Bandwidth Analysis of a Simulated Computer Network Running OTB

Juan J. Vargas, Ronald F. DeMara, Avelino J. Gonzalez, Michael Georgiopoulos

Department of Electrical and Computer Engineering
University of Central Florida
Orlando, FL 32816-2450
jvargas@ucf.edu, demara@mail.ucf.edu, michaelg@ucf.edu, gonzalez@ucf.edu

Abstract

A network simulation is conducted to determine the bandwidth requirements of military mission rehearsal activities while enroute to deployment. A predefined vignette running under OneSAF Testbed Baseline (OTB) provides the base data for the simulator. This data consists of Protocol Data Units (PDUs) captured by the OTB logger. The PDUs are the messages exchanged by all the entities in the simulation and include the timestamp and byte length, used in the simulation to recreate the network traffic.

The simulation includes 24 computers onboard 8 military airplanes, 3 computers per plane, representing 3 combat vehicles connected via a 100Mbps Ethernet LAN. A router interconnects the resources aboard each plane to those of other planes and also to a remote ground station via 64, 128, 256 and 1024 Kbps wireless communication channels. The results of the simulation are collected, plotted, and conclusions regarding bandwidth and latency implications of the Embedded Training exercises are drawn.

The embedded training computer network simulation is developed using the OMNeT++ public-domain modeling tool. OMNeT++ is a general discrete event simulation tool that contains features aimed to facilitate computer network modeling. It contains a graphical development environment used to define the topology of the network by instantiating and interconnecting different icons that represent the objects of the simulation, like computers, communication channels, routers, airplanes, etc. The behavior of each object is specified in C++. The final product is an executable standalone program that models the network behavior. It can run as an animation or as an express simulation. A wide range of statistics is obtained for analysis.

Introduction

In 2002, the U.S. Army Simulation, Training and Instrumentation Command (STRICOM) and the Computer Engineering Department of the School of Electrical Engineering and Computer Science at the University of Central Florida (UCF) began a joint project to assess the “Bandwidth and Latency Implications of Integrated Training and Tactical Communication Networks.” The main goal of the project is to determine the bandwidth requirements for running Objective Force Embedded Training (OFET) methods.

OFET methods benefits include the ability to perform tasks such as mission planning and rehearsal while enroute to deployment. Benefits include the ability to perform "in-situs" exercises on actual equipment, more direct provision of support for the variety of equipment in the field, and a greater opportunity to develop new training exercises using much shorter lead times than were previously possible with stand-alone training systems.

A fully operational OFET platform also presents several technology challenges. In particular, management of Command, Control, Communications, Computers, Intelligence, Surveillance, and Reconnaissance (C4ISR) resources is required for successful integration of simulation within the actual environment. The purpose of this study is to assess communication requirements to support Enroute Mission Planning and Rehearsal (EMPR) in a Future Combat System (FCS) environment.

The first step towards the development of the network simulator was the design of the vignette illustrating a mission rehearsal operation enroute to deployment. This vignette included a database terrain, friendly and enemy forces, strategies, and other details representative of FCS operations. The vignette was run using the software simulator known as OneSAF Testbed Baseline (OTB).

Entities in OTB communicate to each other by passing messages. Messages in OTB are called Protocol Data Units (PDUs). PDU formats are part of the Distributed Interactive Simulation (DIS) protocol defined in the IEEE Standards 1278.1 (1995), 1278.2 (1995), 1278.3 (1996) and 1278.1a (1998). After running the OTB simulator, the PDUs generated in that session are collected and stored in an output file for further processing. Concerning the bandwidth project, the three most relevant features of the PDUs are the sender ID, the byte length and the timestamp of PDU creation because these parameters are the basic inputs for modeling communication traffic. The destination address is not considered given that OTB broadcasts all the messages to all the participating entities.

The developing team wanted a discrete simulation tool with the following characteristics: great flexibility from a programming point of view, Windows based for availability and readiness, Graphical User Interface (GUI) capable of showing simulation with animation, easy to learn for C++ users, and low cost. After a short survey, the team decided to use OMNeT++ a software tool developed by András Varga at the Budapest University of Technology and Economics (OMNeT++, 2003). OMNeT++ is a general discrete event simulation tool that contains features to facilitate computer network simulations. OMNeT++ runs under different platforms including UNIX systems, Linux and Windows. It contains a graphical development environment used to instantiate the entities of the simulation like computers, communication channels, routers, airplanes, etc. and connect them according to the desired topology.

Using OMNeT++

OMNeT++ is a discrete event simulation environment based on C++. It provides means for describing the topology of the network, either graphically or by using of the NED language (Varga, 2003). In particular, OMNeT++ operates on two types of files: NED and CPP.

The first type corresponds to NED files (xxx.ned) that describe the topology of the network composed of the different nodes involved in the simulation, their input and output gates and the connections between them. A NED file uses a special language called the NED language to describe modules. There are two types of modules: simple and compound. Simple modules contain no other modules inside them and are used to describe the most basic elements of the simulator. In this project, message generators, message sinks, communication channels (wireless and Ethernet buses) and routers correspond to simple modules. Compound modules contain other modules inside. For example, a computer onboard an airplane is a compound module because it contains a message generator and a sink. A plane is also a compound module that contains a computer, a router and an Ethernet bus. The largest compound module corresponds to the total network that contains airplanes, and a wireless bus to link the planes.

The second type of files corresponds to CPP files (xxx.cpp) that use C++ to describe the functionality of the simple modules. Each simple module requires a C++ source code that indicates how to process each packet that arrives to an input gate, as well as how to send a packet to an output gate. Using C++ code we can program the packet contents, its destination, its length, and the distribution used (exponential, normal, uniform, etc.) to schedule packets. The CPP files corresponding to the buses (Ethernet or wireless) handle the packet transmission and collision detection. Once the source files are ready, the simulator is compiled. In this project, Microsoft Visual C++ was used to accomplish this task. A special compiler supplied with OMNeT++ pre-processes the NED files creating cpp files. Next, all the cpp files are compiled together with the Tcl/Tk graphic library producing the executable file. Tcl/Tk is a graphic library in the public domain (Tcl Developer Xchange, 2003) and described in several books like (Zimmer, 1998). For C++ programmers, a sufficient understanding of the way OMNeT++ works along with the capability of developing basic models can be quickly achieved because the main concepts come from general knowledge about C++, as compared to other simulators in which there are several long manuals to read and the concepts are specific to that particular simulator.

Model Design

The model to be simulated is composed of eight airplanes carrying three computers and a router each, plus a satellite and a ground station. A random message generator with a specific distribution can be used to control the rate of packet generation. Alternatively, as in our study, each component actually reads the messages to be broadcasted from an input text file that contains the type, length and timestamp of each message as recorded by the OTB's logger when the vignette was run.

Module descriptions

The GNED editor is used to edit the NED files using a graphic interface. Figures 1, 2 and 3 show the OMNeT++ representation of the compound modules node, airplane and the whole network, respectively.

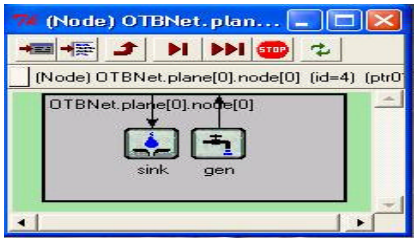

Figure 1. OMNeT++ view of a node containing a sink and a generator

Each node onboard an airplane contains a message generator and a message sink. The legend "`OTBNet.plane[0].node[0]`" in Figure 1 indicates that this compound module belongs to node # 0 located inside plane # 0 which is part of the general network called OTBNet. The planes are identified by using consecutive integers 0, 1, 2, …, 7. Within each plane, the computers are identified as nodes 0, 1 and 2. The brackets indicate that the set of planes and the computers onboard each plane are represented as arrays of simple modules. The arrows represent connections to input and output gates.

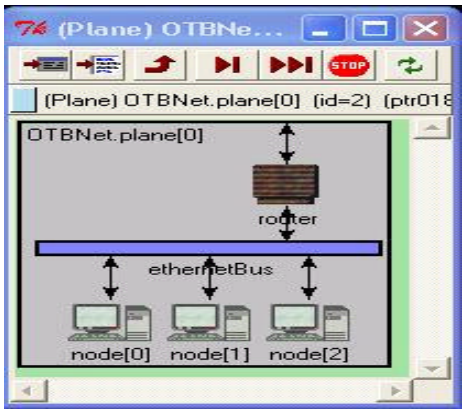

Figure 2. Airplane modules: embedded training stations, bus and router

Figure 2 shows the three nodes, the router and the Ethernet bus onboard each airplane. The three nodes and the router are connected via an Ethernet bus. Figure 3 shows the complete network composed of 8 airplanes, a ground station, a satellite and 3 wireless channels. The first wireless link connects the routers in all planes to each other. A second wireless link communicates the routers in the planes to the satellite, and the third one communicates the satellite to the ground station. In this way, each router is connected to three different links, and the satellite is connected to two.

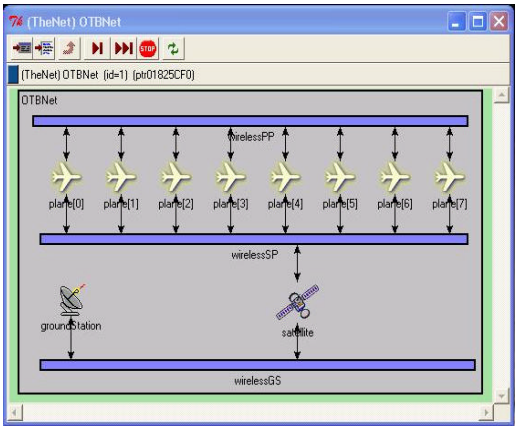

Figure 3. General view of the network showing 8 planes, satellite, ground station and the 3 wireless channels

Figure 4 contains the NED code of a generator, a sink and the satellite. For simple modules the NED code indicates the module name, the input parameters and the input and output gates. The C++ code of simple modules is written in a separate file. Each simple module becomes a C++ class.

```
simple Generator
parameters:
startTime: numeric,
fromAddr: numeric,
totalNodes: numeric;
gates:
out: out;
endsimple

simple Sink
gates:
in: in;
endsimple

simple Satellite
parameters:
startTime: numeric,
satelliteID : numeric,
satServiceTime : numeric,
totalNodes   : numeric,
WGSposition : numeric,
WSPposition : numeric;
gates:
in:  inBus1;
out: outBus1;
in:  inBus2;
out: outBus2;
endsimple
```

Figure 4. NED code of some simple modules

On the other hand, the NED code of a compound module includes additional features like the values of parameters of internal modules and connections between module gates. Compound modules do not need user-written C++

source code because their behavior is completely defined by their simple modules.

Input parameters of the model are used to control the conditions under which each simulation runs. These parameters can be directly given in the NED files, or they can be read from a configuration file at run time. Each simulation starts by reading the configuration file `OMNeT++pp.ini` containing initialization values.

Input Data

The input data to the model comes from the OTB logger. After setting up a particular vignette, it is simulated and the OTB logger records all of its PDUs into an output file that is later converted to text. OTB generates Persistent Object PDUs (PO_PDUs), which are a specialization of the general category of PDUs.

```
<dis204 po_variable PDU>:
dis_header.version=4
dis_header.exercise=1
dis_header.kind=250
dis_header.family=140
dis_header.timestamp=:01:33.432 (rel)
dis_header.sizeof=196
po_header.po_version=28
po_header.po_kind=2
po_header.exercise_id=1
po_header.database_id=1
po_header.length=147
po_header.pdu_count=7905
do_header.database_sequence_number=0
do_header.object_id.simulator.site=1082
do_header.object_id.simulator.host=23825
do_header.object_id.object=685
do_header.world_state_id.simulator.site=0
do_header.world_state_id.simulator.host=0
do_header.world_state_id.object=0
do_header.owner.site=1533
do_header.owner.host=23825
do_header.sequence_number=1
do_header.class=11
do_header.missing_from_world_state=0
reserved9=0
variable.total_length=132
variable.expanded_length=7812
variable.offset=0
variable.length=132
variable.obj_class=8
variable.data="Mine Pallet US M75
```

Figure 5. Example of a PO_PDU

Figure 5 is an example of a short PO_PDU. From among all the fields, the most important to our simulation are the site identification (1533), the length in bytes (147) and the timestamp (:01:33.432) interpreted as 1 minute, 33 seconds and 432 milliseconds. The timestamp represents the time the entity generated this PDU and put it on the output queue.

Simulation Results

The figures included in this article are labeled “Experiment # 3” because they correspond to the third experiment in a series of four. The experiment involved PDUs from six senders. The remaining 19 computers were listeners. The sites sending information were assigned to computers in separate airplanes and to the ground station. Two types of analyses were performed. The first one is a static analysis of the input data with no simulation involved. This analysis is subdivided into 2 categories: distribution of PDUs and minimum bandwidth requirements.

Distribution of PDUs

```
<dis204 fire PDU>:                  23
<dis204 po_objects_present PDU>:  682
<dis204 po_minefield PDU>:         14
<dis204 minefield PDU>:           117
<dis204 iff PDU>:                 851
<dis204 acknowledge PDU>:          36
<dis204 stop_freeze PDU>:           3
<dis204 po_line PDU>:             912
<dis204 po_task_authorization PDU>: 6
<dis204 po_task_frame PDU>:       382
<dis204 po_message PDU>:          119
<dis204 aggregate_state PDU>:     256
<dis204 po_delete_objects PDU>:   110
<dis204 po_parametric_input PDU>:1196
<dis204 laser PDU>:                 3
<dis204 detonation PDU>:           25
<dis204 po_fire_parameters PDU>:  713
<dis204 po_simulator_present PDU>:370
<dis204 entity_state PDU>:      28569
<dis204 mines PDU>:               386
<dis204 po_point PDU>:            659
<dis204 po_task_state PDU>:     11960
<dis204 signal PDU>:              237
<dis204 po_task PDU>:            2274
<dis204 po_unit PDU>:            1793
<dis204 transmitter PDU>:        8642
<dis204 start_resume PDU>:          3
                    Total PDUs = 60341
-----------------------------------
Site Assignment:
Site 1519 (50230 PDUs): plane=0, node=0
Site 1526 ( 1056 PDUs): plane=1, node=0
Site 1529 (  483 PDUs): plane=2, node=0
Site 1533 (  553 PDUs): plane=3, node=0
Site 1538 (  637 PDUs): plane=4, node=0
Site 1532 ( 7382 PDUs): ground station
```

Figure 6. Frequency distribution of PDUs

Figure 6 shows a frequency distribution of all the types of PDUs involved in the experiment, as well as the

assignment of sites to model nodes. The assignment was based on the number of PDUs generated by each site, giving plane 0 the greatest number of PDUs and ground station the next greatest.

Minimum Bandwidth Requirements

The PDUs of all the sites were merged into one single stream of data and sorted according to their timestamps prior to any bandwidth computation. This was done because in the DIS protocol all the PDUs are broadcasted and any listening site will have to receive the PDUs from all the generating sites as one single stream of data. Then, without using simulation, a separate program calculated the minimum instant bandwidths by dividing the total simulation time into small time intervals of 2 seconds each, and computing the ratio of volume of data transmitted in each interval to the length of the interval.

Figure 7 shows the graph of minimum bandwidth requirements at each time interval. PDUs indicate that the activity starts at second 1035 and ends at second 2550 for a total time of 25 minutes and 15 seconds of simulation time.

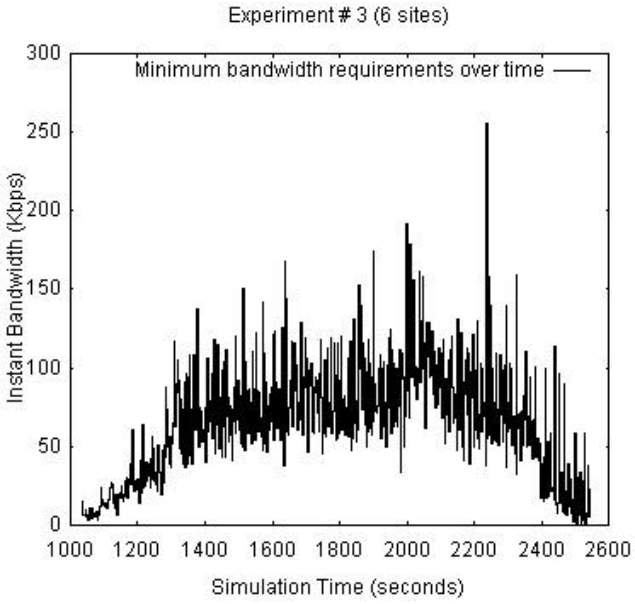

Figure 7. Minimum bandwidth requirements

In the static analysis, overheads like retransmissions, packet losses, or collisions are not considered. Therefore, the resulting bandwidth estimates can be interpreted as an absolute lower bound for the actual required bandwidth. Because a time gap must exist between packets as indicated by the IEEE Standard 802.11 (1997), it was set to 50 microseconds in this analysis. From the graph in Figure 7, the static analysis indicates that the maximum required bandwidth is near 256 Kbps, but the majority of the time the required bandwidth is less than 200 Kbps.

The second analysis performed corresponds to results obtained by running the simulator. The analysis is subdivided into 4 categories: slack time analysis, travel time analysis, queue length analysis and collision analysis.

Slack Time Analysis

The slack time for each node generator is defined as the difference between the timestamp of each PDU and the current simulator time at the moment the PDU is read from the input file. If the difference is positive then the generator is ahead of the planned schedule, otherwise it is behind it. Thus, a negative slack time indicates that the channel bandwidth is not enough to transmit the required PDUs without delay.

Figure 8 shows the slack time for all the units (routers and ground station) when the wireless channels are set to 64 Kbps

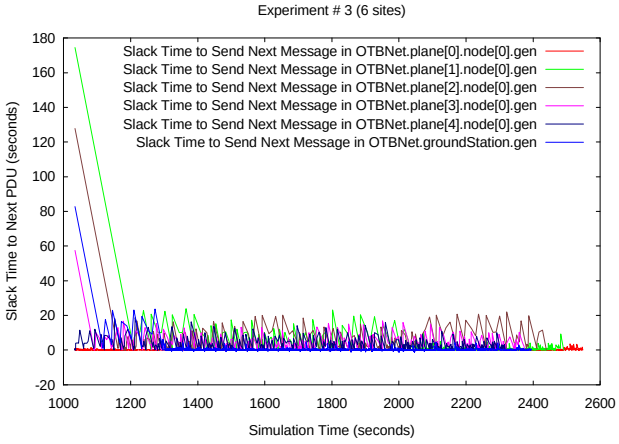

Figure 8. Slack times at each transmitting node using 64 Kbps in the wireless channels

Although Figure 8 gives the impression that no negative slacks are produced, a close up near zero in the Y axis reveals that many negative spikes do exist, most of them generated by the ground station. This is explained by the fact that the generators onboard the planes are directly connected to high speed Ethernet buses (100 Mbps), while the ground station is connected to a low speed wireless channel (64 Kbps).

Figure 9 shows a close up of the slack time for the ground station at 64 Kbps. An increase in the wireless bandwidth decreases the negative spikes, but does not eliminate them completely. One contributing factor to the negative spikes is the fact that during certain events OTB produces sequences of PDUs having exactly the same timestamp. For instance, sequences of 8 or more "`po_fire_parameters`" PDUs having the same timestamp were

detected. Each PDU requires some positive transmission time, and so when the next PDU arrives its timestamp is previous to the current time and a negative spike starts to build. Further research is being conducted to analyze the composition of the PDUs participating in the negative spikes and to propose ways to eliminate these spikes. One possibility is to group together PDUs of the same type and length into one single packet.

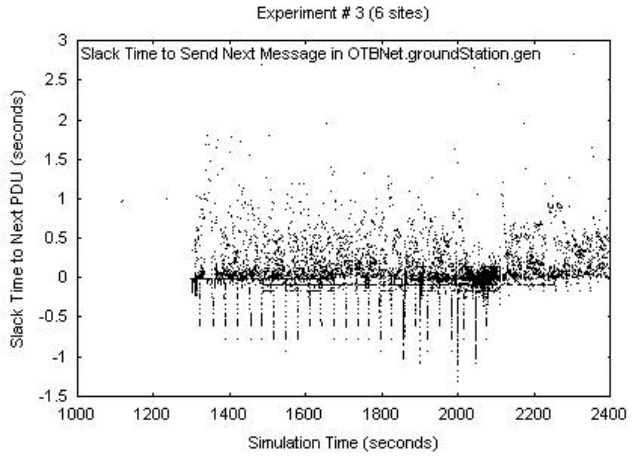

Figure 9. Slack time at ground station using a bandwidth of 64 Kbps

Travel Time Analysis

The travel time is the difference between the sending time of a PDU from a node generator and the arrival time to a node sink. All the transmission times, propagation times and waiting times in router queues are part of the travel time. Figure 10 shows the travel time of PDUs measured by the sink at node 0, plane 0, using 64 Kbps on the wireless links.

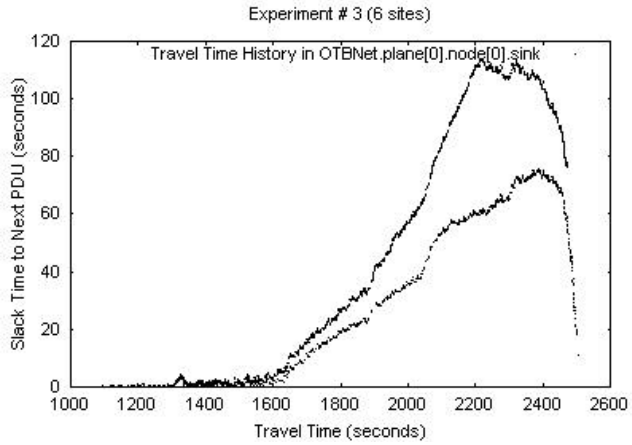

Figure 10. Travel times measured at node 0, plane 0

At node 0 the graph clearly shows two traces corresponding to two types of PDUs. The PDUs that take longer to arrive come from the ground station. These PDUs had to wait on the satellite queue as well as on the router queue. On the other hand, the PDUs coming from computers onboard the other planes had to wait on the router queue only. As seen, at 64 Kbps the travel times of most PDUs are completely unacceptable. Some PDUs took more than 100 seconds since the time they were sent to the time they arrived.

Although not shown, the ground station presents a similar behavior due to the relatively long queue and transit times associated with the satellite. An increase in the bandwidth produces a significant reduction in travel times. At 200 Kbps, the travel times to the ground station are less than 1 second. Considering that the minimum travel time is about 0.25 seconds due to the distance from the satellite to Earth, latencies less than 1 second are within the same order of magnitude from being optimal. At 200 Kbps, a better packet scheduling policy could diminish the negative spikes, especially if OTB were set to deliver the PDUs in a less burst mode, which is also the topic of future research.

Queue Length Analysis

There is a message queue at the satellite and at each router. Every time a PDU arrives to a router or satellite, the number of messages in the system is counted including the just arrived PDU, the PDUs in the queue and any one being serviced, and the tally is recorder along with the arrival time.

The two most important queues to analyze are the queues at the router onboard plane 0 and at the satellite. Figure 11 shows the messages in the router queue, using 64 Kbps in the wireless channels.

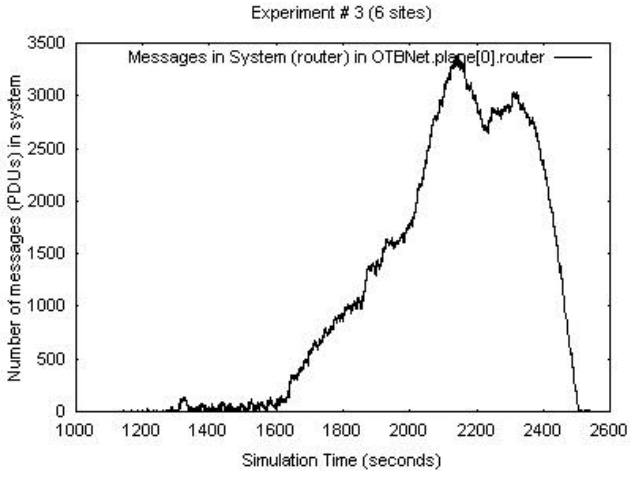

Figure 11. Messages in router queue at plane 0

As observed, the queue length becomes really unacceptable, reaching more than 3000 messages during the worst periods. Similarly, the queue at the satellite has a

peak of more than 2200 messages. On the other hand, routers at other planes have acceptable queues. For instance, plane 3 (graph not shown) has a queue with a maximum of 23 messages. The reason is that the corresponding node transmits only 553 PDUs, a number easily handled by the router.

Simulation runs using bandwidths of 64, 200, 512 and 1024 Kbps in the wireless channels showed the effect of a bandwidth increase over the queue length in routers and in the satellite. The results indicated that just by increasing the bandwidth to 200 Kbps, the queue length decreases to less than 60 and less than 40 messages at plane 0 and at the satellite, respectively, in the highest peaks, which is quite significant.

Collision Analysis

The satellite and routers keep separate counters of collisions on each of the buses they are connected to. The satellite and the routers are connected to 2 and 3 links, respectively, as explained in the Model Design section. Each time a collision is detected, the corresponding counter is updated and its value along with the current simulation time is recorder for future processing. OMNeT++ includes procedures to collect and plot such information. The buses were programmed in such a way that packets transmitted by a module are not returned to its sender, even if they collide. Therefore, active senders are not good candidates to monitor collisions. In this analysis, the router at plane 7 was chosen to count collisions because none of the nodes onboard plane 7 is an active sender, making the router a good indicator.

The simulation results at 64 Kbps indicate that the highest collision rate occurred in the bus connecting the satellite to the planes and was close to 12 collisions per second. More than 6000 collisions were detected at 64 Kbps, which represents approximately 10 % of the total number of PDUs. At 200 Kbps, the router at plane 7 detected a maximum of 4 collisions per second. At this bandwidth, the total number of collisions was near 4500, or 7% of all the PDUs. Collision accumulation in plane 7 at different bandwidth rates is given in Figure 12.

The collisions were calculated as the sum of collisions detected in the three buses the router is connected to. However, the main component comes from the wireless link between the airplane and the satellite. The behavior of this link depends on the assignment of OTB transmitting sites to computer nodes. For example, if the PDUs currently assigned to the ground station were assigned to computer node 1 onboard plane 0, the Ethernet link running at 100 Mbps will produce fewer collisions than a wireless link running at 64 Kbps due to the shorter transmission times.

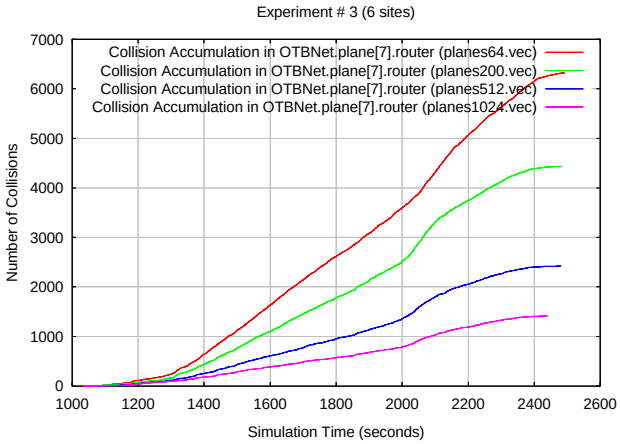

Figure 12. Collision accumulation at plane 7 at 64, 200, 512 and 1024 Kbps

Table 1 shows statistics about the total number of collisions, percentage of all the PDUs, and average number of collisions per second at different bandwidths.

Table 1. Collision accumulation, percentage and average number of collisions per second at different bandwidths

64KBps	6300 coll.	10 %	2.5 coll/sec
200KBps	4400 coll.	7.3 %	1.7 coll/sec
512KBps	2300 coll.	3.8 %	0.9 coll/sec
1024KBps	1300 coll.	2.1 %	0.5 coll/sec

Conclusions

As predicted by the static analysis, a bandwidth of 64 Kbps in the wireless links is insufficient to handle embedded training traffic under a DIS protocol. Latencies of more 100 seconds were detected for traffic coming from the ground station where the simulated Opposing Force resides. A significant improvement was achieved at 200 Kbps, where latencies less than 1 second were almost always the case for messages received at the ground station.

At the router in plane 0 and at the satellite, the queue lengths changed from 3400 and 2200 (max. peak) to less than 60 and less than 40 (max. peak) messages, just by increasing the bandwidth from 64 to 200 Kbps. As seen by the router at plane 7, collisions decrease and become manageable as the bandwidth increases.

Regarding modeling tool used, OMNeT++ is easy to learn and versatile to use for modeling computer networks. Several network models have been built that are available for downloading from the Internet. These models can be tailored to the user needs. OMNeT++ offers the possibility of running the simulation at several speeds with or without animation, including faster-than-real time. The results can be displayed and plotted on the screen as the simulation progresses, and also can be stored into text files for further processing.

References

IEEE (1995). *Std 1278.1. Standard for Distributed Interactive Simulation — Application Protocols*. IEEE Computer Society Press.

IEEE (1995). *Std 1278.2. Standard for Distributed Interactive Simulation — Communication Services and Profiles*. IEEE Computer Society Press.

IEEE (1996). Std 1278.3. *Recommended Practice for Distributed Interactive Simulation — Exercise Management and Feedback.* IEEE Computer Society Press.

IEEE (1997). *Std 802.11 Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications*. IEEE Computer Society Press.

IEEE (1998). Std 1278.1a. *Standard for Distributed Interactive Simulation — Application Protocols*. IEEE Computer Society Press.

Tcl Developer Xchange (2003). Retrieved November 15, 2003 from http://www.tcl.tk/.

Varga, A. (2001). OMNeT++ Discrete Event Simulation System. In *Proceedings of the European Simulation Multiconference* (ESM'2001). June 6-9, 2001. Prague, Czech Republic. User manual retrieved December 1, 2003 from http://www.omnetpp.org/index.php.

Varga, A. (2002). OMNeT++. In the column *"Software Tools for Networking", IEEE Network Interactive*. July 2002, Vol. 16 No. 4.

Zimmer, J. (1998). *Tcl/Tk for Programmers*. IEEE Computer Society Press.