Further Definition of MAC/Ph Interface Primitives

October 10,1993

Submitted by
Larry Van Der Jagt
Knowledge Implementations, Inc.
32 Conklin Road
Warwick, NY 10990
Voice: 914-986-3492 FAX:914-986-6441 EMail:kiilvdj@attmail.com

Abstract

The MAC/Ph interface primitives that were discussed at the Atlanta meeting are elaborated upon. This elaboration takes the form of a brief tutorial representing the author's understanding of the language of ISO 7498 as it applies to our situation (comments regarding the accuracy of this understanding are welcome) as well as the application of this language to the coordination by the MAC-Entity of the functions of the Ph-Entity to provide the functional layering described in document 92/125 by this author and 93/140 by Diepstraten, Ennis and Belanger. One of the intentions of this paper is to examine various architectural options in the light of the actual language that might be used in our standard to describe the architectural elements and the interactions between those elements.

Note on Exposed Interfaces: At this point in time it is the opinion of this author that further discussion of the location and implementation detail of any exposed interface is not on the critical path towards achieving a standard and as such we should defer discussion of exposed interfaces until the work of standardization is further along. When we do get back to discussing exposed interfaces it should be in the context of what the use of the exposed interface might be. Once it is clear what it is for, then it will probably be clear how it should be implemented. I know of no instance in IEEE 802 where a desire to have a port for conformance testing has been used as a justification for exposing an interface (as always please, correct me if I am wrong).

Tutorial

In the terminology of ISO 7498 an (N)-Entity communicates to an (N-1)-Entity through an (N-1)-Service-Access-Point {(N-1)-SAP}. The (N-1)-Entity provides the (N)-Entity with (N-1)-Services through the use of this (N-1)-SAP. Communications between (N)-Entities within a network is achieved through the exchange of (N)-Protocol-Data-Units {(N)-PDUs} that conform to an (N)-Protocol.

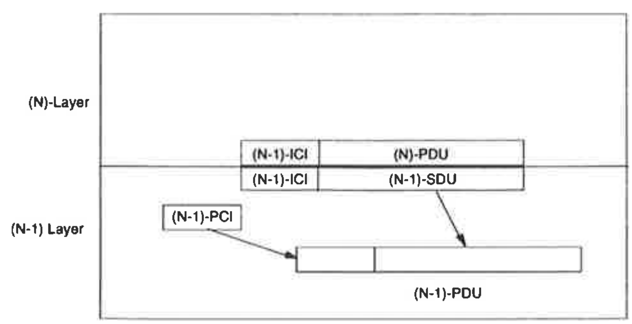
(N)-Entities can communicate (N)-Protocol-Control-Information to other (N)-Entities as part of an (N)-PDU. (N)-Entities can communicate data provided by (N+1)-Entities to other (N)-Entities using (N)-PDUs and this data is called (N)-User Data.

(N)-Interface Control Information is information exchanged between an (N+1)-Entity and an (N)-Entity to coordinate their operation. An (N)-Interface-Data-Unit is the combination of (N)-Interface-Data and (N)-Interface-Control-Information provided by an (N+1)-Entity to an (N)-Entity.

An (N)-Service is a capability of an (N)-Entity that is provided to an (N+1)-Entity. Not everything that a (N)-Entity does {(N)-functions} are (N)-Services, only those things that are accessable to the (N+1)-Entity are considered (N)-Services.

The diagram that appears below which has been copied and somewhat modified from Proccedings of IEEE, Vol. 71, No. 12 December 1983, "The OSI Reference Model", John D. Day and Hubert Zimmermann, is intended to illustrate the concepts. Please note that although descriptions of (N)-Interface-Control-Information appears in the above detailed reference, the presence of a box to generate this information within the (N) and (N-1) layer is an interpretation by this author and does not appear in the referenced text.

	Control	Data	Combined
(N)-(N) Peer Entities	(N)-Protocol-Control-Information	(N)-User-Data	(N)-Protocol-Data-Unit
(N)-(N-1) Adjacent Layers	(N-1)-Interface-Control-Information	(N-1)-Interface-Data	(N-1)-Interface-Data-Unit



PDU-Protocol Data Unit
SDU-Service Data Unit
PCI-Protocol Control Information
ICI-Interface Control Information

How Can This be Applied to Us - Using One SAP

Within the architecture described above information listed in Document 93/140 as information to be exchanged between MAC and PHY such as bit rate, time in current hop, chipping sequence, transmit power level and frequency identifiers can be treated as Ph-Interface-Control-Information and included as part of the PDU transfered to the Ph-Entity by the MAC-Entity. The box shown above within the (N-1) Layer and labeled (N-1)-ICI is reminiscent of the "MAC Management" box that appears within our existing architectural model, although this box might more appropriately be called Ph-Interface-Control-Managment.

If for a moment we conceive of this occuring through a single SAP rather than through 2 SAPs this could be modeled through the use of the primitives discussed in paper 93/162 as a new class parameter for the Ph_DATA_Request primitive that might be called INTERFACE-CONTROL-INFORMATION and to have for that class a data parameter defined to be the Transmit Parameter Information Vector {TPIV} (as described in 92/125) for this particular transmission. This TPIV could contain all of the items listed as to be exchanged between MAC and PHY within Document 93/140.

The interpretation of the data in the TPIV would be different for different Ph-Entities, but the fact a a TPIV is passed would be generic to all Ph-Entities. A certain portion of this TPIV information would actually be MAC Protocol Data within the context of the scheme presented in 93/125 (power setting of the transmitter for instance), and this information would need to be transferred along with the rest of the MPDU by the Ph-Service. The rest of the Ph-Interface-Control-Information would not need to go onto the airwaves.

The sequence of primitives in this case would involve the Ph-Entity issuing a Ph_DATA.confirm immediately upon receipt of the Ph_DATA.request with class START-OF-ACTIVITY with the next Ph_DATA.request from the MAC being one having the class INTERFACE-CONTROL-INFORMATION. This request would then be confirmed by the Ph-Entity when it was appropriately set up and ready to accept data.

Alternatively, the TPIV could be passed as the data parameter of the Ph_DATA.request with class START-OF-ACTIVITY.

On the receive side, a similiar set of constructions could be made. In this case a new class could be introduced for the Ph_DATA.indication that would again be INTERFACE-CONTROL-INFORMATION and which would have Ph-Entity specific interpretations for the data parameter associated with primitives having this class. This technique could be used to transfer the Received Parameter Information Vector, let's call it RPIV which would include information like bitrate,signal level,SNR etc. refered to in Document 93/140.

Again this could also be modeled as the definition of the values associated with the data parameter when the class is START-OF-ACTIVITY.

By using a conceptual model of the communications between MAC-Entities and Ph-Entities as detailed above the utility of a Ph-Independent SubLayer within the Ph-Entity becomes apparent. This is the sublayer that handles interactions with the MAC-Entity that are the same regardless of Ph-Entity type. Namely these interactions involve accepting MPDUs from the MAC-Entity, extracting the TPIV from those MPDUs and transmitting those TPIV to the Convergence Sublayer for this specific type of Ph-Entity to act upon. On the receive side the Ph-Independent Sublayer of the Ph-Entity accepts RPIV information from the Convergence Layer and combines it with other information to be provided to the MAC-Entity as part of the Ph-Service-Data-Unit.

If the above detailed approach to MAC-Ph interface were adopted the changes that would seem to make sense to the architectural model currently defined in document 93/20a2 would be the renaming of the MAC management block to Ph-Interface-Control-Information Management and the combination of the two SAPs detailed at the MAC-Ph interface into one SAP that straddles the line between these two portions of the MAC.

Alternatively, should the MAC architects wish to implement a multiplexing sublayer at the bottom of the MAC that multiplexes the Ph-Interface-Control-Information with the Ph-User-Data, this would also appear to be a reasonable model.

Within the context of document 92/125, the Ph-Specific mapping functions that the MAC would use within the Ph-Interface-Control-Information Management function to determine what the TPIV should be for a particular MPDU might best be conveyed through the normal layer management functions and as such the details of how this interchange takes place (although not the information that could be interchanged) would be (in the author's opinion) an implementation issue beyond the scope of this standard.

<u>How Can This be Applied to Us - Using Two SAPs (or a managed object boundary and a SAP)</u>

The type of interactions described above could also be used if we maintain the model as it is currently formulated and use two SAPs, one for the transmission of MPDUs and one for the transmission of Ph-Interface-Data-Units. If this type of a model is to be used an new set of primitives would be defined for use on the SAP providing interface control services. (Alternatively, this might be architected to be a situation where a set of managed objects are defined for the Ph-Entity and access to these managed objects could be provided through a managed object boundary at the MAC/Ph interface. This would seem logical if the term "MAC-Management" or more likely "Ph-Management" continues to be used in our model. If this approach were taken I think the appropriate changes to the primitives described below would be the replacement of the words request with invoke, confirm with reply and indication with notify.) One possible way to approach this set of service primitives would be through the use of a set of abstract service primitives such as the following:

Ph_SET-VALUE.request

Ph_SET-VALUE.confirm

Ph_GET-VALUE.request

Ph_GET-VALUE.confirm

Ph_ACTION.request

Ph_ACTION.confirm

Ph_EVENT.indication

These primitves are simply named here with the details of any discussion of their possible use defered until a future submission and until after the objects on which they are expected to operate are defined.

Overview of Possible Managed Objects

In order to start the conversation, this author proposes that Managed Objects (MOs) be defined for the Ph-Entity that are organized in groups. These groups would be:

1)**ResourceTypeID** - These objects would indentify the type of Ph-Entity that is present and what revision of the standard it conforms to.

2)**Capabilities Group** - These objects would specify what set of capabilities this particular instance of a Ph-Entity can provide. Proposed objects within this group are:

a) dataRates - indicates data rates at which the Ph-Entity can operate

b) transmitPowerLevel - indicates the transmit power levels the the Ph-Entity is capable of producing

c) transmitDiversityOption - indicates the number and type of diversity options available for transmission. In this context the channel on which a message is sent is considered a diversity option.

d) receiveDiversityOption - indicates the number and type of diversity options available for receiving transmissions.

e) measurableReceiveLevel - indicates the capabilities of the Ph-Entity with respect to reporting receive signal levels

3) **Operational State Group** - These objects would describe the current state of operation of the Ph-Entity. The objects in this group would correspond to the objects in the capbilites group, but would be intended to indicate not what the Ph-Entity could do, but rather what it is doing.

4) **Initialization State Group** - These objects would be used to contain the details of what the values of the Operational State Group objects should be a intialization time

5) **Counter Group** - These objects would count various events of interest for the purposes of providing operational statistics and event notifications.

Having defined these groups of objects it seems reasonable that we define a set of Actions that can operate on these objects. Some of the possible Actions are described below.

a) adjustTransmitPower - this would permit either the setting of the transmit power should the standard provide for the MAC-Entity physically doing this as part of its operation (a concept supported by this author) or of the Ph-Entity executing some algorithm to determine the transmit power at the request of the MAC-Entity.

b) adjustEventthresholds - this would permit either the setting of event thresholds (such as what level should be considered silence in an energy detect situation) by the MAC or the instruction of the MAC-Entity to the Ph-Entity to perform an algorithm to determine and set these thresholds.

c) selfTest - this would allow the MAC-Entity to cause the Ph-Entity to excercise various self tests including various loopbacks.

Finally, the Events that will be reported to the MAC-Entity by the Ph-Entity should be defined. On obvious event that is required by a number of MAC proposals is receiveLevelThresholdExceeded. Others might be jabberDetected, etc.

Preliminary Definition of Managed Objects

Having briefly outlined the scope of the definitions that might be required an introductory attempt at the definition of a few objects will be made. For the purposes of this document it will be assumed that all of the objects for the Ph-Entity will be subclasses of the class Ph-Entity that for now is not defined. At the time this is being written the author does not have a working knowledge of the content of IEEE 802.1.F and this may have some impact on whether this format should or should not be used by IEEE 802.11 in its standard. In any event the work of formal definition of objects will not be wasted as translation to whatever symantic framework is required should be relatively straight forward. Please forgive possible shortcomings in this attempt to provide formal descriptions, it is intended as a starting point.

```
TransmitEntity MANAGED OBJECT CLASS
   DERIVED FROM              PH-Entity;
   CHARACTERIZED BY          transmitPackage;
   CONDITIONAL PACKAGES
        adjustPower
         PRESENT IF          !Multiple Transmit Power Level Support
                             is implemented in this instance!;
        adjustDataRate
         PRESENT IF          !Multiple Data Rate Support is
                             implemented in this instance!;
        adjustDiversity
         PRESENT IF          !Diversity Support is implemented in
                             this instance!;
REGISTERED AS {TBD};

transmitPackage PACKAGE
   BEHAVIOUR            adjustableParameters;
   ATTRIBUTES           transmitPower GET, transmitDataRate GET
                        transmitDiversityOption GET;
   ATTRIBUTES  GROUPS capabilitiesGroup,operationalStateGroup,
                        initializationGroup;
   NOTIFICATIONS        None;

adjustPower PACKAGE
   BEHAVIOUR            adjustableParameters;
   ATTRIBUTES           transmitPower REPLACE;
   ATTRIBUTES  GROUPS capabilitiesGroup,operationalStateGroup,
                        initializationGroup;
   NOTIFICATIONS        None;

adjustDatRate PACKAGE
   BEHAVIOUR            adjustableParameters;
   ATTRIBUTES           transmitDataRate REPLACE;
   ATTRIBUTES  GROUPS capabilitiesGroup,operationalStateGroup,
                        initializationGroup;
   NOTIFICATIONS        None;

adjustDiversity PACKAGE
   BEHAVIOUR            adjustableParameters;
   ATTRIBUTES           transmitDiversityOption REPLACE;
   ATTRIBUTES  GROUPS capabilitiesGroup,operationalStateGroup,
                        initializationGroup;
   NOTIFICATIONS        None;
```

```
ReceiveEntity MANAGED OBJECT CLASS
   DERIVED FROM               Ph-Entity;
   CHARACTERIZED BY           receivePackage;
   CONDITIONAL PACKAGES
       adjustThreshold
        PRESENT IF            !Multiple Receive Level Threshold
                              Support is implemented in this
                              instance!;
       adjustDiversity
        PRESENT IF            !Diversity Support is implemented in
                              this instance!;
REGISTERED AS {TBD};

receivePackage PACKAGE
   BEHAVIOUR               adjustableParameters;
   ATTRIBUTES              receiveThreshold GET, receiveDataRate GET,
                           receiveLevel GET, receiveDiversityOption
                           GET;
   ATTRIBUTES GROUPS       capabilitiesGroup,operationalStateGroup,
                           initializationGroup;
   NOTIFICATIONS           receiveThresholdCrossed;

adjustThreshold PACKAGE
   BEHAVIOUR               adjustableParameters
   ATTRIBUTES              receiveThreshold REPLACE;
   ATTRIBUTES GROUPS       capabilitiesGroup,operationalStateGroup,
                           initializationGroup;
   NOTIFICATIONS           None

adjustDiversity PACKAGE
   BEHAVIOUR               adjustableParameters
   ATTRIBUTES              receiveDiversity REPLACE;
   ATTRIBUTES GROUPS       capabilitiesGroup,operationalStateGroup,
                           initializationGroup;
   NOTIFICATIONS           None

capabilitiesGroup ATTRIBUTE GROUP
  GROUP ELEMENTS           transmitPower, transmitDataRate,
                           transmitDiversityOption, receiveLevel,
                           receiveThreshold, receiveDiversityOption,
                           receiveDataRate;
  DESCRIPTION              !Attribute Group that includes all
                           capabitity options associated with Ph-
                           Entity Class;
REGISTERED AS {TBD};
```

```
operationalStateGroup ATTRIBUTE GROUP
  GROUP ELEMENTS      transmitPower, transmitDataRate,
                      transmitDiversityOption, receiveLevel,
                      receiveThreshold, receiveDiversityOption,
                      receiveDataRate;
  DESCRIPTION         !Attribute Group that includes operational
                      state options associated with Ph-Entity Class;
REGISTERED AS {TBD};

initializationGroup ATTRIBUTE GROUP
  GROUP ELEMENTS      transmitPower, transmitDataRate,
                      transmitDiversityOption, receiveLevel,
                      receiveThreshold, receiveDiversityOption,
                      receiveDataRate;
  DESCRIPTION         !Attribute Group that includes initialization
                      state options associated with Ph-Entity Class;
REGISTERED AS {TBD};

adjustableParameters BEHAVIOUR
  DEFINED AS !Elements of managed classes exhibiting this behaviour
             obtain their initial values from the value of the
             appropriate initializationGroup element with the object
             name, report their possible values with the
             capabilities Group Element of the object name and
             report their current value with the operationStateGroup
             element of the object name!;

receiveThresholdCrossed NOTIFICATION
  BEHAVIOUR
     !Generated when receiveLevel traverses the receiveThreshold
     Level!;
  WITH INFORMATION SYNTAX   NotificationModule.ReceiveRange;
```

ASN.1 Modules

```
NotificationModule  {TBD}
DEFINITIONS  ::=BEGIN
ReceiveRange  ::=  SET{[0]BelowThreshold,[1]AboveThreshold}
END
```

Conclusion

The intent of this paper has been to further define the primitives associated with the MAC/Ph interface and to begin to formalize the way in which the architectural model relates to the process of standard writing. In conjunction with this effort the author has attempted to detail how the functional partitioning requested in document 93/140 might be implemented within the context of the existing architectural model and to examine likely modifications that could improve the utility of that model.