Mathematica Based Integrated MAC/PHY Performance Simulation Framework Including Capture Effect

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

Main Issues Addressed:
29.1 How Does IEEE 802.11 Do Simulation
9.1 MAC throughput & throughput probability
9.4 How will Standard address attenuation

■ Overview

A detailed computational framework is established for the execution of performance simulations of Media Access Control State machines operated over Physical Layer Entities. This framework allows the experimenter to locate stations geographically and to assign attributes to those stations. Some of the attributes that can be defined are the station **Location**, **TxPower**, **State** and **MessageProbability**. A random sample from a statistical model is used to simulate the path loss between stations. The path loss matrix that results from this sampling is used to determine the signal level at each station resulting from transmissions from other stations. If the signal to interference ratio at the receiving station is sufficient (as determined by whether or not it exceeds that value of the **CaptureMargin** parameter) the transmission is considered successful. Each station maintains a **StationTxQueue** and a **StationRxQueue** that tracks messages that it desires to send and that it receives. The filling and emptying of these queues is performed by traffic generating functions and by a state machine model of MAC operation. In this document that state machine model is a very simple ALOHA type model and the transmission environment is considered static once the orignal sampling of the fading distributions are taken. The framework is flexible enough, however, to be extended to other more complex MAC state machines and dynamically varying channel models. There is also a provision in the framework to enter station specific interference levels from external sources not related to the transmission of other simulated stations.

In presenting this framework it is our intention to provide a common methodology for analyzing MAC/PHY performance that can be executed on a variety of platforms and be within the expense budget of all IEEE 802.11 participants. We feel that the use of *Mathematica* as a software tool accomplishes this goal. The code can be developed by experimenters with PC, Macintoshes and workstations alike. When PC or Macintosh based execution of scenarios runs out of gas, the members that are fortunate enough to have workstations might assist those less fortunate in executing the simulations. *Mathematica* is flexible enough to provide accurate modeling of real world conditions. The only potential con is that people would have to learn *Mathematica* but this will be a con for any simulation platform.

As has been the case with previous submissions we have made this submission executable, when the corresponding package of support functions is available. Unlike previous submissions, however, this support function package is not included with this submittal. We would be happy to discuss terms for making these support functions available to interested parties.

▩ Initialization

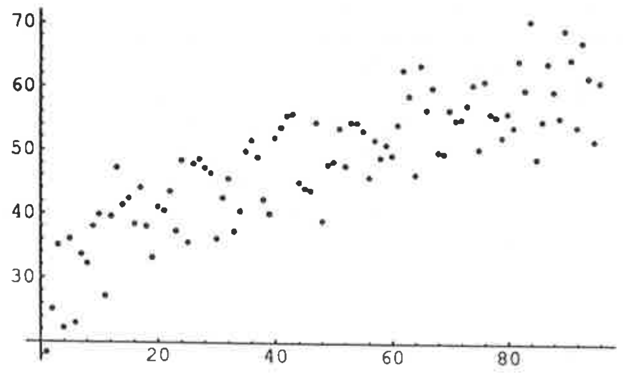
The process of executing a simulation begins with the loading of the support functions that are contained in the package **RadioLAN`RadioSupport**. Each of these functions will be described in the sections of the notebook that follow. The next cell loads the support package.

```
<<RadioLAN`RadioSupport`
```

In order to utilize this notebook it is necessary to initialize the fading statistics. This is accomplished using the function **SetFadingParameters**. This function takes three parameters that are used in calculating the path loss as a function of distance. The large scale fading is determined by using the first parameter and the second parameter as the mean and standard deviation of a normal distribution function and taking a random sample of this distribution to be the **n** used to calculate the large scale attenuation 1/d^n where d is the distance at which the attenuation is desired. The value of 1/d^**n** (expressed in dB) found as the large scale fading value is used as the mean of a second normal distribution with a standard deviation equal to the third parameter. This second distribution models the local fading characteristics. A random sample from this distribution is the value returned for the attenuation at a specific distance.

The cell that follows sets the fading parameters for this specific instance of the simulation. This cell must be executed prior to the start of the simulation. An example of the output of the fading model is provided in the following few cells. The function **GetAttenuation[d]** is the actual function that is used when a sample of the statistical model established with **SetFadingParameters** is required for a given distance.

```
SetFadingParameters[3,.1,5];

Table[-1 GetAttenuation[d],{d,5,100}]

{18.3963, 25.151, 35.0406, 22.2475, 36.0195, 22.9756, 33.5822, 32.1841, 37.9893, 39.8256, 27.2057, 39.5391, 47.2109, 41.358,
  42.4815, 38.3741, 44.1254, 38.0362, 33.2227, 41.0843, 40.5207, 43.5098, 37.3255, 48.3905, 35.5756, 47.9233, 48.612,
  47.2344, 46.3836, 36.1678, 42.5582, 45.5401, 37.2992, 40.5007, 49.9283, 51.631, 48.9692, 42.3184, 40.0866, 51.9977,
  53.6759, 55.5021, 55.804, 45.015, 44.0488, 43.7306, 54.4328, 38.9679, 47.8289, 48.2987, 53.6021, 47.6136, 54.4817, 54.4211,
  53.175, 45.86, 51.7246, 48.9437, 50.9567, 49.3227, 54.1231, 62.7712, 58.7104, 46.3249, 63.56, 56.524, 60., 49.9135,
  49.6546, 56.5146, 54.9053, 55.0889, 57.3055, 60.5633, 50.3674, 61.0733, 56.0175, 55.4668, 52.2562, 56.0537, 53.8633,
  64.3407, 59.7712, 70.6298, 48.8805, 54.8483, 63.9587, 59.5789, 55.4111, 69.197, 64.6123, 54.008, 67.3151, 61.7355, 51.7799,
  61.0449}

ListPlot[%]
```



```
-Graphics-
```

This simulation allows the experimenter the flexibility of setting up the station population that will be involved in the simulation. Each station must have **Location** (in a three dimensional space), **TxPower** (in dBmw), **MessageProbability** (the mean of a Poission Process) and **State** (initial state of the station of the MAC state machine) specified. The cells that follow specify these parameters for five stations. The names of these stations are arbitrary, however, a list of those must be placed in a list assigned to the symbol **Stations** in order for the simulation to operate. Also, a station should not be named **empty** since this is the indicator of an empty transmit queue. Further discussion of the transmit queue appears later in this paper. Other station attributes that are initalized by the function **initialize** are the starting state of the **StationTxQueue** and **StationRxQueue**, as well as the **Attempts**,**Success**,**Failure**, and **RetryCount** attributes of each station.

```
Initialize:=(
Location[Station1]^={1,1,1};
TxPower[Station1]^=10;
MessageProbability[Station1]^=.2;
State[Station1]^=Idle;
StationRxQueue[Station1]^={empty};
StationTxQueue[Station1]^={empty};
Attempts[Station1]^=0;
Success[Station1]^=0;
Failure[Station1]^=0;
RetryCount[Station1]^=0;

Location[Station2]^={5,5,5};
TxPower[Station2]^=10;
MessageProbability[Station2]^=.2;
State[Station2]^=Idle;
StationRxQueue[Station2]^={empty};
StationTxQueue[Station2]^={empty};
Attempts[Station2]^=0;
Success[Station2]^=0;
Failure[Station2]^=0;
RetryCount[Station2]^=0;

Location[Station3]^={10,10,10};
TxPower[Station3]^=10;
MessageProbability[Station3]^=.2;
State[Station3]^=Idle;
StationRxQueue[Station3]^={empty};
StationTxQueue[Station3]^={empty};
Attempts[Station3]^=0;
Success[Station3]^=0;
Failure[Station3]^=0;
RetryCount[Station3]^=0;

Location[Station4]^={15,15,15};
TxPower[Station4]^=10;
MessageProbability[Station4]^=.2;
State[Station4]^=Idle;
StationRxQueue[Station4]^={empty};
StationTxQueue[Station4]^={empty};
Attempts[Station4]^=0;
Success[Station4]^=0;
Failure[Station4]^=0;
RetryCount[Station4]^=0;

Location[Station5]^={20,20,20};
TxPower[Station5]^=10;
MessageProbability[Station5]^=.2;
State[Station5]^=Idle;
StationRxQueue[Station5]^={empty};
StationTxQueue[Station5]^={empty};
Attempts[Station5]^=0;
Success[Station5]^=0;
Failure[Station5]^=0;
RetryCount[Station5]^=0;)
```

The following cell executes the initialization function defined above and also builds the list of stations that is required by other processing steps.

```
Initialize
Stations={Station1,Station2,Station3,Station4,Station5};
```

Once the station **Location** parameters and the fading parameters are set the function **BuildAttenuationTable** can be called with the station list as a parameter and a table of attenuations between stations will be calculated. In this calculation, an assumption is made that reciprocity applies with respect to attenuation of the channel. That is, the path loss between two stations is the same regardless of which station is transmitting and which is receiving. This assumption leads to a symmetric matrix representation for the attenuation between stations. The row and column numbers represent which stations the attenuation value applies to. For instance, the attenuation between the **Station1** and **Station2** in this example appears in both (row one,column two) and (row two, column one). The cell that follows illustrates the process of generating an attenuation matrix. The symbol **Attenuation** is assigned to the calculated Attenuation Table for use by other routines, specifically, by **GenerateReceiveLevels**, the routine that calculates the receive level at each station for transmissions originating at each other station and the overall level of receive power at a given station.

```
BuildAttenuationTable[Stations]
Attenuation=%;

0          -30.9655   -28.781    -35.0935   -41.0809
-30.9655   0          -21.3285   -42.8213   -43.4019
-28.781    -21.3285   0          -22.7665   -43.2612
-35.0935   -42.8213   -22.7665   0          -25.2425
-41.0809   -43.4019   -43.2612   -25.2425   0
```

Having established the parameters associated with the station configuration the next task undertaken is to begin the process of simulating a transmission environment using these stations. The function **GenerateTraffic** uses the **MessageProbability** associated with each station to determine how many messages a particular station will generate during this time interval. A station name is selected using a uniform discrete distribution from the possible destinations stations that a particular station has available to it (Note: all stations other than itself in this model) for each message that is generated. These station's names are placed in a list and are appended to the end of the list currently in existance as the **StationTxQueue** attribute of each station. This queue consists of a list of destination stations for which these messages are intended. The cells that follow illustrate the generation of the initial set of traffic. Each successive call will append traffic to the end of the **StationTxQueue** for each station. When operating in conjunction with state machines that take messages from the beginning of **StationTxQueue**, the function **GenerateTraffic** implements a FIFO style queue for the desired transmissions.

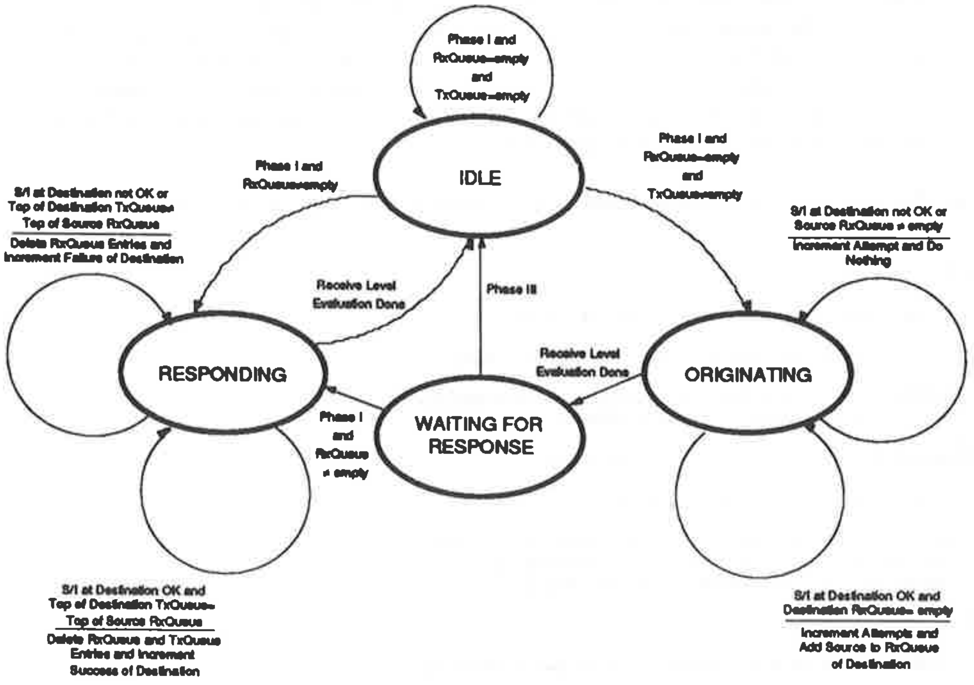
```
GenerateTraffic[Stations];
StationTxQueue[#]&[Stations]

{{empty}, {empty}, {Station2, empty}, {Station1, empty}, {empty}}
```

The function that evaluates the instantaneous receive level at a given station is **GenerateReceiveLevels**. This function monitors the state of all stations each time it is executed and calculates the receive power of all stations that are in the **Originating** and **Responding** state at all other stations. The level of receive power at a particular station resulting from the transmission of another station are stored in the matrix **ArrivingTxPowerTable** by **GenerateReceiveLevels**. In order to avoid negative infinity problems a minimum receive power level is established for all stations. This level is set by setting the symbol **MinPower**. Also in order to facilitate evaluation of how this set of stations will operate in the presence of external noise sources the table **ExternalNoiseTable** of external noise values at individual stations is required. This table is an array with one entry for the external noise level at each station (Note: noise in this array is expressed as power in Watts, not dBmw) . **GenerateReceiveLevels** also calculates the array **ArrivingPowerTable** that has one entry for each station representing the total power arriving at the station from all sources including external noise. The following cell sets the required parameters. The parameters **CaptureMargin** and **retrylimit** are used later in the notebook and are only initialized here for convenience. They are not used by **GenerateReceiveLevels**.

```
MinPower=-100;
ExternalNoiseTable=10^({-100,-100,-100,-100,-100}/10).001;
CaptureMargin=12;
retrylimit=3;
```

■ The MAC/PHY State Machines Used For This Simulation

Having established the support functions required, the functions that actually process the station state machines can now be presented. The state machine model that has been chosen for this simulation involves four states namely **Originating**, **WaitingForResponse**, **Responding** and **Idle**. The relationship between these states and the transition conditions required to move from one state to another are detailed in the following diagram.


Phase I and RxQueue=empty and TxQueue=empty
IDLE
Phase I and RxQueue≠empty
Phase I and RxQueue=empty and TxQueue≠empty
S/I at Destination not OK or Top of Destination TxQueue≠ Top of Source RxQueue
Delete RxQueue Entries and Increment Failure of Destination
S/I at Destination not OK or Source RxQueue ≠ empty
Increment Attempt and Do Nothing
Receive Level Evaluation Done
Phase III
Receive Level Evaluation Done
RESPONDING
ORIGINATING
WAITING FOR RESPONSE
Phase I and RxQueue ≠ empty
S/I at Destination OK and Top of Destination TxQueue= Top of Source RxQueue
Delete RxQueue and TxQueue Entries and Increment Success of Destination
S/I at Destination OK and Destination RxQueue= empty
Increment Attempts and Add Source to RxQueue of Destination

These state machines are provided as an example and are not being suggested as typical of the MAC state machines we may ultimately chose for the standard and can readily be modified to reflect actual MACs being considered. The execution of the state machines proceeds in three phases. During the first phase, stations that need to go into a transmitting state (**Originating** or **Responding**) make their transition to that state. During the second phase, receiver signal to interference levels are calculated to determine if the reception will be successful. For this example a full duplex PHY is assumed that can remove the impact of its own transmission from the receive signal. It is also assumed that the MAC can only deal with one item on the **StationRxQueue** at a time and that if there is already an item on this queue the transmission will fail regardless of the signal to interference ratio. It is during this second phase of operation that decisions are reached with respect to whether a particular transmission succeeds or fails. Finally, a third phase of operation results in stations transitioning back to **Idle** having evaluated the reception conditions. As currently configured every transmitted message is responded to by the destination station and failure of either the basic message or the response is considered a failure of the overall transmission, resulting in incrementing of the originating stations **Failures** attribute.

A number of support functions are used in the execution of the state machines. The first of these is **CheckForReceive**. This function accepts two parameters, source and destination. It begins by incrementing the **Attempts** attribute of the source station. Next it calls **GenerateReceiveLevels** to test to see if the transmission will be successful based on the signal to interference conditions. It also checks to make sure that the **StationRxQueue** of the receiving station is **empty**. If all conditions for successful transmission are satisfied the name of the source is placed on the destination's **StationRxQueue** by a call to the function **GenerateMessage**. If transmission conditions will result in the destination station not being able to successfully receive the transmitted message the address of the source is not placed on the destination **StationRxQueue** and subsequent processing will reveal a failure of the source to receive a response from the destination.

The function **CheckForResponse** is used by a station in the **Responding** state to determine if the response will be successfully received by the station being responded to. The processing in this function is very similar to that performed by **CheckForReceive**, however, if the transmission is found to be successful a single item is deleted from the appropriate Tx and Rx queues and the source station's **Success** attribute is incremented. If the transmission is not successful the Rx queue of the responder is cleared and the Tx queue of the destination station is left intact and the **Failures** attribute of the destination station is incremented.

The function defined in the following cell executes the state machines described in the previous paragraphs.

```
ExecuteStateMachine[stationlist_]:=

(*All Stations are evaluated to put them in transmitting states if required*)

(*First test to see if any stations are in idle needing to go to responding*)

(Do[If[(((testtable=Table[{State[#1][[i]],StationRxQueue[#1][[i,1]]}&[stationlist],
   {i,1,Length[stationlist]}])[[k,1]]===Idle)||(testtable[[k,1]]===WaitingForResponse))
   &&(testtable[[k,2]]=!=empty),
   State[stationlist[[k]]]^=Responding,{}],{k,1,Length[stationlist]}];

(*Next check to see if any station is in idle needing to go to originating*)

Do[If[((testtable=Table[{State[#1][[i]],StationTxQueue[#1][[i,1]]}&[stationlist],
   {i,1,Length[stationlist]}])[[k,1]]===Idle)&&(testtable[[k,2]]=!=empty),
   State[stationlist[[k]]]^=Originating,{}],{k,1,Length[stationlist]}];

Print[State[#]&[stationlist]];

(*With stations in transmitting states evaluate s/i conditions at destinations*)

Do[Switch[State[#1]&[stationlist][[i]],

 (*Now check to see if s/i conditions at destinations allow reception from orginators*)

 Originating,CheckForReceive[
    stationlist[[i]],StationTxQueue[#1]&[stationlist][[i,1]],stationlist],

 (*Now check to see if s/i conditions at destinations allow reception from responders*)

 Responding,CheckForResponse[
      stationlist[[i]],StationRxQueue[#1][[i,1]]&[stationlist],stationlist],

 (*Any Station not in transmitting state is dealt with later*)

 True,{}],{i,1,Length[stationlist]}];

(*Now that impact of all transmitters is defined move to next state*)

Do[Switch[State[#1]&[stationlist][[i]],

 (*If Orginating go to idle*)

 Originating,State[stationlist[[i]]]^=WaitingForResponse,
(*If Responding go to idle*)

Responding,State[stationlist[[i]]]^=Idle,

(*If Waiting go to idle*)

WaitingForResponse,State[stationlist[[i]]]^=Idle,

(*All Other Stations stay in as is*)

True,{}],{i,1,Length[stationlist]}]]
```

■ Simulation Execution and Results

Finally, the stage is set for performing an actual simulation. The only other function that is used that has not been discussed previously is the function **MonitorRetries**. This function accepts two parameters, one being the last state of all the **StationTxQueues**, and the other being the **stationlist** that is used to generate the current state of the **StationTxQueues**. This function monitors the transmit queues and discards messages that fail to be transmitted a **retrylimit** number of times. This keeps the simulation from giving pessimistic results when one station can not be reached by another station under any interference situation and prevents deadlocks associated with the only one entry on the Rx queue limitation that has been imposed on the simulated MAC. The following cell executes 10 cycles of the defined simulation and provides diagnostic information as it is executing as well as summary statistics upon completion.

Two sets of simulation output are presented here, one for a -20 dB capture margin such as you might expect to see in Spread Spectrum systems with 30 dB of processing gain and one for a 12 dB capture margin such as you might expect to see in a Non-Spread Spectrum system.

```
Initialize
lasttraffic=StationTxQueue[#1]&[Stations];
Do[(MonitorRetries[lasttraffic,Stations];
     GenerateTraffic[Stations];
     lasttraffic=StationTxQueue[#1]&[Stations];
          Print["Slot Number ",n];
          Print[StationTxQueue[#1]&[Stations]];
          Print["StationRxQueue     ",StationRxQueue[#]&[Stations]];
          Print["State          ",State[#]&[Stations]];
          Print["Attempts       ",Attempts[#]&[Stations]];
          Print["Success        ",Success[#]&[Stations]];
          Print["Failure        ",Failure[#]&[Stations]];
          Print["Retries        ",RetryCount[#]&[Stations]];
          totaliterations=n;
          ExecuteStateMachine[Stations]),{n,1,10}]

Print["Message Probabilities   ",MessageProbability[#]&[Stations]]
Print["Capture Margin      ",CaptureMargin]
OverallEfficiency=N[Apply[Plus,Success[#]&[Stations]]/(totaliterations-2)];
Print["Overall Efficency        ",%]
SuccessEfficency=Apply[Plus,Success[#]&[Stations]]/Apply[Plus,Attempts[#]&[Stations]]//N;
Print["Success Efficency        ",%]
FailureEfficency=Apply[Plus,Failure[#]&[Stations]]/Apply[Plus,Attempts[#]&[Stations]]//N;
Print["Failure Efficency        ",%]
```

```
Slot Number 1
{{empty}, {empty}, {Station2, empty}, {empty}, {Station3, empty}}
StationRxQueue    {{empty}, {empty}, {empty}, {empty}, {empty}}
State        {Idle, Idle, Idle, Idle, Idle}
Attempts     {0, 0, 0, 0, 0}
Success      {0, 0, 0, 0, 0}
Failure      {0, 0, 0, 0, 0}
Retries      {0, 0, 0, 0, 0}
{Idle, Idle, Originating, Idle, Originating}
Evaluating TX Conditions between Station3 and Station2
Evaluating TX Conditions between Station5 and Station3
Slot Number 2
{{empty}, {empty}, {Station2, empty}, {empty}, {Station3, empty}}
StationRxQueue     {{empty}, {Station3, empty}, {Station5, empty}, {empty},
{empty}}
State        {Idle, Idle, WaitingForResponse, Idle, WaitingForResponse}
Attempts     {0, 0, 1, 0, 1}
Success      {0, 0, 0, 0, 0}
```

```
Failure      {0, 0, 0, 0, 0}
Retries      {0, 0, 1, 0, 1}
{Idle, Responding, Responding, Idle, WaitingForResponse}
Evaluating Response Conditions between Station2 and Station3
Evaluating Response Conditions between Station3 and Station5
Slot Number 3
{{empty}, {empty}, {empty}, {empty}, {empty}}
StationRxQueue    {{empty}, {empty}, {empty}, {empty}, {empty}}
State        {Idle, Idle, Idle, Idle, Idle}
Attempts     {0, 0, 1, 0, 1}
Success      {0, 0, 1, 0, 1}
Failure      {0, 0, 0, 0, 0}
Retries      {0, 0, 0, 0, 0}
{Idle, Idle, Idle, Idle, Idle}
Slot Number 4
{{Station5, empty}, {empty}, {empty}, {empty}, {empty}}
StationRxQueue    {{empty}, {empty}, {empty}, {empty}, {empty}}
State        {Idle, Idle, Idle, Idle, Idle}
Attempts     {0, 0, 1, 0, 1}
Success      {0, 0, 1, 0, 1}
Failure      {0, 0, 0, 0, 0}
Retries      {0, 0, 0, 0, 0}
{Originating, Idle, Idle, Idle, Idle}
Evaluating TX Conditions between Station1 and Station5
Slot Number 5
{{Station5, Station2, empty}, {empty}, {empty}, {empty}, {empty}}
StationRxQueue    {{empty}, {empty}, {empty}, {empty}, {Station1, empty}}
State        {WaitingForResponse, Idle, Idle, Idle, Idle}
Attempts     {1, 0, 1, 0, 1}
Success      {0, 0, 1, 0, 1}
Failure      {0, 0, 0, 0, 0}
Retries      {1, 0, 0, 0, 0}
{WaitingForResponse, Idle, Idle, Idle, Responding}
Evaluating Response Conditions between Station5 and Station1
Slot Number 6
{{Station2, empty}, {empty}, {empty}, {empty}, {empty}}
StationRxQueue    {{empty}, {empty}, {empty}, {empty}, {empty}}
State        {Idle, Idle, Idle, Idle, Idle}
Attempts     {1, 0, 1, 0, 1}
Success      {1, 0, 1, 0, 1}
Failure      {0, 0, 0, 0, 0}
Retries      {0, 0, 0, 0, 0}
{Originating, Idle, Idle, Idle, Idle}
Evaluating TX Conditions between Station1 and Station2
Slot Number 7
{{Station2, empty}, {empty}, {empty}, {empty}, {Station4, empty}}
StationRxQueue    {{empty}, {Station1, empty}, {empty}, {empty}, {empty}}
State        {WaitingForResponse, Idle, Idle, Idle, Idle}
Attempts     {2, 0, 1, 0, 1}
Success      {1, 0, 1, 0, 1}
Failure      {0, 0, 0, 0, 0}
Retries      {1, 0, 0, 0, 0}
{WaitingForResponse, Responding, Idle, Idle, Originating}
Evaluating Response Conditions between Station2 and Station1
Evaluating TX Conditions between Station5 and Station4
Slot Number 8
{{empty}, {empty}, {empty}, {Station3, empty}, {Station4, empty}}
StationRxQueue    {{empty}, {empty}, {empty}, {Station5, empty}, {empty}}
State        {Idle, Idle, Idle, Idle, WaitingForResponse}
Attempts     {2, 0, 1, 0, 2}
Success      {2, 0, 1, 0, 1}
Failure      {0, 0, 0, 0, 0}
Retries      {0, 0, 0, 0, 1}
{Idle, Idle, Idle, Responding, WaitingForResponse}
Evaluating Response Conditions between Station4 and Station5
Slot Number 9
{{empty}, {empty}, {empty}, {Station3, empty}, {empty}}
StationRxQueue    {{empty}, {empty}, {empty}, {empty}, {empty}}
State        {Idle, Idle, Idle, Idle, Idle}
Attempts     {2, 0, 1, 0, 2}
```

```
Success       {2, 0, 1, 0, 2}
Failure       {0, 0, 0, 0, 0}
Retries       {0, 0, 0, 1, 0}
{Idle, Idle, Idle, Originating, Idle}
Evaluating TX Conditions between Station4 and Station3
Slot Number 10
{{Station2, empty}, {empty}, {empty}, {Station3, empty}, {empty}}
StationRxQueue     {{empty}, {empty}, {Station4, empty}, {empty}, {empty}}
State         {Idle, Idle, Idle, WaitingForResponse, Idle}
Attempts      {2, 0, 1, 1, 2}
Success       {2, 0, 1, 0, 2}
Failure       {0, 0, 0, 0, 0}
Retries       {0, 0, 0, 2, 0}
{Originating, Idle, Responding, WaitingForResponse, Idle}
Evaluating TX Conditions between Station1 and Station2
Evaluating Response Conditions between Station3 and Station4
Message Probabilities     {0.2, 0.2, 0.2, 0.2, 0.2}
Capture Margin      -20
Overall Efficency        0.75
Success Efficency        0.857143
Failure Efficency        0
```

```
Initialize
lasttraffic=StationTxQueue[#1]&[Stations];
Do[(MonitorRetries[lasttraffic,Stations];
      GenerateTraffic[Stations];
      lasttraffic=StationTxQueue[#1]&[Stations];
            Print["Slot Number ",n];
            Print[StationTxQueue[#1]&[Stations]];
            Print["StationRxQueue      ",StationRxQueue[#]&[Stations]];
            Print["State             ",State[#]&[Stations]];
            Print["Attempts          ",Attempts[#]&[Stations]];
            Print["Success           ",Success[#]&[Stations]];
            Print["Failure           ",Failure[#]&[Stations]];
            Print["Retries           ",RetryCount[#]&[Stations]];
            totaliterations=n;
            ExecuteStateMachine[Stations]),{n,1,10}]

Print["Message Probabilities    ",MessageProbability[#]&[Stations]]
Print["Capture Margin       ",CaptureMargin]
OverallEfficiency=N[Apply[Plus,Success[#]&[Stations]]/(totaliterations-2)];
Print["Overall Efficency        ",%]
SuccessEfficency=Apply[Plus,Success[#]&[Stations]]/Apply[Plus,Attempts[#]&[Stations]]//N;
Print["Success Efficency        ",%]
FailureEfficency=Apply[Plus,Failure[#]&[Stations]]/Apply[Plus,Attempts[#]&[Stations]]//N;
Print["Failure Efficency        ",%]
```

```
Slot Number 1
{{empty}, {empty}, {empty}, {empty}, {empty}}
StationRxQueue     {{empty}, {empty}, {empty}, {empty}, {empty}}
State         {Idle, Idle, Idle, Idle, Idle}
Attempts      {0, 0, 0, 0, 0}
Success       {0, 0, 0, 0, 0}
Failure       {0, 0, 0, 0, 0}
Retries       {0, 0, 0, 0, 0}
{Idle, Idle, Idle, Idle, Idle}
Slot Number 2
{{empty}, {empty}, {Station4, empty}, {Station1, empty}, {empty}}
StationRxQueue     {{empty}, {empty}, {empty}, {empty}, {empty}}
State         {Idle, Idle, Idle, Idle, Idle}
Attempts      {0, 0, 0, 0, 0}
Success       {0, 0, 0, 0, 0}
Failure       {0, 0, 0, 0, 0}
Retries       {0, 0, 0, 0, 0}
{Idle, Idle, Originating, Originating, Idle}
Evaluating TX Conditions between Station3 and Station4
Evaluating TX Conditions between Station4 and Station1
Slot Number 3
{{empty}, {empty}, {Station4, Station4, empty}, {Station1, empty}, {Station3,
empty}}
StationRxQueue     {{empty}, {empty}, {empty}, {Station3, empty}, {empty}}
State         {Idle, Idle, WaitingForResponse, WaitingForResponse, Idle}
Attempts      {0, 0, 1, 1, 0}
Success       {0, 0, 0, 0, 0}
Failure       {0, 0, 0, 1, 0}
Retries       {0, 0, 1, 1, 0}
{Idle, Idle, WaitingForResponse, Responding, Originating}
Evaluating Response Conditions between Station4 and Station3
Evaluating TX Conditions between Station5 and Station3
```

```
Slot Number 4
{{empty}, {empty}, {Station4, Station4, empty}, {Station1, Station5, empty},
{Station3, empty}}
StationRxQueue    {{empty}, {empty}, {empty}, {empty}, {empty}}
State          {Idle, Idle, Idle, Idle, WaitingForResponse}
Attempts       {0, 0, 1, 1, 1}
Success        {0, 0, 1, 0, 0}
Failure        {0, 0, 0, 1, 1}
Retries        {0, 0, 0, 2, 1}
{Idle, Idle, Originating, Originating, WaitingForResponse}
Evaluating TX Conditions between Station3 and Station4
Evaluating TX Conditions between Station4 and Station1
Slot Number 5
{{empty}, {Station3, Station5, empty}, {Station4, Station4, Station5, empty},
{Station1, Station5, empty},

  {Station3, Station1, empty}}
StationRxQueue    {{empty}, {empty}, {empty}, {Station3, empty}, {empty}}
State          {Idle, Idle, WaitingForResponse, WaitingForResponse, Idle}
Attempts       {0, 0, 2, 2, 1}
Success        {0, 0, 1, 0, 0}
Failure        {0, 0, 0, 2, 1}
Retries        {0, 0, 1, 3, 2}
{Idle, Originating, WaitingForResponse, Responding, Originating}
Evaluating TX Conditions between Station2 and Station3
Evaluating Response Conditions between Station4 and Station3
Evaluating TX Conditions between Station5 and Station3
Slot Number 6
{{empty}, {Station3, Station5, empty}, {Station4, Station4, Station5, empty},
{Station5, empty}, {Station3, Station1, empty}}
StationRxQueue    {{empty}, {empty}, {empty}, {empty}, {empty}}
State          {Idle, WaitingForResponse, Idle, Idle, WaitingForResponse}
Attempts       {0, 1, 2, 2, 2}
Success        {0, 0, 1, 0, 0}
Failure        {0, 1, 1, 2, 2}
Retries        {0, 1, 2, 0, 3}
{Idle, WaitingForResponse, Originating, Originating, WaitingForResponse}
Evaluating TX Conditions between Station3 and Station4
Evaluating TX Conditions between Station4 and Station5
Slot Number 7
{{empty}, {Station3, Station5, empty}, {Station4, Station4, Station5, empty},
{Station5, empty}, {Station1, empty}}
StationRxQueue    {{empty}, {empty}, {empty}, {Station3, empty}, {Station4,
empty}}
State          {Idle, Idle, WaitingForResponse, WaitingForResponse, Idle}
Attempts       {0, 1, 3, 3, 2}
Success        {0, 0, 1, 0, 0}
Failure        {0, 1, 1, 2, 2}
Retries        {0, 2, 3, 1, 0}
{Idle, Originating, WaitingForResponse, Responding, Responding}
Evaluating TX Conditions between Station2 and Station3
Evaluating Response Conditions between Station4 and Station3
Evaluating Response Conditions between Station5 and Station4
Slot Number 8
{{Station3, empty}, {Station3, Station5, empty}, {Station4, Station5, empty},
{empty}, {Station1, empty}}
StationRxQueue    {{empty}, {empty}, {empty}, {empty}, {empty}}
State          {Idle, WaitingForResponse, Idle, Idle, Idle}
Attempts       {0, 2, 3, 3, 2}
Success        {0, 0, 1, 1, 0}
Failure        {0, 2, 2, 2, 2}
Retries        {0, 3, 0, 0, 1}
{Originating, WaitingForResponse, Originating, Idle, Originating}
Evaluating TX Conditions between Station1 and Station3
Evaluating TX Conditions between Station3 and Station4
Evaluating TX Conditions between Station5 and Station1
Slot Number 9
{{Station3, empty}, {Station5, empty}, {Station4, Station5, empty}, {Station1,
empty}, {Station1, empty}}
StationRxQueue    {{empty}, {empty}, {Station1, empty}, {empty}, {empty}}
```

```
State       {WaitingForResponse, Idle, WaitingForResponse, Idle,
WaitingForResponse}
Attempts    {1, 2, 4, 3, 3}
Success     {0, 0, 1, 1, 0}
Failure     {0, 2, 3, 2, 3}
Retries     {1, 0, 1, 0, 2}
{WaitingForResponse, Originating, Responding, Originating, WaitingForResponse}
Evaluating TX Conditions between Station2 and Station5
Evaluating Response Conditions between Station3 and Station1
Evaluating TX Conditions between Station4 and Station1
Slot Number 10
{{Station3, empty}, {Station5, Station4, empty}, {Station4, Station5, Station4,
empty}, {Station1, empty},

  {Station1, Station1, empty}}
StationRxQueue    {{empty}, {empty}, {empty}, {empty}, {empty}}
State       {Idle, WaitingForResponse, Idle, WaitingForResponse, Idle}
Attempts    {1, 3, 4, 4, 3}
Success     {0, 0, 1, 1, 0}
Failure     {1, 3, 3, 3, 3}
Retries     {2, 1, 2, 1, 3}
{Originating, WaitingForResponse, Originating, WaitingForResponse, Originating}
Evaluating TX Conditions between Station1 and Station3
Evaluating TX Conditions between Station3 and Station4
Evaluating TX Conditions between Station5 and Station1
Message Probabilities   {0.2, 0.2, 0.2, 0.2, 0.2}
Capture Margin      12
Overall Efficency      0.25
Success Efficency      0.111111
Failure Efficency      0.833333
```

■ Conclusion

This document has presented an accurate framework for evaluating MAC/PHY performance. This framework is flexible enough to handle many differing simulation needs and scenarios. By way of example, a very brief picture (insuffcent in detail) of the fact that Capture Effect can not be ignored in simulations of throughput efficency has been shown. The requested action from this submittal is to close issue 29.1 and to direct the parties working on channel models and MAC/PHY throughput presentations to begin working with a common set of tools developed in *Mathematica*.