The mLab-PENGWUN Hybrid Emulation Environment for Airborne Networks

Marco Carvalho, Adrian Granados, Marco Arguedas and Carlos Perez
Florida Institute for Human and Machine Cognition (IHMC)
Ocala, FL
{mcarvalho,agranados,marguedas,cperez}@ihmc.us

Michael Muccio, Joe Suprenant, Daniel Hague and Brendon Poland
Air Force Research Laboratory (AFRL)
Rome, NY

Abstract—In this paper we introduce a hybrid emulation infrastucture that combines theoretical path loss models with data-driven statistical link models. The goal is to provide a specialized emulation environment for the development and validation of airborne networking protocols, in support of the AFRL’s PENGWUN research effort. After a brief discussion of our architecture design and policy enforcement mechanisms, we discuss our approach for data-driven link emulation for airborne networks.

Keywords-component: Airborne Networks, Wireless Emulation, Statistical Link Modeling, Airborne protocol design, mLab, PENGWUN

I. INTRODUCTION

Modeling and simulation have always been an important part of wireless protocol design, test and evaluation. A significant part of the early MANET research was based on computer simulations of the physical environment and wireless links. As researchers began to question the accuracy and validity of simulated environments for protocol design [6], emulated environments with more advanced and precise link models started to appear [1][5], playing a very important role in the advances of the research in the field.

In most simulation (or emulation) environments, theoretical data link models are used to recreate the effects of the physical environment. The effectiveness and accuracy of the simulation is directly affected by the accuracy of such models, and the simulation approach. For example, the Common Open Research Emulator (CORE) [7] provides capabilities for emulation and modeling mainly of level 3 network characteristics. Its primary intent is to facilitate the development of routing algorithms and therefore the physical and link layer models are quite simplistic for modeling wireless networks. Consequently, more sophisticated models are provided through an integration with the Extendable Mobile Ad-Hoc Network Emulator (EMANE) [1].

This research has been sponsored by the United States Air Force Research Laboratory under contract FA8750-10-2-0145.
that network latency at the physical level must be significantly smaller than the required emulated latencies, and the capacity at the physical level must be significantly larger than the capacity of the emulated system.

Testbed nodes, in this configuration, have two interfaces, one connected to a data-only high bandwidth switched network and the other one connected to a control network, which is used for the control and monitoring of link behavior and network policies. In our current version of mLab, the physical infrastructure has been replaced by a virtualized environment where the controller can be dynamically connected with emulation nodes running as virtualized machines in a cloud environment. This new design for the “physical” infrastructure is functionally similar to the original concept.

B. The Modeling Component

The modeling component is the software responsible for the modeling of physical links and node behaviors. As a modular component, it can be replaced to represent any arbitrary physical link using theoretical propagation models, data-driven statistical models, or a combination of them. As part of our joint research with AFRL, we have built statistical models from actual flight tests that are to be mixed with theoretical propagation and loss models.

C. The Coordination Component

A controller node (referred to as controller) holds a model of the network topology represented as a graph structure. Each vertex in the graph maps to a specific host in the testbed. Based on theoretical propagation and interference models, the controller continually estimates the physical topology and the characteristics of each communications link. For example, in the case of the enforcement performed by virtual drivers at each emulation node, the characteristics that define a communications link are only two, a probability distribution for packet loss and a probability distribution for packet delay. For each link in the graph, the controller maintains four distributions, two (delay and packet loss) for each direction. As nodes move, and traffic pattern changes, the distributions are then adjusted based on the controller’s models. The parameters of each distribution are then used to configure the actual data links (blue network) between nodes.

The way that the enforcement is done is defined in Figure 1. Basically, the controller continuously monitors the position and other attributes of the nodes to re-calculate and parameterize the models and enforcement components when network conditions change.

D. The Enforcement Component

This is a distributed software component that exists both at the controller node and at each host in the testbed. It provides coordination of policies related to current link conditions and communication constraints, and provides the enforcement capabilities (at the lowest possible levels) of packet loss and delays for the emulation. In our current implementation, the enforcement infrastructure is provided through customized virtual network drivers, directly interfacing with the controller node, or through EMANE, which provides pluggable Medium Access Control (MAC) and physical (PHY) layers for emulation of heterogeneous networks with varying wireless technologies.

III. MANAGEMENT AND VISUALIZATION

In mLab, emulations are managed and visualized using MView (Figure 2). MView provides a control interface for the mLab controller that allow users to import the definition of a emulation scenario (Figure 3).

When a scenario is loaded in mLab as part of an experiment, the controller automatically assigns the required number of testbed nodes and configures them with the specified network interfaces. If not enough testbed nodes are available, the operation fails and the user is informed. Note, however, that there is not a direct mapping between testbed nodes and emulation nodes. In fact, multiple runs of the same scenario may use a different set of testbed nodes. Therefore, if the same testbed node must always be assigned to a particular emulation node (e.g. a testbed node has been configured with some specific software), then the user may specify such condition in the scenario file. In this case, if the specified testbed node is not available, then the scenario cannot be loaded.
Additional physical or virtual testbed nodes can always be added to mLab. Nodes that have been loaded with the mLab software can automatically find and register with the controller as part of the mLab testbed. Any further configuration that is necessary to provide link enforcement will be deployed automatically when the node is assigned to an emulation scenario.

mLab nodes are also configured with a Network File System (NFS) that allows users to have a shared home directory. This feature simplifies the deployment and configuration of applications used in an experiment. It also facilitates the retrieval of results or log data after the experiment has concluded. Nonetheless, users are responsible for ensuring that applications will work correctly (e.g. they will not overwrite output files) when using a shared directory.

Figure 3. Listing of an mLab scenario file.

Once a scenario is loaded by the controller, the user can connect to it through MView and see the 3D world representation of the location of each emulation node and the connections (links) between them. Furthermore, the user can see real-time updates of node and link properties such as drop rates, delay times, location and orientation. In some cases, the user is also able to modify these properties, including the location and orientation or nodes in the scenario.

IV. EMANE INTEGRATION

EMANE is an infrastructure used for emulation of simple as well as complex heterogeneous mobile ad-hoc networks [1]. It supports pluggable MAC and PHY layers that allow for the emulation of commercial and tactical networks with multiple tiers and varying wireless technologies. It also supports multiple platforms (Windows, Linux and OS X) and provides mechanisms to setup small as well as large-scale emulations using centralized, distributed or virtualized deployments.

EMANE provides a modular architecture with well-defined APIs to allow independent development of emulation functionality for different radio models (network emulation modules), boundary interfaces between emulation and applications (transports), and distribution of emulation environmental data (events).

Each node in the emulation is represented by an instance of an emulation stack, which is comprised of three components: transport, Network Emulation Module (NEM), and Over-the-Air (OTA) Manager. These set of components encapsulate the functionality necessary to transmit, receive, and operate data routed through the emulation space. In particular, the transport component is responsible for handling data to and from the emulation space and interfaces with the underlying operating system using different platform-dependent mechanisms such as TunTap (Linux, OS X) and WinTap (Windows). On the other hand, a NEM provides emulation functionality for the MAC and PHY layers, including CSMA, TDMA, queue management, and adaptive transport protocols for the MAC layer, and waveform timing, half-duplex operations, interference modeling, probability of reception, out-of-band packets, and more for the PHY layer. Finally, the OTA Manager provides the necessary messaging infrastructure to deliver emulation radio model data to all nodes participating in the emulation. Control messages from the OTA Manager are distributed to each NEM through a multicast channel.

In EMANE, each NEM and its corresponding MAC and PHY components are responsible for the enforcement of link characteristics. Hence, packets in the emulation space contain a special EMANE identification header and it is up to each layer to decide whether a packet is dropped or delayed before passing it to the upper layers. Each NEM requires a base configuration that specifies the network emulation model and the parameter values to configure the capabilities provided by the MAC and PHY components of the model. This configuration dictates the behavior of the emulation stack, which can be modified through the dissemination of emulation environmental data, such as location information or path loss, to emulate the dynamics of the network (e.g. node movement). This data can be distributed to NEMs in real time from the EMANE Event Service using a multicast channel, similarly to how the OTA Manager distributes control messages.

The type of environmental data that can be used to change communication constrains and how this data affects the behavior of the emulation depends on the configured network emulation model of the NEMs. For example, the IEEE 802.11 abg model, which emulates IEEE 802.11 MAC layer’s Distributed Coordination Function (DCF) channel access scheme and IEEE 802.11 Direct Spread Spectrum Sequence (DSS) and Orthogonal Frequency Division Multiplexing (OFDM) signals in space, supports path loss as a type of emulation environmental data that can be utilized to change link emulation conditions between IEEE 802.11 or RFPipe NEMs. When the IEEE 802.11 abg or the RFPipe models receive a Pathloss event, which contains a variable list of path loss and reverse path loss values between the receiving NEM and one or more NEMs in the emulation, the model’s PHY layer changes the characterization of the links to other NEMs based on the specified path loss values.

The integration between mLab and EMANE takes advantage of EMANE network emulation models and event functionality to specify communication constrains and enforce
link conditions between mLab emulation nodes. In mLab, emulation modes can be configured with multiple network interfaces over one or more different media (channels). In the case of EMANE, a medium corresponds to a network emulation model, such as IEEE 802.11 abg, which was discussed previously, or RF Pipe, a model that provides low fidelity emulation of a variety of waveforms. The fact that EMANE is a highly configurable emulation platform also makes it one of EMANE’s drawbacks. Configuring one (or multiple) scenarios in EMANE, where each node may have different parameters for the emulation, may result in a cumbersome task, as potential changes in scenario characteristics could require modification of several XML files in multiple physical emulation nodes. mLab helps overcome this limitation by providing a more simple and more agile mechanism for configuring a testbed with EMANE.

When the mLab controller loads a scenario file, the EMANE enfocer component in the controller generates the necessary EMANE configuration files for the physical nodes that will take part in the emulation. Then, these XML configuration files are deployed to create and configure a NEM with the given network emulation model (e.g. IEEE 802.11 abg) to emulate the desired medium. Note that each medium must be configured with the name the class responsible for the deployment and configuration of one of the EMANE’s supported models (currently IEEE 802.11 abg, RF Pipe and CommEffect). In addition, mLab makes available a subset of the most relevant configuration parameters of each EMANE’s network emulation model that can be specified through the definition of the medium in the mLab scenario file. Once the required EMANE’s configuration files are deployed and the EMANE services are started on each of the emulation nodes, link characterization is performed by generating the appropriate EMANE events for each of the configured network emulation models (Figure 4).

Figure 4. MLAB-EMANE Integration

For example, in the case of a medium that uses the IEEE 802.11 abg model, EMANE Pathloss events are generated each time there is change in node position. The pathloss values are derived from mLab’s theoretical or data-driven propagation model implementations.

V. STATISTICAL LINK MODELING

We have developed and validated a statistical data link model to predict link characteristics from the PENGWUN (AFRL’s Protocol Emulation for Next Generation Wireless UAS Networks) experimental data, which consists on measurements of different platform, network and communication parameters during the transmission of data between a UAV (sender) and a ground node (receiver). In the remaining of this section we describe two different approaches to model the link characteristics based on certain conditions of the nodes in the test data. We also describe our strategy for model validation and conclude with experimental results.

A. Model Development

The PENGWUN experimental data includes the positions of the ground node and the UAV, the yaw, pitch and roll of the UAV, the RSSI and the packet loss. The variables that most likely affect the RSSI are the distance and the roll or banking of the UAV. In one case, the signal is attenuated with the distance, and in the other case, the ground antenna and the UAV antenna lose line of sight when the aircraft is banking. In most cases, the theoretical models only consider the distance, however, our empirical model considers both distance and angle as dimension for estimating the RSSI.

Figure 5 shows the scatter plots of the distance vs. RSSI and the banking angle vs. RSSI for one of the flights. According to these graphs, the distance seems to be a better predictor for the RSSI than the angle, however, notice that at certain points (200m and 400m) the RSSI varies considerably. If at these points we look at the angle (Figure 6), this high variability can be explained. The variability for these distances occurs when the banking angle changes from 0 to 40 degrees.

Figure 5. Distance and angle as RSSI predictors

Using the experimental data, two types of models are used for estimating the RSSI. First, an empirical model is constructed using non-parametric estimation. The advantage of this approach is that we can introduce as many dimensions as necessary without knowing the real distribution of the data. The main disadvantage of this approach is that the convergence to true values is slower with respect to the number of data points used for the estimation.

The second type of model is a parametric model based on the log-normal shadowing propagation model [2]. For this model there are two parameters that need to be estimated from the data: the path-loss exponent and the variance. The main advantage of this model is that the convergence is much faster. The main disadvantages are that we can only use the dimensions defined by the model for the estimation (i.e. distance), and that we are assuming that the distribution of the experimental data is equal to the one proposed by the model.
1) Empirical Model

For the empirical model, two predictor variables are used to predict the RSSI: the distance and the banking angle of the airplane, also known as the roll. The Cartesian plane formed by these two dimensions is divided in a grid with squares of equal size. Given a square of the grid, the RSSI for that square is modeled using the empirical distribution function, which is commonly used as a non-parametric method of estimating the cumulative distribution function (cdf) of an unknown distribution. Let \((x_1, x_2, \ldots, x_n)\) be data points sampled from the common unknown distribution function.

The cumulative distribution function of the Empirical distribution is defined by:

\[
F(t) = \frac{1}{n} \sum_{i=1}^{n} I(x_i \leq t),
\]

where \(I(x_i \leq t)\) is the indicator function, defined by:

\[
I(x_i \leq t) = \begin{cases} 
1 & \text{if } x_i \leq t \\
0 & \text{otherwise}.
\end{cases}
\]

Having modeled the RSSI for each square in this way, we can now generate pseudo-random numbers drawn from this distribution by generating a \(X \sim U(0, 1)\), and then computing \(F^{-1}(X)\). So, to estimate the RSSI for a given distance and banking angle, we find the corresponding square, and the use the empirical distribution function to randomly generate a RSSI value from that distribution.

2) Theoretical Model

The theoretical model we have considered is the log-normal shadowing model, which is the same probabilistic model used by the network simulator NS-3 [3]. This model assumes that the average received signal power decreases logarithmically with distance, and that the path loss is randomly distributed log-normally (normal in dB) about that mean. The equation describing the path loss at a given distance is:

\[
PL(d) = PL(d_0) + 10 \cdot \beta \cdot \log\left(\frac{d}{d_0}\right) + X_\sigma,
\]

where \(PL(d_0)\) is the average path loss at the close-in reference distance which is based on measurements close to the transmitter or on a free space assumption from the transmitter at distance \(d_0\), \(\beta\) is the path loss exponent which indicates the rate at which the path loss increases with distance, and \(X_\sigma\) is a zero-mean Gaussian distributed random variable (in dB) with standard deviation \(\sigma\) (also in dB). Using the free space assumption, the average path loss at the close-in reference distance can be estimated using the following equation:

\[
PL(d) = -20 \cdot \log\left(\frac{\lambda}{4\pi d}\right),
\]

where \(\lambda\) is the wavelength of the carrier signal.

To be able to use the log-normal shadowing model, we need to estimate two parameters from the experimental data: the path-loss exponent, and the standard deviation of the random variable. These two parameters can be estimated by doing a linear regression of the path-loss using as predictor the logarithm of the distance. The path-loss in the experimental data for any given point can be computed using the following formula

\[
P_{\text{loss}} = P_t - P_r,
\]

where \(P_t\) is the transmitted power and \(P_r\) is the received power, both in dBm.

The slope of the regression line can be used as estimate of the path-loss exponent by dividing the slope by 10, and the variance obtained from the regression can be used as an estimate of the variance of the random variable. For example, using the data from the same flight shown in Figure 5, the path-loss is 0.127 and the deviation is 0.25. If we generate RSSI points using the log-normal shadowing model with these parameters, and for different distances we get the results shown in Figure 7.
B. Model Validation

Validating the models means making sure that data generated by the models resembles the distribution of the experimental data. Using a goodness-of-fit test would only be appropriate for validating the theoretical model, because the empirical model is non-parametric. It would be ideal to use a validation method that would allow us to compare what model fits better. With this purpose in mind, we propose the following validation process. Let \( \{(d_1, r_1, p_1), (d_2, r_2, p_2), \ldots (d_n, r_n, p_n)\} \) be the testing data, composed of distance \( d \), roll angle \( r \) and RSSI \( p \) tuples. For each of the models to validate, we generate points \( \{(d_1, r_1, \theta_1); (d_2, r_2, \theta_2)\ldots(d_n, r_n, \theta_n)\} \), where \( \theta \) is the estimated RSSI value from the model being validated. Dividing the testing and estimated data in the same fashion as it is divided in the empirical model (a grid divided by distance and angle), we can compute the average for each of the squares in the grid and then compute the relative error between the average of the testing data and the average for the estimated data for the same square in the grid. The errors for the errors of the squares in the grid can later be combined using a simple average, providing a single estimate of the error for the estimated model for a given test set. If we then test the models using multiple test sets, we can get a better estimate of the error by averaging the error obtained for each of the test sets.

1) Cross Validation

Ideally, the best way to test the models would be to use data from a different flight, but given that different flights may take place under different conditions, the estimates of the error for each of the models might be misleading. On the other hand, using the data that was used for estimating the models, would not be correct either. A better way to get the data to validate the models is to divide the experimental data in a training set and a testing set. Therefore, we use the training set for estimating the models and then we use the testing set for validation. The main disadvantage with this approach is that the error estimates might be very dependent on the way the data was split. To fix this problem we can split the data randomly in \( k \) different ways, and then use each of the \( k \) training and testing sets to estimate the errors, then averaging the error estimates for each of the \( k \) training and testing set pairs should give us a better estimate of the error. This type of validation is also known as \( k\)-fold cross validation [4].

2) Experimental Results

To validate and compare the accuracy of both the theoretical and empirical models in predicting the position of two nodes (i.e., an UAV and a ground node), we have used test flights from the PENGWUN dataset. We have selected flights for which we have information on the characteristics of the transmitter, in particular the frequency and transmission power. Then, using the specified transmission power and transmitting frequency, and using the RSSI values specified on each data point in the flight, we have computed the value of the Path-loss parameter \( PL \) needed for the shadowing propagation model. With the calculated \( PL \) value in hand, we are now able to compute the theoretical estimated value of RSSI between the nodes of the given dataset. Then, using \( k\)-fold cross-validation with \( k = 10 \) we test out the accuracy of the theoretical model in predicting actual observed values of RSSI.

At the same time, we use the flight link information to feed the empirical model described earlier. To do this, we compute the distance and antenna angle between the two nodes in the link, and binning the distance in increments of ten meters and the antenna angles in increases of one decimal degree, we train our empirical model. For each iteration of the \( 10\)-fold validation and for both the empirical and theoretical models, we have used 90% of the data for training the models and the remaining 10% for validating. Table 1 shows the experimental results of both the theoretical and empirical models. In all cases, the empirical model was able to successfully predict the RSSI values of each link with an error rate that is a fraction of the error rate of the theoretical model.

<table>
<thead>
<tr>
<th>Flight No.</th>
<th>Source Node</th>
<th>Target Node</th>
<th>Empirical Model Error</th>
<th>Theoretical Model Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>UAV</td>
<td>GN1</td>
<td>9%</td>
<td>57%</td>
</tr>
<tr>
<td>1</td>
<td>UAV</td>
<td>GN2</td>
<td>18%</td>
<td>47%</td>
</tr>
<tr>
<td>2</td>
<td>UAV</td>
<td>GN1</td>
<td>7%</td>
<td>53%</td>
</tr>
<tr>
<td>2</td>
<td>UAV</td>
<td>GN2</td>
<td>17%</td>
<td>48%</td>
</tr>
<tr>
<td>3</td>
<td>UAV</td>
<td>GN1</td>
<td>11%</td>
<td>56%</td>
</tr>
<tr>
<td>3</td>
<td>UAV</td>
<td>GN2</td>
<td>16%</td>
<td>57%</td>
</tr>
</tbody>
</table>

VI. CONCLUSIONS

We have developed a hybrid emulation environment that integrates a highly flexible emulation infrastructure with the capabilities of EMANE to support the emulation of airborne networks. Our approach for a bi-dimensional statistical data-driven propagation model has shown to provide better estimates for RSSI values of airborne links than the theoretical propagation models that were considered for comparison in this work. Future work includes the integration of mLab with the PENGWUN database for online generation of data-driven propagation models that can be utilized to emulate specific flight tests as a way to facilitate the development and testing of protocols and applications for airborne networks using different hardware configurations.

REFERENCES