Analyzing the Point Coordination Function of the IEEE 802.11 WLAN Protocol using a Systems of Communicating Machines Specification

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

CS-TR-4357 and UMIACS-TR-2002-36
May 2002

Abstract

A model for the specification and analysis of communication protocols called *Systems of Communicating Machines* is used to specify an IEEE 802.11 Point Coordination Function protocol, and to analyze it for safety and certain restricted liveness properties. The model uses a combination of finite state machines and variables in the specification of each machine, and the communication between machines is accomplished through shared variables. Enabling predicates and actions are associated with each transition; the enabling predicates determine when a transition may be taken, and the actions alter the variable values as the network progresses.

One of the advantages which this model has over most other formal description techniques is that simultaneous transitions are allowed. Another advantage is the use of shared variables rather than FIFO queues for communication between machines. This allows the modeling of the shared medium as a single shared variable variable between all communicating processes.

Unlike the SDL language which is used in the specification of the 802.11 standard, the Systems of Communicating Machines model is more compact, easier to understand, and easier to analyze for safety and liveness.

1 Introduction

Wireless communications are an emerging technology and are becoming an essential feature of everyday's life. Eventually, each device will have a wireless interface (e.g. laptops, cameras, phones etc.). The need in wireless access to local networks grows with the number mobile devices such as notebooks and PDAs, as well as with the desire users to be connected to a network without dealing with a network cable. According to forecasts [1], in 2003 there will be more than a billion of mobile devices, and the wireless LAN (WLAN) market is estimated to be more than two billion of dollars by 2002. The

IEEE 802.11 WLAN standard is the most widely used WLAN standard today. The IEEE 802.11 standard covers the MAC (Medium Access Control) sub-layer and the physical layer of the OSI (Open System Interconnection) reference model. The IEEE 802.11 standard provides for three variations of the physical layer. These include Direct Sequence Spread Spectrum (DSSS), Frequency Hopped Spread Spectrum (FHSS), and Infrared (IR). In practice, only the first two, DSSS and FHSS, have any significant presence in the market. The DSSS and FHSS PHY options were designed to operate in the 2.4GHz ISM (Industrial, Scientific and Medical) band. The 2.4GHz ISM band is particularly attractive because it enjoys worldwide allocations for unlicensed operation.

The MAC layer supports two services: Distributed Coordination Function (DCF) and Point Coordination Function (PCF). The DCF is an asynchronous data transmission function, which best suits delay insensitive data (e.g. email, ftp). It is the only possible function in ad-hoc networks, and can be either exclusive or combined with PCF when used in an infrastructure network (equipped with an AP). The PCF best suits delay sensitive data transmissions (e.g. real-time audio/video).

In this paper, we provide a formal model of the PCF service of the MAC protocol using a Systems of Communication Machines [2] specification and analyze it for safety and liveness properties. Unlike the original standard which is formalized using the SDL-92 language, a Systems of Communication Machines specification is more compact, easier to understand, and easier to analyze for safety and liveness.

The balance of this section gives an introduction to the IEEE 802.11 WLAN protocol and the PCF service protocol. In section 2 the definition of the Systems of Communication Machines model is given. Section 3 provides the Systems of Communication Machines model for the PCF protocol. The analysis of the protocol is given in section 4. Finally, the conclusions and future work are given in section 5.

1.1 IEEE 802.11 Standard

The 802.11 protocol [3, 4] works on two lower layers of the ISO/OSI model: the physical layer and the MAC layer(Figure 1). Any network applications,

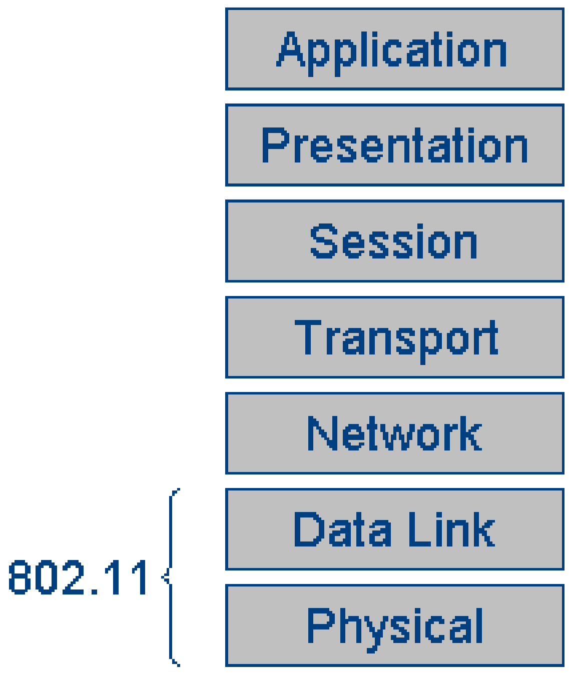

Figure 1: Levels of the ISO/OSI model and their correspondence with the 802.11 standard.

network operating system or protocol (e.g., TCP/IP) will work in the 802.11 network.

1.1.1 Operating modes

The 802.11 deals with two types of equipment - a client which is a computer equipped with a wireless Network Interface Card (NIC), and an Access point (AP) which serves as a bridge between a wireless and a wired network. An access point usually contains a transceiver, a wired network interface (802.3) and software for data processing.

The IEEE 802.11 standard has two operating modes of a network: point-to-point (Independent) mode and infrastructure mode. In the independent mode (Figure 2) a wireless network consists of at least one access point connected to a wired network, and a set of wireless terminal stations. Such combinations are called Basic Service Set (BSS). Two or more BSSs which make a single subnetwork form an Extended Service Set (ESS). Since most of wireless stations require an access to file servers, printers, and the Internet available in a wired LAN, they will work in the infrastructure mode.

The Independent mode (referred to as the Independent BSS, IBSS) is a

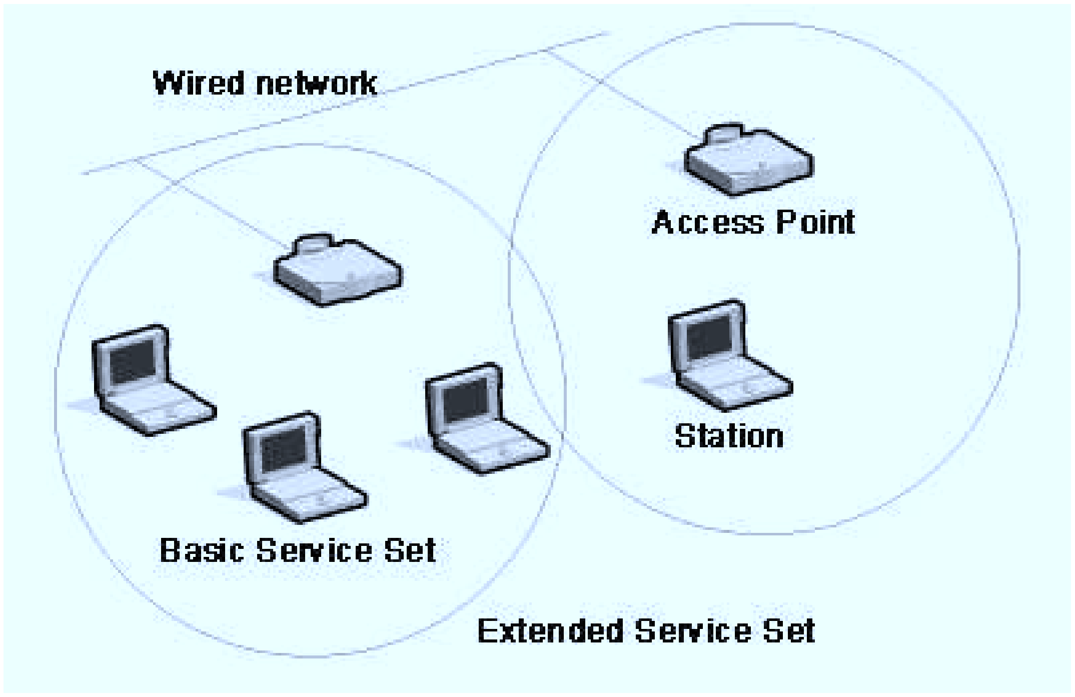

Figure 2: Architecture of the ESS mode.

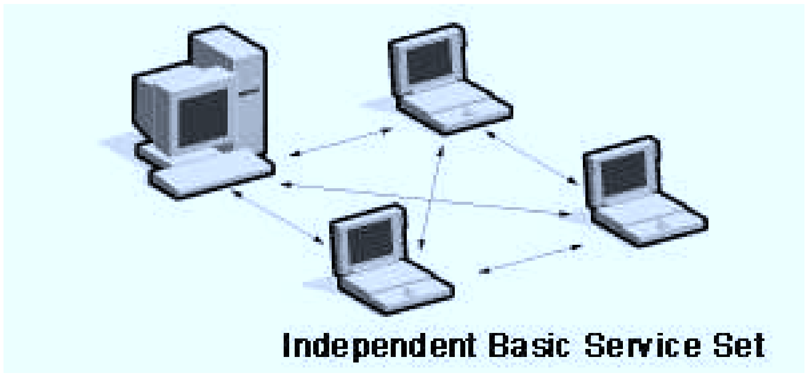

Figure 3: Architecture of the IBSS mode.

simple network where stations are connected directly, without a special access point (Figure 3).

1.1.2 Physical layer

The physical layer deals with two wide-band radio-frequency transfer methods and one in the infrared range. The radio frequency methods, which are the practical ones, work in the ISM at 2.4 GHz.

The 802.11 standard uses the Direct Sequence Spread Spectrum method and Frequency Hopping Spread Spectrum method. These methods are incompatible.

In the FHSS method the 2.4 GHz band is divided into 79 channels of 1 MHz. The sender and receiver match the hop pattern of switching the channels and data are transferred successively in different channels according to the chosen pattern. Each data transfer process in the 802.11 network is implemented according to a different hop pattern, and the hop patterns are developed so as to minimize the probability of usage of the same channel simultaneously by two senders. The FHSS method allows a very simple construction of a transceiver, but it is limited to 2 Mbps speed.

The DSSS method divides the 2.4 GHz range into 14 overlapping channels (there are 11 channels available in the USA). Each channel has a bandwidth of about 20 MHz regardless of the data rate. This means that only three channels can be used at one place. Data are transferred in one of these channels without switching to others. To compensate the unwanted noise a Barker's 11-bit sequence is used: each data bit of a user is converted into 11 bits of data for transmission. Such high redundancy for each bit increases the reliability of transmission considerably, with the power of a signal transferred being much lower. Even if a part of a signal is lost, it will be recovered, in most of the cases. This minimizes the number of repeated data transmissions.

Details about the IR method can be found in [3].

1.1.3 MAC layer

The MAC layer of the IEEE 802.11 protocol supports two services: Distributed Coordination Function (DCF) and Point Coordination Function (PCF). DCF is the basic medium access in IEEE 802.11. It is part of the family of CSMA/CA-based medium access protocols. Transmissions are separated by inter packet gaps known as Inter Frame Spaces (IFS). Channel access is granted based on different priority classes. These classes are mapped on different gap durations: Distributed-IFS (DIFS, also known as DCF inter-frame space), Priority-IFS (PIFS, also known as PCF inter-frame spacing), and Short-IFS (SIFS), with SIFS being the shortest (highest priority) and DIFS being the longest (lowest priority). Stations (STAs) sense the medium to determine if it is idle. If so the STA may transmit. However if it is busy, each STA waits until transmission stops, and then enters into a random back off procedure. This prevents multiple STAs from seizing the medium immediately after completion of the preceding transmission. Packet reception in DCF requires acknowledgment. The period between completion of packet transmission and start of the ACK frame is one SIFS. Fast acknowledgment is one of the salient features of the 802.11 standard, because it requires ACKs to be handled at the MAC sublayer. Transmissions other than ACKs must wait at least a DIFS before transmitting data. If a transmitter senses a busy medium, it determines a random back-off period by setting an internal timer. After medium becomes idle, STAs wishing to transmit wait a DIFS plus an integer number of Slot Times depending on the timer setting (0 to 7 on first attempt). Upon expiration of a DIFS, the timer begins to decrement. If the timer reaches zero, the STA may begin transmission. However, if the channel is seized by another STA before the timer reaches zero, the timer setting is retained at the decremented value for subsequent transmission.

If a transmission is not acknowledged, the packet may not have been received due to a collision. On a second attempt, the random back-off window is increased to 15 Slot Times. The window is doubled on each successive attempt up to a maximum value of 256 Slot Times. Unacknowledged packets may be due to a failure in reception of the packet, or in failure to receive an ACK. Either case is indistinguishable to the sending STA and the same retransmission procedure is followed. The method described above relies on the ability of each STA to sense signals from all other STAs within the BSS. This approach is referred to as Physical Carrier Sense. The underlying assumption that every STA can hear all other STAs is not always valid. Consider, for example, Figure 4; nodes A and C cannot hear each others transmission, however, both will collide at station C. This is know as the *Hidden Node problem.* In order to combat this problem, a second carrier sense mechanism, Virtual Carrier Sense, is described in the standard. Virtual Carrier Sense is implemented by reserving the medium for a specified period of time for an impending transmission. This is most commonly achieved by use of RTS/CTS frames. The RTS frame contains

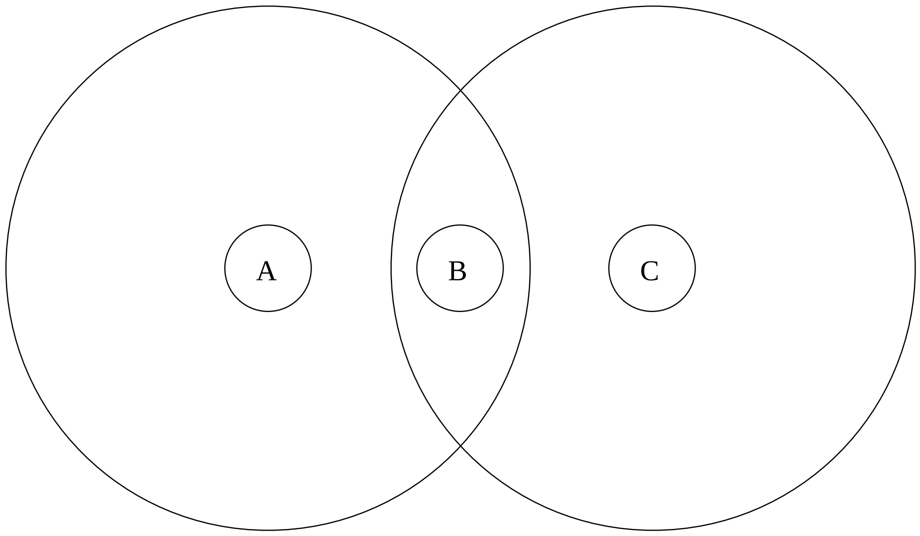

Figure 4: Hidden Node Problem.

a duration/ID field which specifies a period of time for which the medium is reserved for a subsequent transmission. The reservation information is stored in the Network Allocation Vector (NAV) of all STAs detecting the RTS frame. Upon receipt of the RTS, the AP responds with a CTS frame, which also contains a duration/ID field specifying the period of time for which the medium is reserved. All stations that did not hear the RTS frame will detect the CTS and update its NAV accordingly. Thus, collision is avoided even though some nodes are hidden from other STAs. The RTS/CTS procedure is invoked according to a user specified parameter. It can be used always, never, or for packets which exceed an arbitrarily defined length.

As mentioned above, DCF is the basic media access control method for 802.11 and it is mandatory for all STAs. An optional extension to DCF is the Point Coordination Function (PCF). PCF works in conjunction with DCF as shown in Figure 5. PCF was included specifically to accommodate time bounded connection-oriented services such as cordless telephony. Details about the PCF protocol is given in the next section.

The authors of the 802.11 standard allowed for the possibility that the wireless media, distribution system, and wired LAN infrastructure would all use different address spaces. IEEE 802.11 only specifies addressing for over the wireless medium, though it was intended specifically to facilitate integration with IEEE 802.3 wired Ethernet LANs. IEEE 802 48-bit addressing scheme was therefore adopted for 802.11, thereby maintaining address compatibility with the entire family of IEEE 802 standards. In the vast majority of installations, the distribution system is an IEEE 802 wired LAN and all three logical addressing spaces are identical.

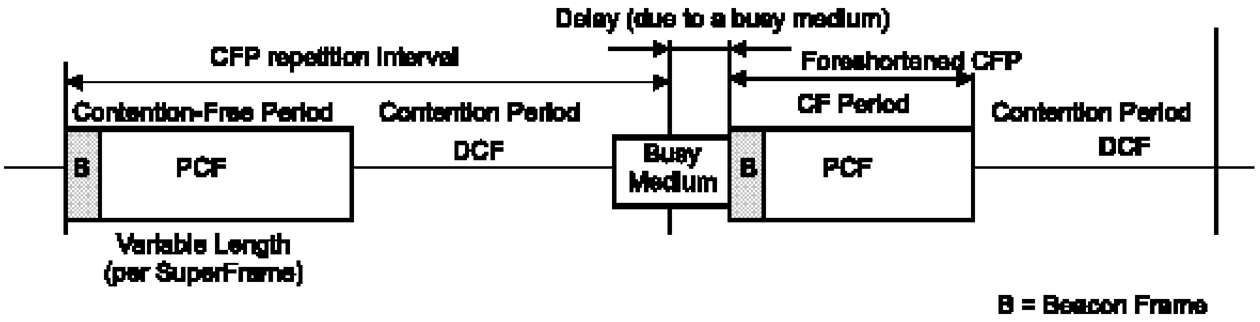

Figure 5: DCF/PCF Interaction.

1.2 PCF Protocol

Point Coordinator Function (PCF) is a centralized, polling-based access mechanism which requires the presence of a base station that acts as Point Coordinator (PC). If PCF is to be used, time is divided into superframes where 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 base station, using the ordinary DCF access method. Therefore, the CFP may be shortened since the base station has to contend for the medium. During the CFP, the PC polls each station in its polling list (the high priority stations), when they are clear to access the medium. To ensure that no DCF stations are able to interrupt this mode of operation, the interframe space between PCF data frames (PIFS) is shorter than the usual IFS (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.

In PCF mode the access point uses a Round-Robin scheduler to poll stations in the wireless cell. Stations that have been polled shall always respond to a poll. If there is no pending transmission, the response shall be a null frame containing no payload. If the CFP terminates before all stations have been polled, the polling list will be resumed at the next station in the following CFP cycle. A typical medium access sequence during PCF is shown in Figure 6. A station being polled is allowed to transmit a data frame. In case of an unsuccessful transmission the station may retransmit the frame after being repolled or during the next Contention Period.

1.2.1 Superframe structure

The access point controls the actual medium access scheme using a superframe structure as shown in Figure 6. the CFP repetition rate and the length of the CFP should be determined according to the characteristics of the time-bounded traffic that has to be conveyed. The value of *CFPMaxDuration* shall

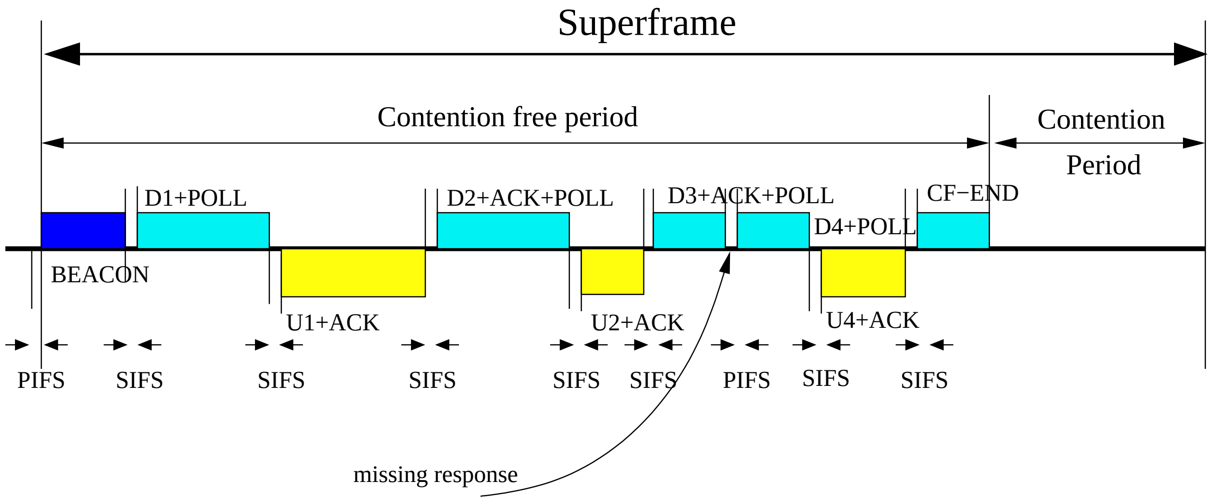

Figure 6: Point Coordination Function

be limited to allow the coexistence of contention and contention-free traffic and must be limited to provide sufficient time to send at least one data frame during the contention period (CP). Two problems may arise within the 802.11 frame structure:

1. A foreshortened CFP may occur after a CP period if the access point is prevented from accessing the channel due to a busy medium. This results in delayed transmissions of data frames during CFP causing additional delay. A CFP is foreshortened by the amount of delay caused by the preceding transmission in the CP.
2. A polled station may transmit frames of any length between 0 and 2312 bytes. At the beginning of a PCF cycle the total amount of bytes to be transmitted by the stations is not known. Due to the variable payload and duration needed for transmission, the AP may fail to poll all stations in the polling list during one cycle. Stations that have not been polled must postpone their frames queued for transmission to the next CFP causing an additional delay penalty.

In [5], they showed that the PCF function should be used for high load scenarios and to reduce contention in WLAN cells containing a large number of mobiles. They also showed that that the PCF enhances the network capacity. The DCF works well under low load conditions, but suffers from significant throughput degradation in high load conditions. Waste of bandwidth is caused by the increasing time used for negotiating channel access.

2 Systems of Communicating Machines

In this section, the model used to specify the protocol is described. A more detailed description appears in [2] and [6] .

A system of communicating machines is an ordered pair $C = (M, V)$, where

$$M = m_1, m_2, ..., m_n$$

is a finite set of machines, and

$$V = v_1, v_2, ..., v_k$$

is a finite set of shared variables, with two designated subsets R_i and W_i specified for each machine m_i. The subset R_i of V is called the set of *read* access variables for machine m_i, and the subset W_i is called the *write* access variables for m_i.

Each machine $m_i \in M$ is defined by a tuple $(S_i, s, L_i, N_i, \tau_i)$, where

1. S_i is a finite set of states;
2. $s \in S_i$ is a designated state called the *initial state* of m_i;
3. L_i is a finite set of *local variables*;
4. N_i is a finite set of names, each of which is associated with a unique pair (p, a), where p is a predicate on the variables of $L_i \cup R_i$, and a is an *action* on the variables of $L_i \cup R_i \cup W_i$. Specifically, an action is a partial function

   $$a : L_i \times R_i \rightarrow L_i \times W_i$$

   from the values of the local variables and read access variables to the values of the local variables and write access variables.
5. $\tau_i : S_i \times N_i \rightarrow S_i$ is a transition function, which is a partial function from the states and names of m_i to the states of m_i.

Machines model the entities, which in a protocol system are processes and channels. The shared variables are the means of communications between the machines. Intuitively, R_i and W_i are the subsets of V to which m_i has read and write access, respectively. A machine is allowed to make a transition from one state to another when the predicate associated with the name of that transition is true. Upon taking the transition, the action associated with that name is executed. The action changes the values of local and/or shared variables, thus allowing other predicates to become true.

The sets of local and shared variables specify a name and range for each. In most cases, the range will be a finite or countable set of values. For proper operation, the initial values of some or all of the variables should be specified.

A *system state tuple* is a tuple of all machine states. That is, if (M, V) is a system of n communicating machines, and s_i, for $1 \leq i \leq n$, is the state of machine m_i, then the n-tuple $(s_1, s_2, ..., s_n)$ is the system state tuple of (M, V). A *system state* is a system state tuple, plus the outgoing transitions which are enabled. Thus two system states are equal if every machine is in the same state, and the same outgoing transitions are enabled.

The *global state* of a system consists of the system state tuple, plus the values of all variables, both local and shared. It may be written as a larger tuple, combining the system state tuple with the values of the variables. The *initial global state* is the initial system state tuple, with the additional requirement that all variables have their initial values. The *initial system state* is the system state such that every machine is in its initial state, and the outgoing transitions are the same as in the initial global state.

A global state corresponds to a system state if every machine is in the same state, and the same outgoing state transitions are enabled. Clearly, more than one global state may correspond to the same system state.

Let $\tau(s_1, n) = s_2$ be a transition which is defined on machine m_i. Transition τ is *enabled* if the enabling predicate p, associated with the name n is true. Transition τ may be executed whenever m_i is in state s_1 and predicate p is true (enabled). The *execution* of τ is an atomic action, in which both the state change and the action a associated with n occur simultaneously.

3 Specification of the PCF Protocol

In this section, the PCF protocol is specified. In 3.1, we present the basic assumptions of the model. In 3.2, we describe how we model the shared medium. In 3.3, we provide the specifications of the three main entities of the PCF protocol: the access point, the pollable stations, and the non-pollable stations.

3.1 Assumptions

The following are assumed:

- We have a total of *NSTA* stations of which *NPoll*, *NPoll* $\leq$ *NSTA*, are pollable stationsand the remaining *NSTA- NPoll* stations are non-pollable.
- The address of a station is its number, i.e. the range of valid addresses is *0.. NSTA- 1* plus the special address *BROADCAST* that is used for broadcast messages sent from the access point.
- Stations can only send unicast messages, i.e. no multicast or broadcast messages are allowed from stations.
- The environment is error free, i.e. there are no communication errors, all stations conform to the specifications, no addresses are used that are outside the valid range, etc.
- There is only one point coordinator (i.e. one access point working in the BSS mode). This removes the case of multiple overlapping access points.

- We consider only one type of the management frames, which is the *Beacon* frame used for the timing of the PCF protocol.
- The time variables PCF_PERIOD (duration for the PCF period), DCF_PERIOD (duration for the DCF period), SIFS, PIFS, ONE_POLL_T (time needed to send a poll to a station and for the station to send the minimum pay load), and BEACON_PERIOD (time between beacons) are set by an external entity to the specification.
- The association functions used to add a station to the polling list of an access point is not handled. That is, an external entity will determine the state of each station.
- We do not model buffering of data at the access point. The model only captures the number of data packets buffered for each station.

3.2 Modeling the 802.11 Shared Medium

The physical communication medium of the 802.11 network is shared between all stations. Some abstraction from reality is necessary. We are not attempting to model the electrical signals on the physical medium. We model the medium as a shared global variable between all entities. The medium variable has the following fields (shown in Figure 7):

- Beacon: A flag set to 1 if the message on the medium is a *Beacon* message.
- ToDS: A flag set to 1 if the data message is from a station, set to zero otherwise.
- End: A flag set to 1 if the message on the medium is an *CF_END* message, i.e. end of contention free period.
- Poll: A flag set to 1 if the access point wants to poll a station.
- NullData: A flag set to 1 if the message on the medium contains no data.
- Ack: A flag set to 1 if the message on the medium contains an acknowledgment for a previously sent data message.
- Duration: A field to store the remaining time in the PCF period.
- DA: A field to store the destination address of the message on the medium.

In actual 802.11 networks, some messages are broadcast messages, e.g. the *Beacon* messages, that should be heard by all the stations. In the specification, these messages are denoted as messages that have a destination address set to *BROADCAST*. In order to model this situation, we make the access point insure that all entities hear the broadcast message 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 message sent by the access point. When all the

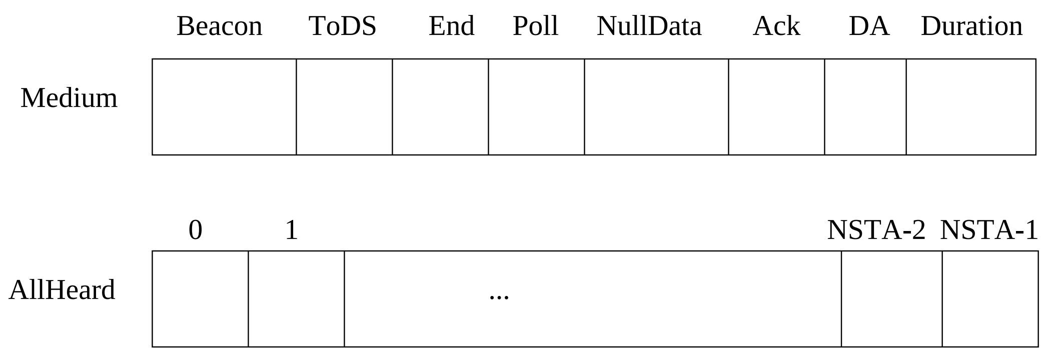

Figure 7: Shared variables

entries of the *AllHeard* array are *true*, the access point can clear the medium and the operation of the protocol continues. Figure 7 shows the *Medium and AllHeard* shared variables.

We could have another entity, for example a controller as in [7], that is responsible for controlling the shared medium. However, we chose to use the access point as this simplifies the specification.

3.3 Formal specification

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 7), local and timer variables for the access point (Table 1), the state machine and the predicate-action table for the access point (Figure 8 and Table 3), the local and timer variables for stations (Table 2), the state machine and the predicate-action table for the pollable stations (Figure 9 and Table 4),the state machine and the predicate-action table for the non-pollable stations (Figure 10 and Table 5), the state machines and predicate-action tables for the timers (Figure 11 and Tables 6 through 8), and the initial state.

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

For the current pollable station, denoted by the variable *CurPSTA*, the access point 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 access point sets the *NullData* flag in the medium to 0, otherwise, it is set to 1. If there is a pending acknowledgment, the variable *AckNeeded* is *true*, the access point sets the *Ack* field of the medium to 1, otherwise, it is set to 0. 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 access point moves to state 4 waiting for a reply from the polled station, which is mandated by the 802.11 standard to differentiate between a "no-traffic" situation and error situations, such as collisions with other overlapping 802.11 networks. Note that the access point 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 access point 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 (this is the only data interaction with a non-pollable station in the contention free period). If the data message received is destined to another pollable station, the access point stores the data in a buffer for this station (not explicitly modeled in this specification) and sets the *PData* array entry for the pollable station to true, indicating the availability of data for this station. In all cases, the access point returns to state 2 after incrementing the *CurPSTA* variable using modulo *NPoll* arithmetic.

If the access point 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 access point sends *Beacon* messages to other stations containing the remaining time till the end of the contention free period (the *MBeacon* transition). This helps stations that were in sleep mode to know the remaining time in the current contention free period.

At the end of the contention free period, the access point 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 the access point moves to state 6. After the end of the DCF period, a new PCF period starts.

The operation of a pollable station is a response to the actions taken by the access point. When a station receives a *Beacon* message from the access point, it updates its *NAV* to the *Duration* value in the message on the medium. This ensures that the station will not try to access the medium improperly until the end of the contention free period. 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 access point contained data, otherwise the acknowledgment flag is set to 0. If the station does not have data to send and the message received from the access point contained data, then the station sends an acknowledgment back to the access point (state 5). If no data was received from the access point and the station does not have data to send, the station must send a *Null* message. As specified by the standard, a polled station can only send one data packet per poll.

If the station receives a periodic *Beacon* message, it updates its *NAV* accordingly. When the station receives an *End* message from the access point, 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.

For a non-pollable station, when it receives a *Beacon* message from the access point, it updates its *NAV* to the *Duration* value in the message on the medium. The only data interaction for a non-pollable station in the contention free period is receiving and acknowledging messages sent by the access point. This is accomplished in states 1 and 2. If the station receives a periodic *Beacon* message, it updates its *NAV* accordingly. When the station receives an *End* message from the access point, it moves to state 3 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 access point uses three types of timers:

- IFS_T: used to determine the period for the access point to wait before sending a message (inter-frame spacing).
- 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 access point to reset it again to return to the start state.

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

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 1 and 2.

4 Analysis of the PCF Protocol

Analysis of the protocol is done in order to prove that the protocol has certain desirable properties of correctness. These correctness properties for protocols can be grouped into two classes: *safety* properties and *liveness* properties. Safety properties guarantee that the communication is free from some types of errors; liveness properties, also called progress properties, guarantee that certain positive steps will occur. An example of a safety property is freedom from deadlocks. A liveness property might state that "if a station has a data frame to send, it will send it in a finite time".

A well known analysis method for communication protocols called *reachability analysis*, has often been used with the *communicating finite state machine* model [8] to prove safety properties. Three of these safety properties are freedom from three types of errors: *deadlocks, unspecified receptions, and nonexecutable transitions.* A deadlock occurs when all machines in the network reach a state which they cannot transition out of, which is not the designated final state. An unspecified reception occurs when a machine has placed a message in the communication channel to another machine, and the destination machine is not able to remove the message from the channel. a nonexecutable transition is one which is specified but can never occur.

In this section, we show that the PCF protocol, as specified, is free from deadlocks and nonexecutable transitions. The unspecified reception error condition is defined for the CFSM model and is not clearly defined for the model of this paper, because of the differences in the two models [9]. After proving the safety properties in section 4.1, liveness properties are discussed in section 4.2.

4.1 Safety properties

Lemma 1 *If the access point in in state* 2 *then all the other stations must be in state* 1.

Proof By contradiction: Assume the lemma is not true, then there exists at least one station whose state is not 1 and the access point state is 2. Take one of these stations. We have 2 cases:

1. This station is a pollable station:

   The station cannot be in state 2 through 5 as the only way to enter these states is through 1 by receiving a poll from the access point. The access point can only send a poll in state 2, when the transition *Pollable* is enabled, thus moving to state 4. In order for the access point to leave state 4 and return to 2, the station must leave its state returning to 1, which contradicts our assumption that the station is in one of the states 2 through 5 while the access point is in state 2.

   The station cannot be in state 6 either as the only way to enter this states is through 1 by receiving an *End* message from the access point. The access point can only send an *End* message in state 2, when the transition *Tx_End* is enabled, thus moving to state 5. In order for the access point to return to state 2, the transition path *(Proceed, DCF, Tx_SBeacon, Proceed)* must be taken. This can only happen when all the pollable stations, including this station, take the transition path *(Wait, DCF, Wait, Rx_SBeacon)*. This can only happen by the station leaving state 6 and ending in state 1, which contradicts our assumption that the station is in state 6 while the access point is in state 2.

   Using similar arguments, the station cannot be in states 7 or 0 while the access point is in state 2.

2. This station is a non-pollable station:

   The station cannot be in state 2 as the only way to enter this state is through 1 by receiving a *Data* message from the access point. The access point can only send a *Data* message to a non-pollable station in state 2, when the transition *DNPollable* is enabled, thus moving to state q_3. In order for the access point to leave state 3 and return to 2, the station must leave its state returning to 1, which contradicts our assumption that the station is in state 2 while the access point is in state 2.

Using the same argument as in the case of the pollable station, the non-pollable station cannot be in states 3, 4 or 0 while the access point is in state 2.

From the above exhaustive two cases, assuming that there exists at least one station whose state is not 1 and the access point state is 2 leads to a contradiction, which proofs the lemma. ∎

Lemma 2 *If the access point is in state* 2*, then the medium must be idle.*

Proof The proof can be seen directly by noting that all the actions of the transitions that lead to state 2 clears the medium. ∎

Lemma 3 *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 access point and the remaining* $NSTA$ *entries of the tuple represent the state of the stations.*

Proof In the initial state, only the *Tx_SBeacon* transition in the access point is enabled. This makes the access point moves to state 1 setting the medium *Beacon* bit to 1. This enables the *Rx_SBeacon* transition in other stations and they can proceed to state 1 where all the transitions are not enabled. The access point cannot proceed from state 1 until all stations have set their entry in the *AllHeard* array. In this case, the access point moves to state 2 and the global state becomes $(1, 1, ..., 1, 2)$. ∎

Lemma 4 *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 access point, 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.*

Proof From *Lemma 1*, all stations must be in state 1, i.e. all x_i's and y_i's are equal to 1. Moreover, the medium must be idle from *Lemma 2*. This means that the change of the global state can only occur by the access point proceeding to a different state (all other stations require the medium to be not idle in order to leave state 1). For the access point, we have the following cases:

1. IFS_T= 0 $\wedge$ Remain_T=0: In this case, only the *Tx_End* transition is enabled and the access point moves to state 5 leading to a change of the global state and enabling the *Rx_End* transition in other stations. The access point cannot proceed until all stations have heard the *End* message and set their *AllHeard* entry to true. In this case, the *Proceed* transition in the access point is enabled whose action enables the *Wait* transition in other stations. After that, we have a DCF period and after it finishes, we have a *Proceed/Wait* state transition which leads the system to the initial state. From *Lemma 3*, the system will return to the state
$$(x_0, x_1, ..., x_{NPoll-1}, y_0, y_1, ..., y_{NSTA-NPoll-1}, 2).$$
2. IFS_T= 0 $\wedge$ (Remain_T- ONE_POLL_T) $>$ 0 $\wedge$ Beacon_T $>$ 0 $\wedge$ DataNPoll= -1: In this case, only the *Pollable* transition is enabled and the access point moves to state 4 leading to a change of the global state and enabling only one pollable station (depending on the *DA* field of the medium) and enabling one of the transitions (depending on the medium value). The station then sends a reply to the access point after *SIFS* period depending on its state and returning to state 1. This enables the access point to return to state 2 leading to the global state $(x_0, x_1, ..., x_{NPoll-1}, y_0, y_1, ..., y_{NSTA-NPoll-1}, 2)$.
3. IFS_T= 0 $\wedge$ Beacon_T $>$ 0 $\wedge$ (Remain_T- ONE_POLL_T) $>$ 0 $\wedge$ DataNPoll $\neq$ -1: In this case, only the *NDPollable* transition is enabled and the access point moves to state 3 leading to a change of the global state and enabling only one non-pollable station (depending on the *DA* field of the medium) and enabling the *Rx_Data* transition. The station then sends an acknowledgment to the access point after *SIFS* period returning to state 1. This enables the access point to return to state 2 leading to the global state $(x_0, x_1, ..., x_{NPoll-1}, y_0, y_1, ..., y_{NSTA-NPoll-1}, 2)$.
4. Beacon_T= 0 $\wedge$ Medium= 0 $\wedge$ IFS_T = 0: In this case, only the *Tx_MBeacon* transition is enabled and the access point moves to state 7 leading to a change of the global state and setting the *Beacon* bit of the medium to true. This enables the *Rx_MBeacon* transition of all the stations. Each station sets its *AllHeard* entry to true and returns to 1. This enables the access point to return to state 2 leading to the global state $(x_0, x_1, ..., x_{NPoll-1}, y_0, y_1, ..., y_{NSTA-NPoll-1}, 2)$.

From the above exhaustive cases, the lemma is proved. ∎

Theorem 1 (Safety) *The PCF protocol as specified is free from deadlock states.*

Proof From *Lemma* 3, Starting from the initial state, the system moves to the global state $(1, 1, ..., 1, 2)$. From *Lemma 4*, the system will always return to the same state, i.e. there is always progress, and hence no deadlock states. ∎

Corollary 1 *The PCF protocol as specified is free from nonexecutable 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.

4.2 Liveness properties

Theorem 2 (Liveness) *Each pollable station will be polled in a finite number of superframes.*

Proof At each PCF period, the access point must poll at least one pollable station (as specified in the standard). This means that the worst case waiting time for a station happens when the access point polls only one station per PCF period. Since the access point uses round robin scheduling, the maximum number of superframes before a station is polled is *NPoll*. ▮

The above 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 5 and the DCF period length may be unbounded due to collisions. In a more real IEEE 802.11 network, the use of backoff timers and different inter-frame spacing priorities lessens these effects.

4.3 Discussion

The fact that the PCF specification is free from deadlocks stems from the synchronized nature of the operation in the contention free period. The operations in the contention free period works at alternating cycles from the access point sending a poll to a station and waiting for a reply, which is guaranteed by the standard and the assumption of freedom from transmission errors. Even if the assumption of freedom from the transmission errors was relaxed, the standard indicates that the access point must wait for a reply to come within a *PIFS* period. If a reply does not come within this period, the access point moves to the next station in the polling list. So progress is still guaranteed. This can be modeled by a timeout timer in the access point. We leave that to future work.

The liveness property is guaranteed because of the round robin scheduling mechanism at the access point and the minimum length of the contention free period. However, as we indicated above, the length of both the PCF and DCF periods are variable, as shown in Figure 5 and the DCF period length may be unbounded due to collisions. In more real IEEE 802.11 networks, the use of backoff timers and different inter-frame spacing priorities lessens these effects.

5 Conclusions and Future Work

In this paper, we introduced a formal model for the IEEE 802.11 Point Coordination Function protocol using a Systems of Communicating Machines specification. We proved that the PCF protocol as specified is free from deadlocks and unexecutable transitions. We also showed that the protocol has liveness property in that each station in the polling list will be polled after a certain number of superframes. The model uses shared variables for communication between entities. The shared medium was modeled as a shared variable. The protocol modeled was the basic PCF protocol specified in the IEEE 802.11 standard with some abstraction.

The original IEEE standard specify the protocol using the SDL-92 modeling language. We believe that both the SDL model and the Systems of Communicating Machines model are important. The latter is important for simplicity of illustration of the protocol and for ease of proving the safety and liveness properties. The former is needed for the detail description of the protocol and the interaction between the physical and MAC layers. Moreover, since the 802.3 CSMA/CD MAC protocol have been modeled using the Systems of Communicating Machines model [7, 9], it is easier to specify the bridging function of the access point, that connects the wireless medium to the wired medium. The IEEE standard for the 802.11 protocol uses the SDL-92 language as specified above and for the 802.3 uses the programming language Pascal (with some extensions) [9]. By using the same model for both protocols, a translation need to be made only between protocols (which can be difficult enough on its own), and not also between protocol models.

A proof was given that the protocol is free from deadlocks. A full global or system state reachability analysis proved to be impractical due to the large number of states; instead, the proof was based on *invariant* techniques [9]. One technique is to state the desired property as an assertion, or invariant, and prove that the set of states satisfying the invariant is closed; that is, if the system is in a state satisfying the invariant, then it must remain in state which satisfies the invariant. In this paper, no invariant was explicitly stated; however, the proof we gave can be viewed as showing that the set of deadlock-free states is closed. We also showed that the protocol does not contain any nonexecutable transitions.

Another correctness property which of interest in protocol verification is *liveness*. This can be seen as a guarantee that something "good" will happen, such as the successful transfer of data from one machine to another. In the PCF protocol, we proved that each pollable station will be given a chance to transmit, i.e. it will be polled by the access point, after a fixed number of superframes. However, the length of both the PCF and DCF periods are variable, as shown in Figure 5 and the DCF period length may be unbounded due to collisions. In actual IEEE 802.11 network, the use of back-off timers and different inter frame spacing priorities lessens these effect.

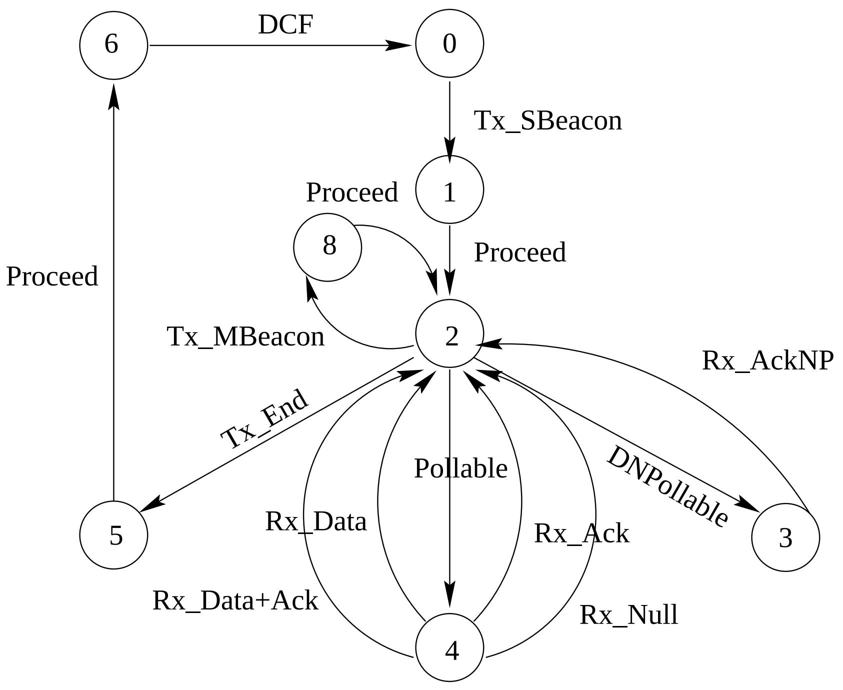

Figure 8: Specification for the access point (PC)

The specified PCF protocol is fair in terms that each pollable station will get a turn to transmit. This is true because the access point, which is the point coordinator, uses round robin scheduling.

For future work, other functions related to the PCF protocol, such as the association functions which are used to add a station to the polling list of an access point, can be modeled. Incorporating the DCF protocol into the model is a natural extension to the work. Relaxing the assumptions in the current model is another direction for future work. For example, the assumption that the environment is error free can be relaxed by allowing transmission error to occur. This can be modeled by another process that alter the medium to simulate transmission errors. Modeling an entire ESS and the interactions between different access points, and further analysis of the protocol are two other possibility for future work.

Table 1: Local, global, and timer variables specification for the access point

Variable name	**Range**	**Initial value**	**Local / Global**	**Purpose**
STAType	array [0..NSTA- 1] of {pollable, non-pollable}		L	StationType[i] stores the type of the i^{th} station
CurPSTA	0..NPoll- 1	0	L	current pollable station
PData	array [0..NPoll- 1] of integer	0	L	PData[i] is the number of messages buffered in the AP to the i^{th} pollable station
DataNPoll	-1.. NSTA- 1	-1	L	address for a non-pollable station for which the AP has data to send, -1 if none
AckNeeded	boolean	false	L	true if the AP received a data packet that needs acknowledgment
AllHeard	array [0.. NSTA- 1] of boolean	false	G	When all elements are true $\rightarrow$ all stations have heard a message
IFS_T	integer	PIFS	G	Controls the IFS (Shared with the timer)
Remain_T	integer	PCF_PERIOD	G	Controls the superframe duration (Shared with the timer)
Beacon_T	integer	BEACON_T	G	Controls the beacon interval (Shared with the timer)

Table 2: Local, global, and timer variables specification for stations

Variable name	**Range**	**Initial value**	**Local / Global**	**Purpose**
NAV	integer		L	Holds the remaining time of the current PCF period
HaveData	boolean		L	True if the station has data to send (set by an external entity)
Address	0.. NSTA- 1		L	Holds the destination address of the data to be transmitted (set by an external entity)
AllHeard	array [0.. NSTA- 1] of boolean	false	G	When all elements are true → all stations have heard a message
IFS_T	integer	PIFS	G	Controls the IFS (Shared with the timer)

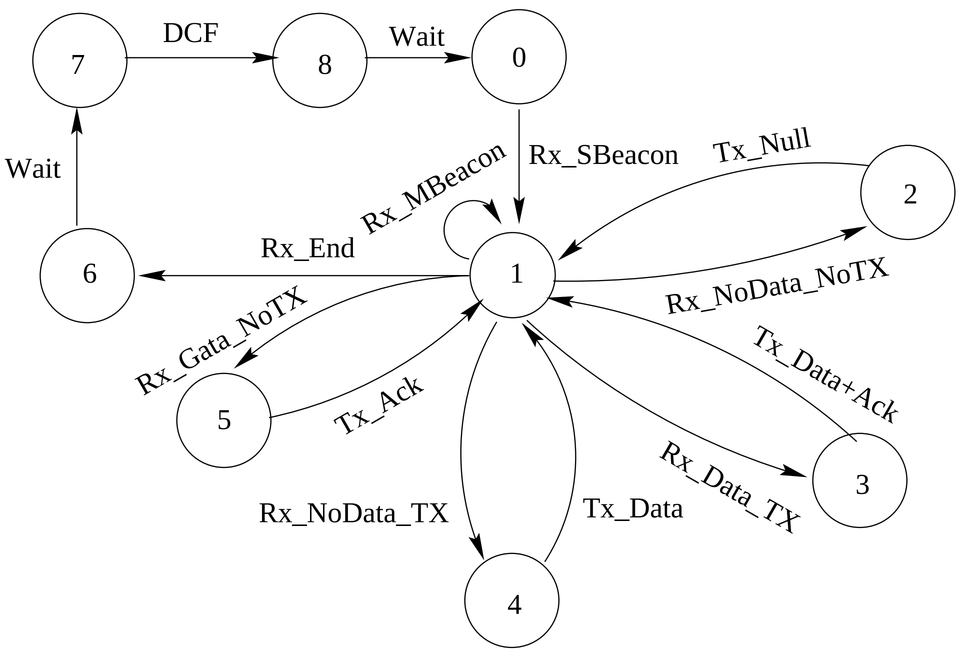

Figure 9: Specification for a Pollable Station i

Table 3: Predicate action table for the AP

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 DataNPoll= 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 DataNPoll= 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	$Beacon\_T = 0 \wedge IFS\_T = 0$	Beacon_T= BEACON_PERIOD $\wedge$ Remain_T $\neq$ 0; Medium.(Beacon, ToDS, End, Poll, NullData, Ack, Duration, DA) $\leftarrow$ (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;

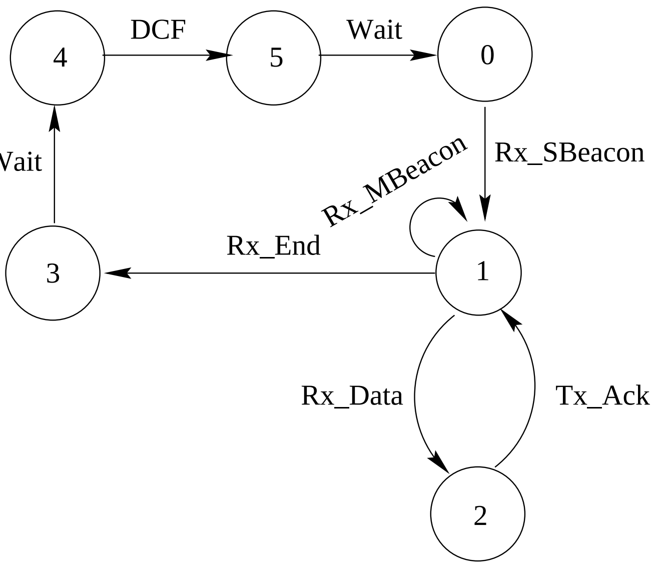

Figure 10: Specification for a non-Pollable Station i

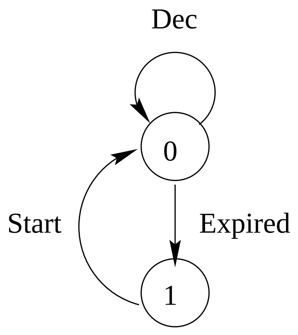

Figure 11: Specification for a Timer$i$

Table 4: 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 $\wedge$ Medium= 0	AllHeard[i]= true
Rx_Data_TX	Medium.(ToDS, Poll, NullData, DA) = (0, 1, 0, i) $\wedge$ 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) $\wedge$ 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) $\vee$ (Medium.(ToDS, Poll, NullData, DA) = (0, 1, 0, i) $\wedge$ 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	$Medium.(ToDS, Poll, NullData, DA) = (0, 1, 1, i) \wedge$ 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	$Medium.Beacon = 1 \wedge AllHeard[i] = false$	NAV= Medium.Duration; AllHeard[i]= true;
Wait	AllHeard[i]= false	

Table 5: Predicate action table for a non-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$\wedge$ 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	$Medium.Beacon = 1 \wedge AllHeard[i] = false$	NAV= Medium.Duration; AllHeard[i]= true
Wait	AllHeard[i]= false	

Table 6: Predicate action table for the IFS Timer

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

Table 7: Predicate action table for the Phase Period Timer

Transition	**Enabling predicate**	**Action**
Dec	$Remain\_T > 0$	Remain_T= Remain_T- 1
Expired	$Remain\_T = 0$	
Start	$Remain\_T > 0$	

Table 8: Predicate action table for the beacon interval timer

Transition	**Enabling predicate**	**Action**
Dec	$Beacon\_T > 0$	Beacon_T= Beacon_T- 1
Expired	$Beacon\_T = 0$	
Start	$Beacon\_T > 0$	

References

[1] http://www.digit-life.com/articles/wlan/.

[2] 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.

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

[4] William Stallings. *Wireless Communications and Networks.* Prentice Hall, first edition, 2002.

[5] Robert J. Orr and Gregory D. Abowd. A Performance Comparison of Point and Distributed Coordination Function of an IEEE 802.11 WLAN in the Presence of Real Time Requirements. In *7th International Workshop on Mobile Multimedia Communications*, Waseda, Tokio, Japan, October 2000.

[6] G. M. Lundy and Raymond E. Miller. *A Variable Window Protocol Specification and Analysis.* Elsevier Science Publishers, B.V. (North-Holland), 1988.

[7] 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.

[8] Harry Rudin. An Informal Overview of Formal Protocol Specifcation. *IEEE Communications Magazine*, 23(3):46–52, March 1985.

[9] G. M. Lundy and Raymond E. Miller. Analyzing a CSMA/CD Protocol through a Systems of Communicating Machines Specification. Technical Report TR-90-01, CESDIS TR, January 1990.