SAND REPORT

SAND2004-0885
Unlimited Release
Printed March 2004

High Level Architecture (HLA) Federation with Umbra and OPNET Federates

Brian P. Van Leeuwen, Fred Oppel III, and Brian Hart

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under Contract DE-AC04-94AL85000.

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865)576-8401
Facsimile: (865)576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: http://www.doe.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone: (800)553-6847
Facsimile: (703)605-6900
E-Mail: orders@ntis.fedworld.gov
Online order: http://www.ntis.gov/ordering.htm

SAND2004-0885
Unlimited Release
Printed March 2004

High Level Architecture (HLA) Federation with Umbra and OPNET Federates

Brian P. Van Leeuwen
Networked Systems Survivability & Assurance Department

Fred Oppel III and Brian Hart
Intelligent Information Systems Department

Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185-0785

Abstract

Network-centric systems that depend on mobile wireless ad hoc networks for their information exchange require detailed analysis to support their development. In many cases, this critical analysis is best provided with high-fidelity system simulations that include the effects of network architectures and protocols. In this research, we developed a high-fidelity system simulation capability using an HLA federation. The HLA federation, consisting of the Umbra system simulator and OPNET Modeler network simulator, provides a means for the system simulator to both affect, and be affected by, events in the network simulator. Advances are also made in increasing the fidelity of the wireless communication channel and reducing simulation run-time with a dead reckoning capability. A simulation experiment is included to demonstrate the developed modeling and simulation capability.

This page intentionally left blank.

Contents

Figures

Tables

1. Introduction

Timely and robust information exchange is a major challenge for many defense and security network-centric systems. These network-centric systems require survivable, secure, and efficient wireless networks that enable flows of information between participating mobile nodes. Wireless network components will be mounted on moving air, sea, and land platforms as well as individual soldiers and fielded equipment such as sensors and will number in 100s to 1000s of mobile nodes that form dynamic network topologies. Development of the wireless networks is further complicated by the resource-constrained nature of typical mobile nodes. Operation environment impacts, both natural and adversarial, have significant influence on network operation. Adversarial forces may attempt to gather intelligence by monitoring data and data streams and attempt to disrupt the networks through hacking, jamming, and inserting false information streams into the networks. In cases, networks built from recently developed technologies must operate with legacy communication networks. Thus development of effective wireless networks requires detailed analysis of the required operating parameters, node and network resources, and operating environment.

Analysis of mobile wireless networks requires consideration of all aspects of their inherent non-linear relationships that influence network operation. The aspects include networking protocols, security methods, operation environments, controls, sensors, and applications. Wireless ad hoc routing protocols and medium access control (MAC) mechanisms that make up networks are receiving considerable attention from military, commercial industry, universities, and Internet Engineering Task Force (IETF) Working Groups. Security protocols and methods to provide information assurance (IA) have been and continue to be developed at Sandia Labs and other research groups. Operation environments including terrain, weather, and adversarial stressors will have significant impact on network performance and must be considered. As the networking protocols are combined to form an operational network it is necessary to evaluate the network performance under the expected operation scenarios. Detailed analysis of network architectures and supporting protocols with the application and environment impacts is necessary for effective network design and to understand the network effects on the overall system operation.

An effective means to analyze network-centric systems is to run realistic simulations. This provides a means to examine the performance of the network and to examine the overall system operation. Simulation studies provide a means to collect data necessary to aid in the analysis, including tradeoff studies of large complex wireless networks. Modeling and simulation provide the designer with a tool to represent a wide range of operating environments and application scenarios. The result of the research presented in this report is an effective solution to support system level modeling and simulation that incorporates the effects of the mobile wireless networks on system operation.

The remainder of the research report is organized as follows. A detailed discussion of the High Level Architecture (HLA) federation development is provided in Section 2. In Section 3, we present a number of developments that enable higher fidelity simulations. In Section 4, a demonstration simulation experiment and results are described. Future work and a conclusion are presented in Sections 5 and 6.

2. System Modeling and Simulation Methodology

Our approach to solving the simulation challenges begins with interoperating multiple simulators using the High Level Architecture (HLA). The distributed simulation approach permits specialized simulators to accurately portray the physical behavior, network behavior, and other aspects of the simulated world. Our solution uses the Sandia National Laboratories developed Umbra system-level simulator to model and capture the behaviors of many of non-network related components, such as mobile nodes that can represent cell phones, sensors, soldiers, tanks, ships, aircraft, or satellites. Umbra models platform attributes such as location, orientation, motion profiles, and platform capabilities and is used to evaluate system effects of intelligent agents in heterogeneous simulations involving behaviors, controls, mobility and terrain. Our approach incorporates a network simulator, OPNET Modeler [13], to accurately model and simulate the networking issues. With the distributed simulator approach, Umbra retains its behavioral focus of the physical world while OPNET addresses the communications network effects.

2.1 System Simulator

The system level or application simulator used for this research is Umbra, a Sandia National Laboratories developed software architecture. Umbra is modular software framework [8, 9, 10] that enables constructing and analyzing complex systems. Umbra provides realistic physical environments to study the behaviors of agents, such as autonomous robots that operate either as a single entity or as a collection of entities to perform high-level system tasks. Umbra uses a world module to encapsulate the physical environments such as terrains, weather, plumes, and vehicle dynamics. Umbra's world module enables heterogeneous physics to co-exist in the same simulation environment and share data between worlds in a loosely coupled relationship. System designers represent their system components as Umbra modules. These Umbra modules represent the basic system functionality such as sensors, controls, and behaviors. A collection of these modules are connected together to form a meta-module. Figure 1 illustrates a meta-module that provides the desired agent functionality with forward and feedback connections as shown (feedback connectors are shown with a circle-F symbol). Umbra calls the module's update methods based on the connection graph in order to synchronize the inputs to outputs. Data typically flows from left to right except for feedback connections

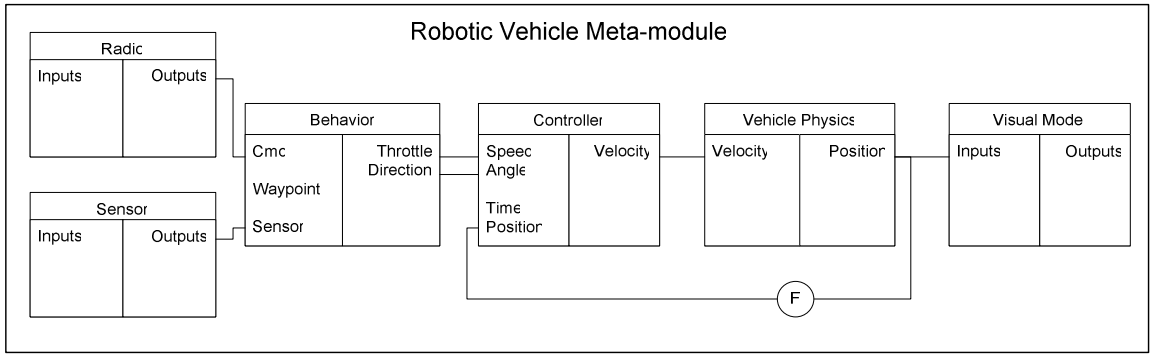


Figure 1: Example collection of Umbra modules

In addition, Umbra provides the system designer with the ability to *computationally steer* the simulation by dynamically creating/deleting modules, reconfiguring module connections, or adjusting module parameters during the simulation execution. Umbra can be executed with or without a 3D visualization environment.

2.2 Communication Network Simulator

The communication network simulator used for this research is the OPNET Modeler and Radio simulation package Version 10.0.A. Optimum Network Performance (OPNET) [13] is a comprehensive engineering system capable of simulating large communication networks with detailed protocol modeling and performance analysis capability. OPNET features include: graphical specifications of models; a dynamic, event-scheduled simulation kernel; integrated tools for data analysis; and hierarchical, object-based modeling. OPNET analyzes system behavior and performance with a discrete-event simulation engine. Discrete-event simulation is an approach that supports realistic modeling of complex systems that can be represented as a progression of related events. This approach models system behavior based on objects and distinct events such as the arrival of packets at various points in a network. Each object has associated attributes that control its behavior in the simulation.

In OPNET, a node is a collection of interconnected modules in which data is manipulated as defined by the modules. Modules represent the internal capabilities of a node such as data creation, transmission, processing, internal routing, reception, storage and queuing. The modules are used to model aspects of node behavior. A single node model is usually comprised of multiple modules. The modules are connected together by packet streams and statistic wires. Packet streams are used to transport data between the modules while statistic wires allow one module to monitor a varying quantity within another module. The ability to integrate the use of modules, packet streams, and statistic wires allows the developer to create highly realistic simulations of node behavior.

Each node module contains a set of inputs and outputs, some state memory, and a method for computing the module's outputs from its inputs and its state memory. OPNET provides a module library that represents standard functions and protocols. To support development activities, the modules are open source and can be customized to represent user-defined behavior. In addition, completely new modules can be created to represent user developed functions and protocols. Example OPNET modules are processors, queues, generators, receivers, and transmitters. The processor, queue, and generator modules are strictly internal to a node. The transmitters, receivers, and antennas have external connections to data transmission links.

For wireless communication network simulation experiments, OPNET includes the effects of the wireless channel, such as path loss, channel interference, and bit-error-rates with pipeline models. The standard pipeline models did not, however, include a fading model. Since our targeted applications are expected to operate in environments where channel fading is present we developed a model to include the effects of fading. The fading model is implemented by the

OPNET federate and uses line-of-sight information generated by the Umbra Line-of-Sight federate as described in Section 3.

2.3 High Level Architecture (HLA) Federation with Umbra and OPNET Federates

The co-simulation approach allows complex communication protocols and network architectures to be simulated with a network simulator such as OPNET while being offered realistic traffic and operating environment conditions provided by Umbra. This co-simulation approach allows for the simulation of system operation with the impacts of realistic communication network operation. This method results in an analysis capability that can provide more realistic system results than when a single simulation capability is used that typically abstracts some part of the overall system model to a much lower fidelity. For example, Umbra contains a communications world module that provides an abstraction model of the transport, network, and hardware layers while OPNET typically uses an abstraction model (a stochastic approach) of the application layer. In general, the co-simulation approach provides a means for specialized simulators to accurately portray the physical behavior, network behavior, and other aspects of the simulated world. With this distributed simulator approach, Umbra can retain its behavioral focus of the physical world and OPNET can address the communications network effects.

Our co-simulation approach uses High Level Architecture (HLA) [5, 6] to create an Umbra/OPNET federation. HLA is an architecture for creating computer models and combining simulations from component models and sub-simulations. The HLA federation architecture provides a very efficient computational approach when combining pure event simulators (e.g., communication events modeled by OPNET) with time-stepping simulators (e.g., platform motion modeled by Umbra). Event simulators are not efficient at modeling time-stepping algorithms, while time-stepping simulators usually group closely occurring time-based events within the same time slot resulting in a lower fidelity and less efficient simulation. The enabling feature within HLA to coordinate these two different types of simulators is the time management service discussed in Section 2.3.3. Umbra and OPNET support the Defense Simulation and Modeling Office's (DMSO) Run Time Infrastructure (RTI) Next Generation 1.3 (NG1.3) library for HLA services. An HLA simulation, called a federation, is made up of numerous federates executing in a distributed computing environment

In our co-simulations, Umbra provides the system level simulator to model and capture the behaviors of many of non-network related components, such as mobile nodes that can represent cell phones, sensors, soldiers, tanks, ships, aircraft, or satellites. OPNET Modeler provides the network simulator to accurately model and simulate the networking issues. Currently we employ three federates for our Umbra/OPNET federation. They are an Umbra Platform federate, OPNET Modeler federate, and an Umbra *line-of-sight* (LoS) federate as shown in Figure 2. The Umbra Platform federate provides the robotic behaviors of the platforms. The Umbra LoS federate provides the *LoS* terrain effects for the communication analysis.

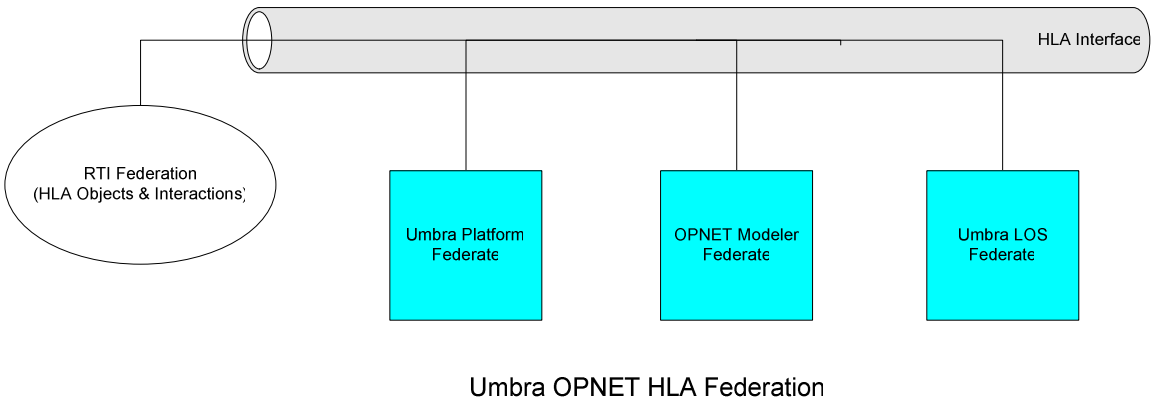


Figure 2: Umbra/OPNET federation used in our modeling and simulation study

Figure 3 illustrates a pair of node models in OPNET that represent the various protocols and interactions that support network communications. During the co-simulation, if a communication node simulated in the Umbra Platform federate requires a data transfer to another communication node, the Umbra Platform federate sends the necessary details of this communication to the OPNET Modeler federate via HLA. The OPNET simulator directs the data to the appropriate node. This node receives the data and creates a message and if necessary, initiates a TCP connection with the intended destination. The message is acted upon by the various protocols and transmitted to the destination node. Upon receipt at the destination node, the destination node unpacks the message and submits the data to the Umbra Platform federate via an HLA interaction. In the co-simulation, the effects of physical layer collisions, interference, and noise are all included.

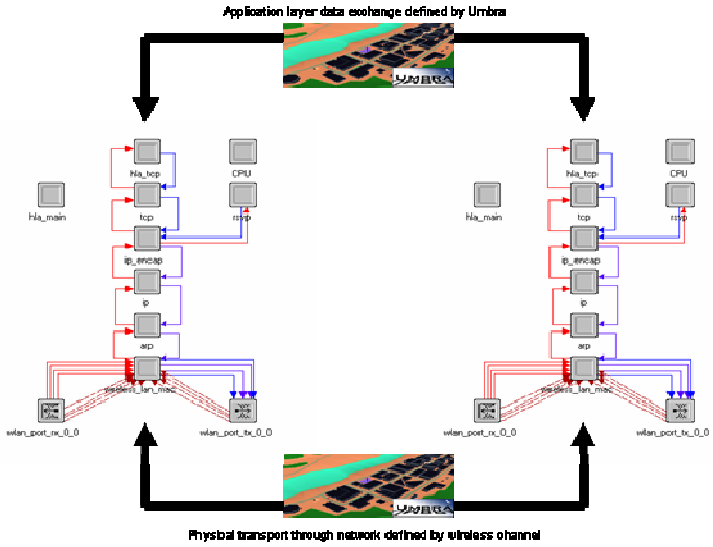


Figure 3: Communication protocols represented in OPNET and Umbra

2.3.1 Umbra HLA Federate

An Umbra World module, also referred to as the Ambassador module, provides the interface to the RTI library and is a factory of the Umbra HLA modules that correspond to the HLA objects and interactions. An Umbra HLA federate consists of:

- Ambassador module,
- Regular modules providing the federate functionality,
- HLA modules acting as proxies to the HLA objects in the federation.

The Ambassador module is responsible for managing all the HLA data entering or leaving the Umbra HLA federate. During an ambassador module update, the module checks for incoming HLA data and dispatches the data to the appropriate HLA module. Next, it will update its HLA modules in order to send or publish the data to other federates. The Umbra HLA modules handle HLA interactions between the federates.

A generic interface between Umbra and RTI-NG1.3 is described in [9]. This interface enables Umbra-based models to be federated into HLA environments and interface to DMSO's RTI NG1.3 software library. An HLA federation is made up of numerous federates executing in a distributed computing environment. Federates provide simulation services for the federation by communicating with each other through HLA objects (persistent data) and interactions (non-persistent data). The Federation Object Model (FOM) provides descriptions of the objects and interactions that exist in the federation. When a federate joints a federation, the federate informs the RTI which objects and interactions it will publish and subscribe.

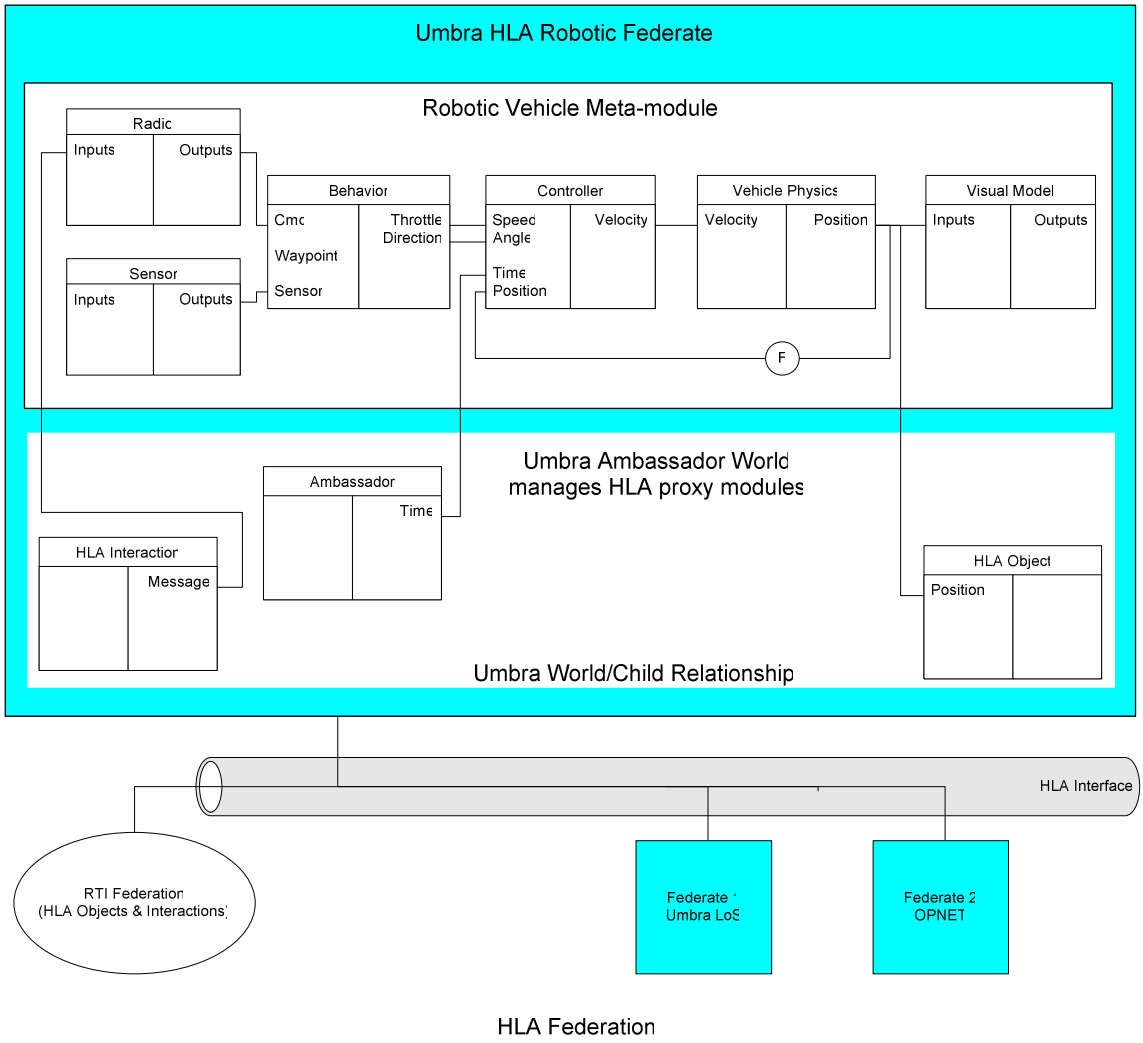


Figure 4: Umbra federate and modules

2.3.2 OPNET HLA Federate

OPNET Modeler has a module that enables OPNET Modeler to participate in a federation of multiple simulators. The OPNET/HLA module provides an interface for various simulators, called federates, to share common object representations, to exchange messages, and to maintain appropriate time synchronization. With OPNET participating as a federate, it can both affect, and be affected by, events in other federates. The OPNET module provides the following HLA functions [13]:

- Time management to coordinate the synchronization of OPNET simulation time and overall HLA time.
- Creation, destruction, or discovery of Run-Time Infrastructure (RTI) objects during the simulation. OPNET can cause the creation and mapping of objects in other federates, however, OPNET cannot create objects in its own federate during simulation run time.

Thus all objects expected to be represented in OPNET must be created before the simulation begins.
- User-defined mapping specifying the relation between RTI and OPNET objects and interactions.
- Generation and reception of RTI interactions.
- Generation and reception of RTI object attribute updates.

The OPNET/HLA module includes an HLA handling node that acts as a bridge between the HLA RTI and the OPNET Modeler nodes that represent the communication protocols on the federation objects. The module will not generate the federate's supporting Simulation Object Model (SOM) file, the federation's Federation Object Model (FOM) file, or the related *.fed* file. The simulation user must define these files along with developing the corresponding network, node, and process models that implement the federate's actions and SOM-specified behavior. In addition, the user must define the packet data structures Modeler uses to represent interactions, identify the destination process module the interaction is directed, and define the computation triggered by the reception of an interaction.

Figure 3 illustrates a pair of node models in OPNET. If an application bound to a node, which is simulated in Umbra, requires data to be communicated to another node, it sends the necessary data to the OPNET simulator via HLA. The OPNET simulator directs the data to the HLA Layer-7 Process Module of the appropriate node. The HLA Layer-7 Process Module receives the data and creates a message. The message is acted upon by the various protocols and transmitted to the destination node. Upon receipt at the destination node, the destination node's HLA Layer-7 Process Module unpacks the message and submits the data to the intended federate via HLA.

The data arrives at the destination federate with the appropriate physical properties resulting from the communication network, such as time delays and bit errors. The OPNET simulator also simulates the impacts on the message transmissions through the wireless channel.

Umbra/OPNET Interface Process Module:

An objective of our research was to develop a capability for obtaining high-fidelity simulation results of network-centric systems. The physical characteristics of the nodes comprising the network-centric system are modeled in the Umbra system simulator. To include high-fidelity representations of the communication network of the system, communication network architecture and protocols are modeled in OPNET. The co-simulation of Umbra and OPNET results in high-fidelity outputs.

Our development of the Umbra/OPNET co-simulation capability used the OPNET HLA module. The module provides the interface to share simulation data between the RTI and OPNET Project. Our development activities included building multiple interface modules to extend the simulation data to the targeted node models and process modules.

HLA Layer-7 Process Module:

The primary OPNET module developed in this research is a state machine that receives and sends data to the HLA module for delivery to the other appropriate federates. This module is

called the HLA Layer-7 process module since it behaves as the application layer, or Layer 7, of the OSI Seven Layer Communication Protocol stack [17]. Any data generated by node applications represented in other federates will be directed to this layer of the appropriate node. This layer behaves as the application layer of any communication node. The module directs the appropriate transport layer protocol to manage the data forwarding. In the case of UDP, UDP ports will be established. In the case of TCP, this layer will instruct TCP to setup the necessary connection.

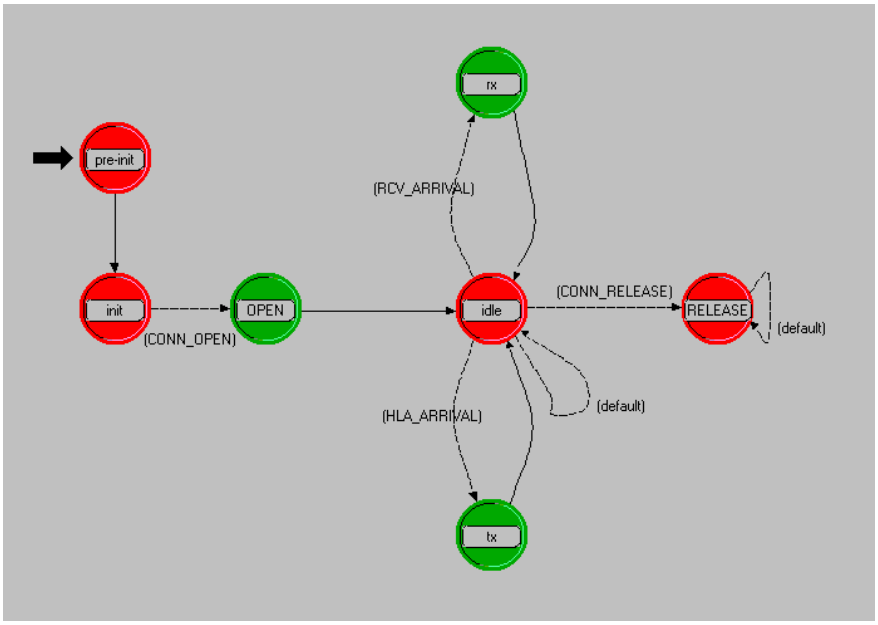


Figure 5: HLA Layer-7 Process Module

Table 1: Description of HLA Layer-7 Process Module States

State	Function
pre-init	This state is to ensure that each node has finished its pre-initialization before going to the init state. A number of process modules, such as the Dynamic Source Routing (DSR) process module, must assign their node address and check that this address is valid within the network.
init	This state initializes every variable, statistic, table, and user parameter that is used by the HLA Layer-7 process model. If UDP is used at the transport layer, then this state will setup the UDP ports at each node. If TCP is used at the transport layer, this state will register the TCP API.
idle	This is the default state where the process waits for an event.

OPEN	This state initiates a TCP connection and initiates the *receive* command. The receive command is issued to the TCP Layer and indicates the TCP client is ready to receive the number of packets as specified. In addition, a self-interrupt is scheduled for sending the *close* command.
tx	This state is entered when a packet is received by the HLA module to be handled by the communication protocol stack. The data is received as an array of strings and must be extracted using the appropriate External Data Representation (XDR) [19] function calls. The data is formatted into a message that is sent to the TCP layer.
rx	This state is entered when a packet is received from the transport layer for the application layer. From this packet an application layer packet is de-capsulated. The data in this packet is packed via XDR into a packet that is submitted to the HLA module for delivery to the RTI for final delivery to the appropriate federate. And finally a receive command is sent to the TCP layer indicating that this layer is ready to receive additional packets.
RELEASE	This state is initiated when the TCP connection should be closed. A close command is issued to the TCP layer to end the connection. In addition, a close command is sent to the application layer on the server's side of the connection.

HLA Object Attribute Process Module:

The HLA Object Attribute Process Module coordinates the object discovery and mapping of OPNET objects and other federate objects. This module also is responsible for maintaining object attribute updates that OPNET modules subscribe to or publish. Any attribute update will be managed by this module.

Table 2: Description of HLA Object Attribute Process Module States

State	**Function**
pre-init	This state is to ensure that each node has finished its pre-initialization before going to the init state. A number of process modules, such as the DSR process module, must assign their node address and check that this address is valid within the network.
init	This state initializes every variable, statistic, table, and user parameter that is used by HLA. It also establishes the mechanism to obtain notification whenever an RTI attribute update is received for the specified object, or that object is mapped or unmapped.

wait	This state is entered when ever the RTI has object information. The state handles object mappings in conjunction with the RTI. Any object attribute that is subscribed to or published by OPNET nodes is managed by this state.

OPNET Line-of-Sight Process Module:

The OPNET Line-of-Sight Process Module establishes and maintains the LoS array within the OPNET process. The Umbra LoS federate calculates the LoS values for each pair of nodes in the OPNET federate and provides LoS values to OPNET via an HLA interaction. The LoS value is dynamically updated during the simulation when a change has occurred from the previous value. When the Umbra LoS federate has an update for a single or multiple LoS values, it packs the values into an HLA interaction and sends it to the RTI, which forwards the data to OPNET. The OPNET Line-of-Sight Process Module receives and unpacks the HLA interaction data. This module maintains a current LoS array by loading the updated data into the appropriate location.

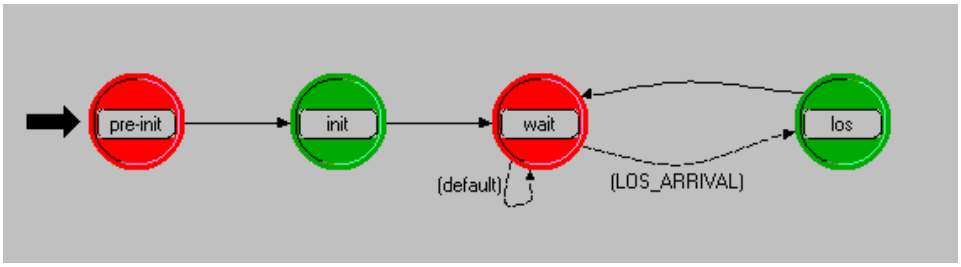


Figure 6: Line-of-Sight Process Module

Table 3: Descrption of Line-of-Sight Process Module States

State	Function
pre-init	This state is to ensure that each node has finished its pre-initialization before going to the init state. A number of process modules, such as the DSR process module, must assign their node address and check that this address is valid within the network.
init	This state initializes every variable, statistic, table, and user parameter that is used by the LoS process model. This state registers the module to receive interaction updates from the HLA module. To increase the efficiency of the LoS table access a hash table is created to index into LoS array. Each possible pair of nodes has an entry in the LoS table.

wait	This is the default state where the process waits for a LoS update.
los	The LoS data is delivered by the RTI as a structure and must be unpacked with XDR and loaded into the LoS array.

2.3.3 Federation Time Management

HLA provides time management services to coordinate events between simulation environments. HLA time management services provide the user with several options. The time management option used in this research is time-constrained and time-regulating. Detailed descriptions of the HLA time management options can be found in [5, 6]. To keep network traffic between federates at a manageable level, the typical HLA federate will pace time in relatively large steps – on the order of seconds. Conversely, to provide sufficiently high fidelity physics and interactive capabilities, typical Umbra simulations are run with relatively small time steps – on the order of milliseconds. The Umbra-HLA integration provides a time management and coordination scheme that allows the HLA federates to make large time steps, greater than one second, while the Umbra environment uses small internal time steps, less than 100 milliseconds. In the limiting case, the design allows Umbra to operate with arbitrarily large time steps designed to keep pace with any federation. The Umbra Ambassador module makes time advancement requests to the RTI in order to advance time in the federate. All the Umbra modules requiring local time data connect to the Umbra Ambassador to get the current time data.

2.3.4 Umbra/OPNET Federation Model

For the Umbra and OPNET federates to interoperate, a Federation Object Model (FOM) was created that defines the required HLA objects and interactions that reside in the federation. For the federation used in this research, we modified the existing Joint Virtual Battlespace (JVB) FOM to incorporate the new HLA objects and interactions. The existing HLA class *platform* was modified to include three new HLA parameters of type string: *address*, *nodeName*, and *commsType*. The HLA parameter *address* is in the form of an internet address, e.g. 192.000.000.010. The parameter *nodeName* is the name of the communication node. The parameter *commsType* is used by the OPNET Modeler federate to decide which HLA platform objects are used as communication nodes within OPNET. If an HLA platform object is created and *commsType* is set to 'opnet', then the OPNET Modeler federate subscribes to that object. If *commsType* is set to something else, the OPNET Modeler federate will simply ignore that HLA platform object. Four new HLA interactions were added to the FOM – *message*, *message_reply, message_ack* and *losTable*.

The HLA interaction, *message,* is sent from the Umbra Platform federate to the OPNET Modeler federate. This is a non-propagated message that contains five HLA interaction parameters: *source address, destination address, message data, message size,* and *message id*. The first three parameters are of type string and the last two are of type integer. The HLA interaction, *message_reply,* is sent from the OPNET Modeler federate to the Umbra Platform federate. This is the propagated message that contains the same parameters as the *message* interaction. The

HLA interaction, *message_ack,* is sent from the OPNET Modeler federate to the Umbra Platform federate. This interaction contains two HLA interaction parameters: *message id* and *message_status.* This HLA interaction enables the logic modules within the Umbra Platform federate to respond to an *ack* or *nack* event for a message sent by a radio. The HLA interaction, *losTable,* is sent from the Umbra LoS federate to the OPNET Modeler federate. This interaction contains a LoS table of all the communication links. The values in the table range from zero, which describes no LoS, to one, which describes clear LoS.

The Umbra Platform federate creates platform objects and publishes data attributes for these HLA objects. This federate also publishes the *message* HLA interaction. The OPNET Modeler federate subscribes to the HLA platform objects, publishes the *message_reply* HLA interaction, and subscribes to the *losTable* HLA interaction. The Umbra LoS federate subscribes to the HLA platform objects and publishes the *losTable* HLA interaction. Table 4 shows the HLA objects and interactions required for modeling the communication service for platforms.

Table 4: Federate Publish and Subscribe Table for HLA Objects and Interactions

Federate	**Interactions**				**Objects**
	message	*message_reply*	*message_ack*	*losTable*	
Umbra Platform	Publish	Subscribe	Subscribe	NA	Publish
OPNET Modeler	Subscribe	Publish	Publish	Subscribe	Subscribe
Umbra LoS	NA	NA	NA	Publish	Subscribe

2.3.5 Co-simulation Process Steps

In order to provide the reader with a general description of the process when co-simulating Umbra and OPNET, we provide a brief description of the steps. This description provides a general overview and further details can be obtained by examining the C programming code in detail [13].

1. *OPNET reads the class mapping file identified in the simulation attribute list and performs appropriate RTI initialization.*
   During this stage OPNET can initiate the federation itself or, if the federation exists, the OPNET federate will join the federation. In our simulation studies, Umbra creates the federation. Once the joining succeeds OPNET indicates its time management mode to the RTI and reads the class mapping file which relates OPNET entities to HLA entities. The HLA entities are listed in the *.fed* file. Only the HLA entities listed in the mapping file will be seen by the OPNET federate and, similarly, only the OPNET entities listed in the mapping file will be seen by other federates.

2. *OPNET discovers HLA objects and maps these discovered objects to an appropriate OPNET object as described in the mapping file.*
   An HLA object created by another federate is discovered and mapped during the simulation. The discovery and mapping can occur at any time during the simulation. Thus other federates may create objects during simulation runtime and OPNET will discover and map these objects. Objects may also be removed from the simulation by other federates. Note that OPNET *cannot* dynamically create objects, thus all necessary OPNET objects must be created when the simulation starts. OPNET can dynamically map its objects to existing or newly created HLA objects.

3. *OPNET contributes and receives time advances from the RTI.*
   The RTI maintains the federation (global logical) time independent of the individual federate logical times. It sorts the events generated by the various federates into a sequence and delivers the events to federates at the appropriate times. The OPNET federate is a time-regulating and time-constrained federate meaning that it participates in the determination of federation time and is also governed by the federation time. Federation time advances only when all time-regulating federates can ensure that no more events will be generated for the current time. When this is satisfied the federation time advances. Each time-constrained federate is informed of the time advance via a callback and advances its own logical time to the federation time. The logical time of a time-constrained and time-regulating federate never advances ahead of the federation time.

4. *OPNET receives HLA attribute updates and updates the corresponding OPNET attribute.*
   The mapping file describes the HLA object attributes to which OPNET subscribes. When any other federate changes these attributes, the RTI informs OPNET with the new attribute value. Corresponding OPNET nodes receive the attribute change and set the attribute value at that time in the simulation.

5. *OPNET publishes HLA attribute updates.*
   The mapping file describes the HLA object attributes which OPNET publishes. When an OPNET object attribute changes, OPNET informs the RTI of the change. In turn, the RTI informs other federates, that subscribe to the attribute, of the change.

6. *OPNET receives or transmits HLA interactions.*
   In the case of receiving an HLA interaction, OPNET converts the interaction to a packet and forwards the packet to the appropriate process module. In the case of transmitting an interaction, an OPNET process model will create a packet and forward it to the OPNET HLA process module, which submits the packet information as an interaction to the RTI.

7. *Concluding the simulation.*
   When the OPNET simulation completes, the OPNET federate will resign from the federation.

Many of the above steps require the exchange of data between the various federates. In the interest of maintaining machine independence of our co-simulation capability, we employ the External Data Representation (XDR) standard [19] for packing and unpacking data that is shared among federates. Data that is transferred between the RTI and OPNET via interactions is packed and unpacked with XDR. XDR is only used when packing and unpacking data that is transferred

via structure data types in interactions because OPNET does not apply byte-order to data structures embedded in the HLA interactions. OPNET does apply byte-order to all other types of fields in interactions or object discoveries. XDR is a set of library routines that allow a C programmer to describe arbitrary data structures in a machine-independent fashion.

3. Simulation Developments for the Umbra/OPNET HLA Federation

To further improve simulation results, our research also developed enhancements for the simulators. The enhancements resulted in either decreased simulation run-times and/or improved simulation result accuracy. A method was developed to use dead reckoning algorithms to enable subscribing mobile platforms to estimate their platform position between published platform position data. This development allows an increase in the Umbra simulator time-step without compromising simulation result fidelity. Simulation fidelity is increased by including the effects of the wireless channel on the transmitted signal. This is accomplished with improved OPNET channel models that incorporate terrain effects.

3.1 Dead Reckoning within the Umbra Federate

Dead Reckoning algorithms enable subsribing mobile platforms to estimate their platform position between published platform position data. The Umbra module (class type *Platform*) performs the dead reckoning for umbra platforms. Typically, Umbra federates (either publising or subscribing) run at 10 to1000 times higher frequency than the federation time advances. That is, module values are computed 10 to 1000 times more frequently than time requests are made. This allows Umbra to accurately integrate behaviors over the relatively large time steps. The following equation computes the dead reckoned position.

$$P = P_o + V*dt + 0.5*A*dt\text{^}2$$

Here, P, V, and A are position, velocity and acceleration vectors, P_o is the last published position vector, and dt is the time since the position was last published. If velocity and acceleration are included with the published position data, then subscribing federates must use these values. If they are not published, then an instantaneous velocity, ($\Delta P/\Delta t$), and acceleration, ($\Delta V/\Delta t$), can be calculated as the simulation progresses. Δt is Umbra's internal time step (higher frequency than federation time).

The use of acceleration in the dead reckoning of ground vehicles introduces a significant noise component. In particular, the action of clamping the position of ground vehicles to the terrain results in subtle velocity changes but very high acceleration changes. In practice, published acceleration is not used in predicted ground vehicle positions and conservatively it is used in predicting air vehicle positions. For UAVs, which tend to have gradual changes in accelerations, the inclusion of acceration values dramatically improve dead reckoning over simply reporting position and velocity.

3.2 Modeling the Effects of a Wireless Channel in Umbra/OPNET Federation

In order to examine the robustness of communication architectures and protocols operating in complex terrains, it is necessary to include the effects of the wireless channel on the transmitted signal. In our simulations, the transmission path between the transmitter and the receiver can vary from simple LoS to one that is severely obstructed by buildings, mountains, and/or foliage. To further complicate channel characteristics the speed of the transmitting, receiving, and surrounding nodes impacts signal level fades and Doppler spreads. Models that represent the effects of the wireless channel can be divided into two groups; *large-scale* propagation models and *small-scale* models.

Large-Scale Models:

Propagation models that predict the mean signal strength for an arbitrary transmitter-receiver separation distance are called *large-scale* propagation models. These models characterize the path loss of a signal as it propagates over large distances.

In our simulations we use a combination of the log-distance path loss model [15 pg. 102] and the log-normal shadowing model [15 pg. 104]. The models combine theoretical calculations and measurements to describe the received signal power in various environments. The large-scale path loss is expressed as a function of distance by using a path loss exponent, *n.*

$$P_L(\text{dB}) = P_{LRef}(d_0) + 10n\log(d/d_0)$$

Where d is the distance between the transmitter and the receiver, d_0 is the close-in reference distance, and n is the path loss exponent that indicates the rate at which the path loss increases with distance. The close-in reference distance and the path loss exponent are selected based on the environment and the degree of LoS of the communication link.

In the case of an unobstructed communication link, a commonly used distance is 1 km [15 pg. 104]. This value is derived from measurements taken to describe cellular communication systems. The reference path loss is calculated using the free space loss formula. For free space propagation, the path loss is described as:

$$P_L = (4\pi d/\lambda)^2$$

The path loss exponent, n, for unobstructed links is computed from measured data using linear regression. From the various values provided in [15] we selected an average value of 3.1, which describes the path loss in an unobstructed urban area.

In the case of an obstructed communication link, the log-normal shadowing model is used. The path loss value is calculated from measured data with a single regression curve fit [15]. The path loss exponent for obstructed communication links can be obtained from multiple references [15, 17], which have collected experimental data and calculated path loss exponent values from the data. We selected an average value of 4, which describes the path loss in an obstructed urban area.

In our model, the path loss is calculated from log-distance model if the LoS value is greater than 0.50. If the LoS value is less than or equal to 0.50, the log-normal shadowing model is used.

In addition to the effects of large-scale propagation effects on the wireless channel the effects of the rapid fluctuations of the received signal strength over very short travel distances (a few wavelengths) or short time durations must be included in high-fidelity simulations of wireless communication networks. The rapid fluctuations are described with *small-scale* propagation models.

Small-Scale Models:

In order to examine the robustness of communication architectures and protocols operating in complex terrains it is necessary to include the effects of multipath delay spread and Doppler fading on the wireless channel. These two types of fading, called *small-scale fading*, describe the rapid fluctuations of the amplitude of a radio signal over a short period of time or node travel distance.

Multipath occurs at the receiver when two or more versions of the same transmitted signal arrive at the same time. Each of the multipath signals arrives at the receiver at a slightly different time resulting in either constructive or destructive interference. Deep nulls can occur in the energy at the receiver if the primary multipath ray is 180 degrees out of phase and similar amplitude when compared to the primary ray. Multipath occurs in terrains that have a large number of reflecting surfaces such as buildings in urban terrains. In most cases, the primary multipath is a ground reflection that arrives at the receiver slightly later than the direct path signal.

An additional factor that results in fading at the receiver is the relative motion between the transmitting node, the receiving node, or other reflecting objects in the surrounding environment. The relative motion results in an apparent shift in frequency called Doppler shift [15]. The Doppler shift is directly proportional to the velocity and direction of motion of the mobile nodes with respect to the direction of arrival of the received multipath ray. In general, the fading effects resulting from the motion of surrounding objects can be ignored if their speed is less than the speed of the transmitting and receiving nodes [15 pg. 141].

Small-scale fading can be further broken down into groups describing *flat* versus *frequency selective* fading and *fast* versus *slow* fading. Detailed descriptions of these groups can be found in [15 pg. 167]. If the mobile wireless channel has a constant gain and linear phase response over a bandwidth that is greater than the bandwidth of the transmitted signal, then the received signal will undergo flat fading. In this case, the spectral characteristics of the transmitted signal are preserved, however, the strength of the received signal changes with time due to the effects of multipath on the channel gain characteristic.

Our targeted applications also suffer from a *slow fading* wireless channel. In a slow fading channel, the channel impulse response changes at a rate much slower than the transmitted baseband signal *s(t)*. When considered in the frequency domain, this implies the Doppler spread is less than the bandwidth of the baseband signal. Here the velocity of the nodes determines the Doppler spread and thus determines if slow or fast fading occurs.

To represent the effects of the fading channel in our simulations we employ Ricean fading distribution to the statistical time varying nature of the received signal envelop. Ricean fading is used to describe the fading effects of our simulated channel when a dominate LoS propagation path is present. When no dominant LoS path is present (LoS value less than 0.50) , the Ricean distribution degenerates to a Raleigh distribution [15 pg. 175]. The Ricean distribution is often described in terms of a parameter *K,* defined as the ratio between the deterministic signal power A and the multipath variance σ,

$$K(\text{dB}) = 10\log\{A^2/(2\,\sigma^2)\ \text{dB}.$$

The parameter K is the Ricean factor. For the case where there is no dominate LoS path, $A \rightarrow 0$ resulting in $K \rightarrow 0$. Since in our simulation scenarios, we expect cases where we have both dominate LoS path signals with multipath and no dominate LoS path with multipath we must dynamically control the value K.

A model that can accurately represent time correlation is the Clarke and Gans fading model [14, 15]. This model represents the Doppler power spectral density as a sequence of in-phase and quadrature components that can be combined to yield the fading envelop with spectral and temporal characteristics very close to measured data. The resulting spectrum shape of the Doppler spread determines the time domain flat fading waveform and dictates the temporal correlation and fade's slope behaviors. In creating realistic fading waveforms, it is critical that accurate time correlation be maintained [14].

A method to model Ricean fading is described in [14]. This method uses a pre-computed dataset containing the components of a time-sequenced fading envelope. This dataset is used in the simulation with a few simple calculations to represent the fading envelop. The following steps are necessary to create the dataset that can be used efficiently in the simulation.

1. Determine the maximum Doppler frequency, f_m, number of frequency domain points, and the spacing between adjacent frequency points. The time duration of the fading waveform is determined by the frequency point spacing $1/(\Delta f)$. Although this will result in a dataset that represents a limited time-sequence, long simulations can be performed by repeating this dataset.

2. Generate complex Gaussian random numbers for each of the frequency components. To yield a real waveform, the complex frequency component and its negative frequency must be complex conjugates.

3. Using the equation described in [15 pg. 180] calculate the fading spectrum. Multiply the frequency components by the fading spectrum.

4. Perform an IFFT on the resulting frequency domain signal and sum the squares of both the in-phase and quadrature component. This results in the time series data.

The result of these steps is a normalized Rayleigh fading signal represented by an in-phase component x_1 and a quadrature component x_2. A normalized Ricean fading envelop can be calculated from the values of the dataset as,

$$r = \text{sqrt}((\sigma x_1 + A)^2 + \sigma(x_2)^2).$$

Fading envelopes resulting from different Doppler frequencies can be calculated from the normalized dataset. A time-stretch method is used to calculate the necessary simulated Doppler spread from the dataset that is created from a pre-selected maximum Doppler frequency f_m by appropriately stretching or squeezing the original time series. This technique is described in [14].

To increase the fidelity of our simulation results we developed a means to dynamically provide the path loss calculation with a value describing the level of LoS. The Ricean fading model used receives a LoS value from Umbra and includes this value in the LoS calculation as described in the next section.

Dynamic Calculation of Line-of-Sight:

Umbra creates and maintains a dynamic LoS table during the simulation run time. This table includes a LoS value for each possible pairing of nodes in the simulation. For each pipeline calculation in OPNET, this LoS table provides a value describing the level of LoS that is used in the fading calculations.

Umbra provides a LoS value between 0 and 1, with 0 describing no LoS and 1 describing clear LoS with no obstruction in the majority of the first Fresnel zone [15 pg. 91]. The Fresnel zones represent successive regions where secondary waves have a path length between communicating nodes that are $n\lambda/2$ greater than the direct path length. We consider a clear LoS if 55% of the first Fresnel zone is unobstructed [15 pg. 94]. In addition, Umbra provides a LoS value of -1 if no communication is possible between the transmitting and receiving node. This case arises when, for example, the transmitting node is in a subterranean feature and the receiving node is on the surface of the earth and no communication is possible. In cases where the LoS value is -1, the OPNET link-closure pipeline stage determines that no communication or interference is possible between the two nodes and no further pipeline stages are calculated.

Example Simulation Results of Path Loss Models:

Figure 7 illustrates the received power when the channel model is used in a simulation. The resulting plot illustrates the effects of fading for a node moving at a velocity on 1 m/s. The transmitter-receiver pair is tuned to 905 MHz and has a channel bandwidth of 20MHz. In this example, there is no dominant LoS path thus the value of $K = 0$. This example illustrates the received power envelop in dB.

This plot is the result of an application transmitting short packets at a period of 10ms. In a discrete event simulator, the received power is computed only once at the arrival of the packet at the receiver. Any other packets that arrive at the receiver during the reception of the original packet contribute noise and the simulator will evaluate the new signal-to-noise ratio at that time. An additional issue is that since the simulator samples the received power only once at the packet arrival only slow fading can be modeled in the simulator. Slow fading occurs when the variations in the signal amplitude occur at a much slower rate than the transmitted symbol duration. It should be clear that the velocity of the node or the velocity of objects in the channel and the baseband signaling determines whether a signal undergoes slow fading or fast fading.

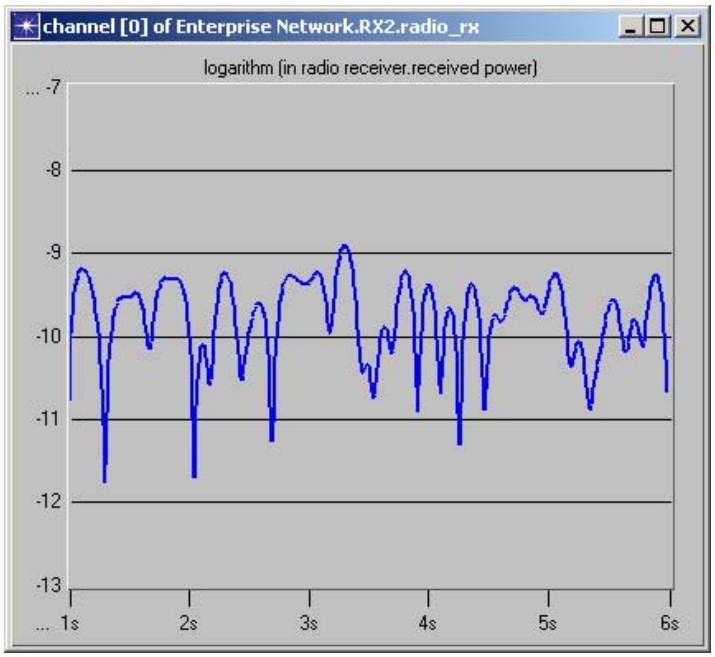


Figure 7: Received power at the radio receiver in a fading channel

4. Umbra/OPNET Federation Simulation Demonstration

To demonstrate the capability and usefulness of the Umbra/OPNET federation in analyzing wireless ad hoc communication network architectures and protocols, we provide a simulation experiment. This simulation experiment demonstrates the operation of a system that depends on wireless communications in a Military Operations in Urban Terrain (MOUT) environment. The objective of the experiment is to determine the *speed of service* of the command center receiving information describing a potential target that has entered an area of interest. The experiment includes the following communication effects:

- Node-to-node communication between patrol units and command center,
- Unattended ground sensor communicating detection events to the command center,
- Command center directing patrol units to perform Combat Identification (CID) [1],

This operation is affected by the accuracy of the command center's Common Operating Picture (COP) since it dispatches an asset based on its position in the COP, which may not be its actual current location. The quality of the COP depends on each node's ability to communicate its position data to the command center via a multi-hop wireless network.

In summary, this simulation experiment demonstrates the quality of the COP for selected network configurations. In this simulation, the quality of the COP is compared when the underlying wireless network uses the DSR or Temporally Ordered Routing Algorithm (TORA) ad hoc routing algorithms. DSR provides a better COP and consumes less network bandwidth. System operation with link-layer message retransmissions was also simulated to determine degree of improvement in the COP and the amount of additional bandwidth consumption. Our experiment demonstrated improvement when link-layer retransmissions were used with the TORA routing protocol. Link-layer retransmissions mitigated packet collisions, which was a problem in the TORA network.

An important finding of this demonstration simulation experiment is that the results indicate that DSR is a better performing protocol in our MOUT demonstration scenario than TORA. This finding is contrary to the protocols' expected performance based on observations of their characteristics. DSR is classified as an on-demand routing protocol, which describes its operation as discovering routes at the time the route is needed. In contrast, TORA is classified as a proactive routing protocol, which describes its operation as discovering and maintaining routes proactively so that when a route is needed, the source uses cached routes. Typically, proactive routing protocols have a lower end-to-end latency when compared to reactive routing protocols. In this experiment, the reactive protocol has equal end-to-end latency performance while consuming significantly less bandwidth than the proactive routing protocol. This better performance by DSR is because the highly dynamic nature of the experiment scenario. TORA proactively discovers routes but the routes become unusable because of the mobility of the nodes. For TORA, the routing protocol must be configured to transmit frequent HELLO messages to discover link breakages. Even with the frequent HELLO messages, the source is not able to deliver the message via the cached route in many cases, resulting in a large number of resends.

4.1 Description of Simulation Experiment

An Umbra OPNET federation simulation was created to analyze the MOUT scenario described above. This scenario contains a stationary command station node with eight mobile vehicles patrolling a region of interest within a city containing buildings. Each mobile vehicle patrols a designated path within the city while communicating its *Situation Report* back to the command station via a wireless ad hoc network as shown in Figure 8. The *Situation Report* may require several hops to reach the command station depending upon the location of the vehicle within the city due to the LoS obstructions produced from the buildings and terrain. Each vehicle periodically dispatches its *Situation Report*. The start times of the eight *Situation Reports* are interlaced to prevent as much communication contention interference as possible. The command station updates its COP each time a *Situation Report* is received. The command station also calculates an estimated position of the vehicle based on the vehicle's position and velocity information provided in the *Situation Report*.

In addition to the mobile vehicles, there are six unattended ground sensors positioned alongside the roads. When an object of sufficient size moves through the sensor's field of view, the sensor tracks the object and sends an *Identified Report* to the command station. The sensor transmits the *Identified Report* when the object reaches the sensor's closest point of approach. The sensor is

capable of tracking multiple objects within its field of view. The eight mobile patrol vehicles, the six sensors, and the command station comprise the ad hoc wireless communication network and participate in forwarding messages as necessary.

When the C2 receives an *Identified Report* from a sensor, the command station examines its COP to determine if one of the eight mobile units triggered the *Identified Report.* If the command station determines that it is one of the eight mobile units, it will not send an *Investigate Report.* If it determines that it is not one of the eight mobile units, it will send an *Investigate Report* to the closest mobile unit. The mobile unit will initiate a path algorithm that will provide it a course to the triggered sensor area to provide a combat ID on the object that tripped the sensor. This scenario is described in Figure 9.

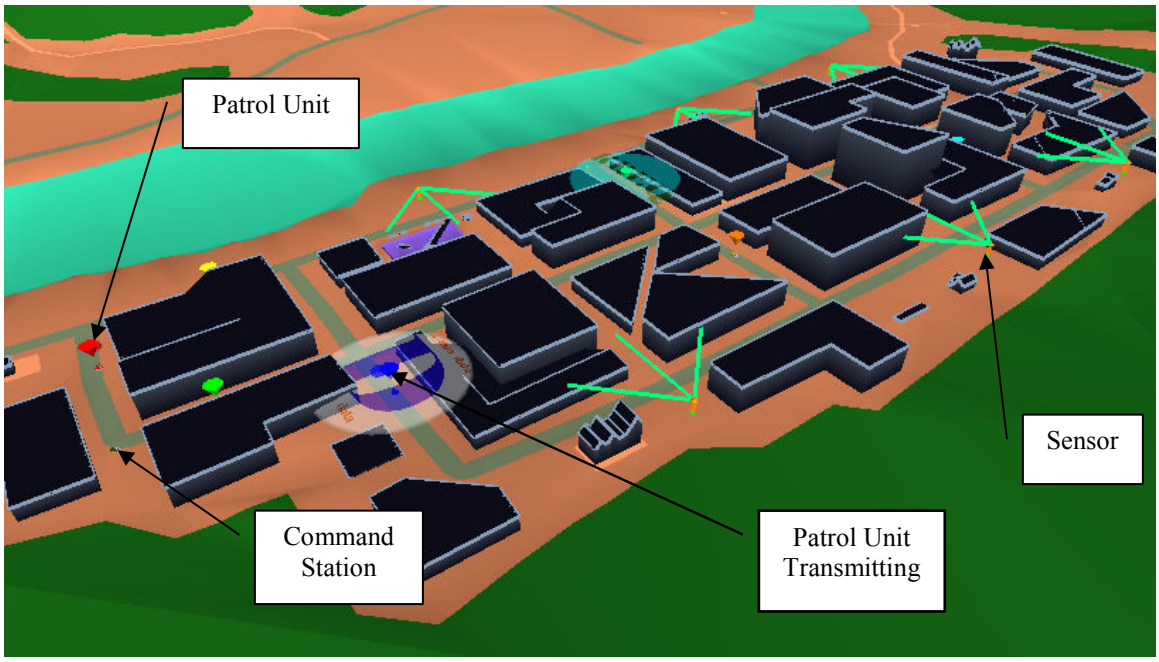


Figure 8: City topology used for the Umbra/OPNET system level simulation

For this simulation experiment, in Umbra we model the city topology, node start positions, initial node patrol pattern, sensor location, and enemy entry point. OPNET models the communication network protocols used on each node and the wireless channel effects upon the communication. This scenario is repeated while parameters under study are varied. In this experiment, we vary a number of communication network parameters. The primary Measure-of-Effectiveness (MOE) in this experiment is the quality of the C2 node's COP. This metric is a measurement of the actual mobile unit location in comparison to where the C2 node's COP estimates the node is located. A secondary MOE for this experiment is the necessary network bandwidth necessary to provide the updates to the C2 node.

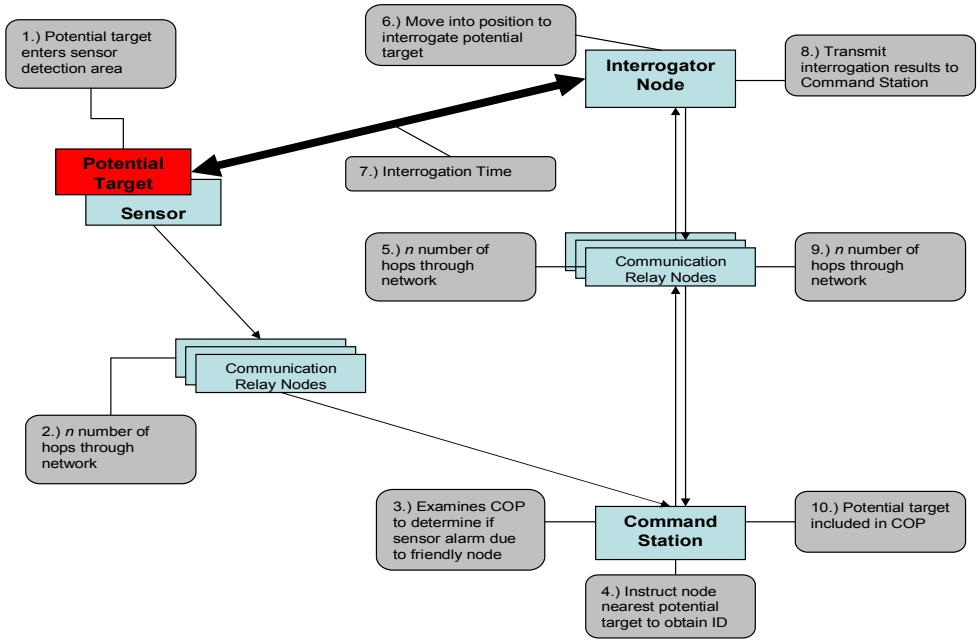


Figure 9: Block diagram of events in the system-level simulation experiment

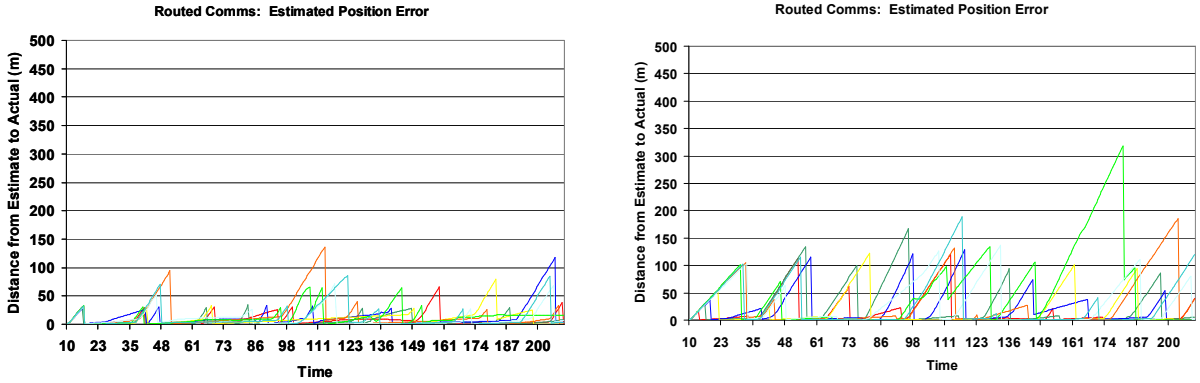
Umbra initiated each simulation run with identical starting conditions. Each simulation run depended on a communication network, modeled in OPNET, with either different network protocols or different configuration parameters. The variation in communication network parameters resulted in different system level performance and different results in the identified MOEs. Table 5 lists the protocols and various configurations of these protocols that were analyzed in this experiment.

Table 5: Protocols used in each node

Layer	Description
Application	The application layer traffic is provided by the Umbra simulator. The *Situation Report* update messages are 50 bytes in size. The sensor messages are 400 bytes in size.
Transport	A protocol that provides persistent resends until an ACK is received is used. This protocol will buffer a packet and will retransmit the message every two seconds until an ACK is received from the destination for a maximum of 10 transmissions.
Network	Two types of ad hoc routing protocols were studied as follows: 1. IP is used at the network layer and DSR [2, 11] is used to

	represent the class of on-demand ad hoc routing protocols employed at the network layer. Our simulation experiments configured DSR to use source routing, 300 second route cache timer, enabled the route replies with route cache, and enabled packet salvaging. 2. TORA [3, 4] is a link reversal protocol that reverses direction of one or more links when a node has no downstream link.
Data link & Physical	The wireless radios access the medium CSMA. The center frequency is 2400 MHz and the data rate is 1Mbps. The maximum number of retransmissions at the link level is 1) one or 2) four.

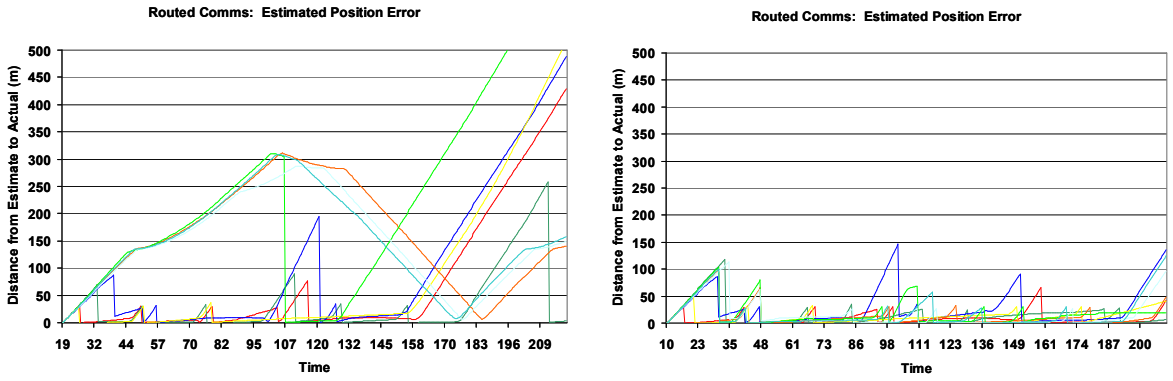
The results of our system level simulations are shown in Figures 10, 11, and 12. Figure 10 describes the quality of the C2 node's COP. Each of the curves in the plot describes the distance between each node's actual location and the location of that node in the COP. The plots compare the error of the COP under two configurations: 1) operating with a dead reckoning position update algorithm and 2) operating with periodic position updates from the mobile units. The dead-reckoning configuration maintains an error value of less than 130 meters. In comparison, the periodic update has a peak value greater than 300 meters. Figure 12b describes the necessary wireless bandwidth necessary for the two scenarios. In this case, the configuration with the lower error value also required less bandwidth.



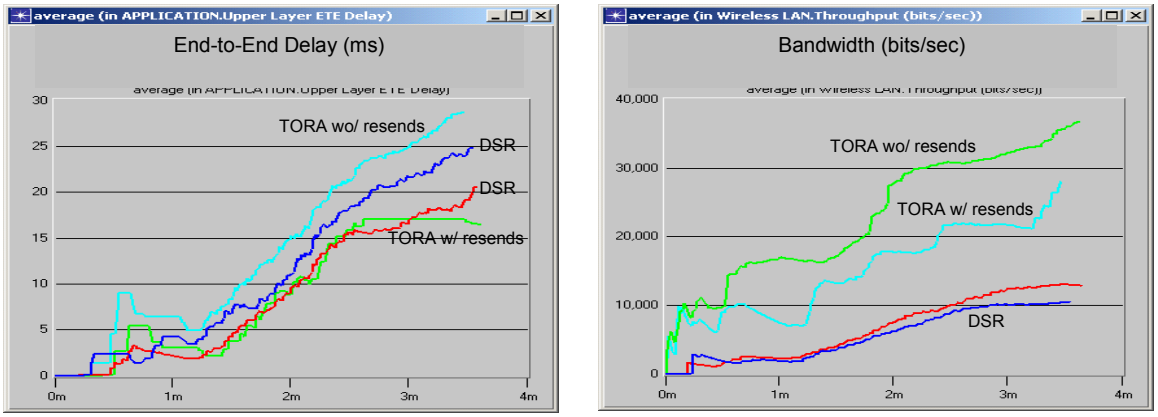
Figures 10a & 10b: Quality of the C2 node's common operating picture when DSR routing is used

Figure 11 shows the error value when the supporting network's ad hoc routing protocol is changed to the Temporally Ordered Routing Algorithm (TORA). The error value at the COP is much larger when TORA is used. The TORA protocol is not able to successfully maintain correct routes and transmit messages in this very dynamic experiment scenario. Figure 12b shows that TORA consumes almost four times the bandwidth as the DSR network. The experiment also demonstrates that this additional bandwidth usage results in more message collision and thus dropped packets. A method to recover from collisions is to include link-layer retransmissions. Figure 11b shows the results of the scenario using TORA at the network layer

but now including up to four retransmissions at the link-layer. The performance of TORA is significantly improved when link-layer retransmissions are used. The bandwidth required when link-layer retransmissions are used is also reduced since the network now can recover from message collisions by resending at the link-layer versus initiating a new route discovery at the network layer. However, the quality of the COP is approximately equal to the quality of the COP when DSR is used but TORA consumes significantly more network bandwidth.



Figures 11a & 11b: Quality of the C2 node's common operating picture when TORA routing is used



Figures 12a & 12b: Upper layer end-to-end delay and wireless network bandwidth consumed for system level simulation experiments

4.2 Conclusion of System Level Impacts when Operating Wireless Ad Hoc Networks in MOUT Environments Experiment

This section described several simulation experiments that illustrate the impacts on applications by variations in the network architecture and/or configuration. Figures 10 and 11 describe the error value in the COP under a MOUT scenario. Our experiment demonstrates how the error value in the COP varies for our selected network variations. This demonstration scenario showed that the application performance can be improved while network bandwidth is reduced by selecting an appropriate ad hoc routing protocol. A proactive and reactive protocol was compared under various configurations in identical scenarios. The results demonstrated that

DSR, the reactive protocol, can match the end-to-end latency performance while consuming less network bandwidth.

An additional finding is the importance of link-layer retransmissions with the TORA protocol. Our experiment demonstrated substantial improvement when link-layer retransmissions were used with the TORA [3, 4]. This improvement occurred because in the TORA scenario, more message collisions occurred on the wireless medium and the link-layer retransmissions reduce the overall number of message transmissions which resulted in the number of message collisions. However, when link-layer retransmissions are implemented for the scenario using the DSR routing protocol, no significant improvement in performance is obtained. This is because message collisions do not impact the DSR scenario to the extent that it impacted the TORA scenario.

An important finding of this demonstration simulation experiment is that the results indicate that DSR is a better performing protocol in our MOUT demonstration scenario than TORA. This finding is contrary to the protocols' expected performance based on observations of their characteristics. This demonstrates the value of simulating with high fidelity models of the application, the communication network, and communication channel.

5. Future Work

A number of areas that require future research to enhance the capability developed in this research include methods to allow simulation of much larger networked systems. One method to increase simulation run-time performance and overcoming the memory limitations of the simulation of large-scale networks is to distribute the computation load to multiple processors. This can be accomplished by distributing federates onto separate computers. HLA should be considered as a means of facilitating the distribution of the computation load.

Methods to include background traffic in the OPNET federate should be investigated. Advances in this area will lead to more efficient simulations of large-scale networks. Simulations could be created that load the network with implicit traffic and the nodes or regions of the network under study can create explicit traffic resulting from system operation.

An important feature of HLA simulations is the capability to create and destroy objects during simulations. This feature is not supported by the OPNET federate since node objects cannot be created dynamically in OPNET. Our current approach overcomes this limitation by creating nodes in a disabled state at the beginning of the simulation and enabling them at the time when they would be created. This approach has limitations in that it is necessary to know the number and type of nodes prior to the simulation start time which may not always be the case. Any change in this limitation requires modification to the OPNET kernel and thus must be done by OPNET Incorporated.

6. Conclusion

Network-centric systems that depend on mobile wireless ad hoc networks for their information exchange require detailed analysis to support their development. In many cases, this critical analysis is best provided with high-fidelity system simulations that include the effects of network architectures and protocols. In this research, we developed a high-fidelity system simulation capability using an HLA federation. The HLA federation, consisting of the Umbra system simulator and OPNET Modeler network simulator, provides a means for the network simulator to both affect, and be affected by, events in the system simulator. Our research resulted in interfaces for both Umbra and OPNET that facilitates publishing and subscribing of system operation information. This capability includes time management with the HLA Run Time Interface (RTI). In addition, a dead reckoning capability was developed in Umbra to reduce the required information exchanges between the two simulators, thus reducing simulation run-time. For increased fidelity of the impacts of the wireless communication channel, we developed a LoS module that dynamically controls the fading algorithm used for each pair-wise node communication.

To demonstrate the value of this modeling and simulation capability a simulation scenario is included. The demonstration scenario illustrates how the quality of the application-layer information is impacted by the wireless network architecture. The measures-of-effectiveness (MOE) for the demonstration scenario was the quality of the C2 node's common operating picture and the parameters that were varied were the wireless network ad hoc routing protocol and medium access control number of retransmissions. This demonstration experiment clearly demonstrated the value of the product of this research.

7. Bibliography

[1] G. Allgood, W. Manges, L. Hatchell, J. Saffold. *Application of Queuing /Renewal Theory in the Development of a Speed of Service Model to Support Fixed Wing to Ground Combat Identification*. 2003 SMART Conference.

[2] J. Broch, D. Johnson and D. Maltz. *Dynamic Source Routing (DSR).* Internet Draft, draft-ietf-manet-dsr-03.txt, October 22, 1999.

[3] S. Corson and V. Park Internet Draft (July 20, 2001) – Temporally Ordered Routing Algorithm (TORA) Version 1.

[4] S. Corson, S. Papademetriou, P. Papadopoulos, V. Park and A. Qayyum. Internet Draft (August 7, 1999) – An Internet MANET Encapsulation Protocol (IMEP) Specification.

[5] Department of Defense; Defense Modeling and Simulation Office. *High Level Architecture Run Time Infrastructure Programmers Guide*. http://www.dmso.mil.

[6] Department of Defense; Defense Modeling and Simulation Office. *High Level Architecture (HLA) Object Model Development Tool (OMDT) User's Guide.* http://www.dmso.mil.

[7] M. Feuerstein, K. Blackard, T. Rappaport, S. Seidel, H. Xia. *Path Loss, Delay Spread, and Outage Models as Functions of Antenna Height for Microcellular System Design.* IEEE Transactions on Vehicular Technology, vol. 43, no. 3, August 1994.

[8] E. Gottlieb, R. Harrigan, M. McDonald, F. Oppel, P. Xavier. *The Umbra Simulation Framework*, June 2001, Sandia Internal Report, SAND2001-1533.

[9] E. Gottlieb, M. McDonald, F. Oppel. *The Umbra High Level Architecture Interface*, January 2002, Sandia Internal Report, SAND2002-xxxx.

[10] E. Gottlieb, M. McDonald, F. Oppel, B. Rigdon, P. Xavier. *Umbra's Joint Virtual Battlespace (JVB) Models*, October 2002, Sandia Internal Report, SAND2002-xxxx.

[11] D. Johnson, D. Maltz and Y. Hu. Internet Draft (February 24, 2003) – The Dynamic Source Routing Protocol for Mobile Ad-Hoc Networks.

[12] OPNET Technologies. *Hybrid Simulation – The Key to Fast and Accurate Results - Description and Case Study*, August 14, 2003.

[13] OPNET Technologies. *OPNET Users Manual*, www.opnet.com.

[14] R. Punnoose, P. Nikitin, and D. Stancil. *Efficient Simulation of Ricean Fading within a Packet Simulator*. Department of Electrical and Computer Engineering. Carnegie Mellon University.

[15] T. Rappaport. *Wireless Communications: Principles & Practices.* Prentice Hall. 1996.

[16] RFC 2501 – Mobile Ad Hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations.

[17] W. R. Stevens. *TCP/IP Illustrated, Volume 1: The Protocols*, Massachusetts, 1994.

[18] B. Van Leeuwen and M. Torgerson. *Performance Impacts of Lower-Layer Cryptographic Methods in Mobile Wireless Ad Hoc Networks*. Sandia National Laboratories, SAND Report 2002-3340, October 2002.

[19] XDR Technical Notes. Chapter 3.