_Ack	IFS_T= 0	Medium.(Beacon, ToDS, End, Poll, NullData, Ack, DA)= (0, 1, 0, 0, 1, 1, AP)
Rx_NoData_NoTX	Medium.(ToDS, Poll, NullData, DA) = (0, 1, 1, i) ∧ not HaveData	IFS_T= SIFS
Tx_Null	IFS_T= 0	Medium.(Beacon, ToDS, End, Poll, NullData, Ack, DA)= (0, 1, 0, 0, 1, 0, AP)
Rx_MBeacon	$Medium.Beacon = 1 \land AllHeard[i] = false$	NAV= Medium.Duration; AllHeard[i]= true;
Wait	AllHeard[i]= false	

Table 9. Predicate action table for a pollable station i

Transition	Enabling predicate	Action
Rx_SBeacon	$Medium.Beacon = 1$	NAV= Medium.Duration; AllHeard[i]= true
Rx_End	$Medium.End = 1$	Remain_T= DCF_PERIOD; NAV= 0; AllHeard[i]= true
DCF	Remain_T= 0 ∧ Medium= 0	AllHeard[i]= true
Rx_Data	Medium.(ToDS, Poll, NullData, DA)= (0, 0, 0, i)	IFS_T= SIFS
Tx_Ack	$IFS\_T = 0$	Medium.(Beacon, ToDS, End, Poll, NullData, Ack)= (0, 1, 0, 0, 1, 1)
Rx_MBeacon	$Medium.Beacon = 1 \land AllHeard[i] = false$	NAV= Medium.Duration; AllHeard[i]= true
Wait	AllHeard[i]= false	

Table 10. Predicate action table for a non-pollable station i

Transition	Enabling predicate	Action
Dec	$IFS\_T > 0$	IFS_T= IFS_T- 1
Expired	$IFS\_T = 0$	
Start	$IFS\_T > 0$	

Table 11. Predicate action table for the IFS Timer

Machines Specification. Technical Report UMIACS-TR 2002-36 and CS-TR 4357, University of Maryland, College Park, May 2002. http://www.cs.umd.edu/Library/TRs/.