PDU Bundling and Replication for Reduction of Distributed Simulation Communication Traffic[1]

Juan J. Vargas[2], Ronald F. DeMara[2], Michael Georgiopoulos[2], Avelino J. Gonzalez[2], Henry Marshall[3]

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

[3] Simulation & Training Technology Center
RDECOM, AMSRD-STTC
12423 Research Parkway
Orlando, FL 32826
Henry.A.Marshall@us.army.mil

Communication bandwidth and latency reduction techniques are developed for Distributed Interactive Simulation (DIS) protocols. DIS Protocol Data Unit (PDU) packets are bundled together prior to transmission based on PDU type, internal structure, and content over a sliding window of up to *C* adjacent transmission requests, for $1 < C < 64$. At the receiving nodes, the packets are replicated as necessary to reconstruct the original packet stream. Bundling strategies including *Always- Wait*, *Always-Send, Type-Only* prediction, *Type-Length* prediction, and *Type-Length-Time* prediction are developed and then evaluated using both heuristic parameters and a gradient descent back-propagation neural network.

Several communication case studies from the One Semi-Automated Forces (OneSAF) Testbed Baseline (OTB) are assessed for multiple-platoon, company, and battalion-scale force-on-force vignettes consistent with Future Combat Systems (FCS) Operations and Organizations (O&O) scenarios. Traffic is modeled using the OMNeT++ discrete event simulator models and scripts developed for a hierarchical communication architecture consisting of eight enroute C-17 aircraft each carrying three Ethernet-connected M1A2 ground vehicles, a wireless flying LAN based on Joint Forces Command's Joint Enroute Mission Planning and Rehearsal System (JEMPRS) for Near-Term (JEMPRS-NT) and follow-on bandwidth capacities. The simulation traffic includes Opposing Force (OPFOR) control via a CONUS-based ground station via its corresponding satellite links. Different bandwidth capacities are simulated and analyzed PDU travel time and slack time, router and satellite queue length, and number of packet collisions are assessed at 64 Kbps, 256 Kbps, 512 Kbps, and 1 Mbps capacities. Results indicate that a *Type-Length* prediction strategy is capable of reducing travel time up to 85%, slack time up to 97%, queue length up to 98% on bandwidth restricted channels of 64 Kbps.

Keywords: DIS protocol, OneSAF Testbed Baseline OTB, PDU replication, PDU bundling, network bandwidth.

INTRODUCTION

One Semi-Automated Forces (OneSAF) is a U.S. Army computer generated force simulation system capable of running training simulations and rehearsing missions within a digital environment [Witman 2001]. In this paper, a modeling environment for assessing OneSAF communication during mission rehearsal of Future Combat System (FCS) vignettes is developed and applied to optimize simulation communication traffic. Several different FCS vignettes were prepared and simulated using the OneSAF Testbed Baseline (OTB) on a Local Area Network. Traffic logs were created from the participating sites under the Distributed Interactive Simulation (DIS) protocol as defined in the IEEE Standards 1278.1 (1995), 1278.2 (1995), 1278.3 (1996) and 1278.1a (1998). The fundamental communication elements under DIS, called Protocol Data Units (PDUs), were logged including relevant information about PDU type, length and timestamp then later extracted for modeling assessment of alternative transmission scheduling strategies.

In particular, the *MR1* vignette illustrating a mission rehearsal operation while en-route to deployment was used to generate PDU traffic logs. This vignette was partly based on and extends TRADOC PAM 525-3-90 FCS Operations & Organizations (O&O) document, dated 22 July 2002, "Annex F – Unit of Action Vignettes." The duration of the vignette is approximately 25 minutes of simulation time. It involves Entry Operations and Maneuver to Attack of a

[1] This work was supported in part by the U.S. Army Research Development and Engineering Command (RDE-COMM) as part of the Embedded Combined Arms Team Trainer and Mission Rehearsal (ECATT-MR) Science and Technology Objective (STO) contract N61339-02-C-0097.

battalion-sized unit tasked with pursuing an enemy delaying force immediately upon landing. The lead elements (Alpha Company) detect a fortified position between the main elements and the target enemy force. Four RAH-66 Comanche helicopters are deployed and follow closely. Next, the East friendly forces, begin to advance on the enemy position, however, they must traverse minefields during their pursuit. The enemy force flees southward from the North and East force. The South force engages the enemy, and is assisted by the North and East forces.

The vignette was executed on OTB and a log file containing all the network traffic (PDUs) transmitted was recorded and used as input to a modeling environment for purpose of studying the communication traffic under several bandwidths. Figure 1 depicts the communication architecture of the model used for rehearsal and training with the MR1 vignette. It consists of eight airplanes flying in formation towards deployment to support En-route Mission Planning and Rehearsal (EMPR) while the vehicles are en-route to the combat destination. Each aircraft carries three ground vehicles, and each vehicle contains a simulation station executing the MR1 vignette using OTB. The number of planes and simulation stations onboard is variable in the model. The three simulation stations on each plane are connected via a hardwired Ethernet bus at 100 Mbps. Connections from plane to plane are achieved via routers and wireless links. A ground station is connected to the flying network through the satellite link as shown. Possible values for wireless bandwidths range from 64 Kbps to 1024 Kbps. This communication model is consistent with the JEMPRS Flying LAN network described in [Frontlines 2002]. The purpose of the research described in this paper is to assess the bandwidth required in the wireless links to support EMPR and develop improved strategies that more effectively utilize the bandwidth available.

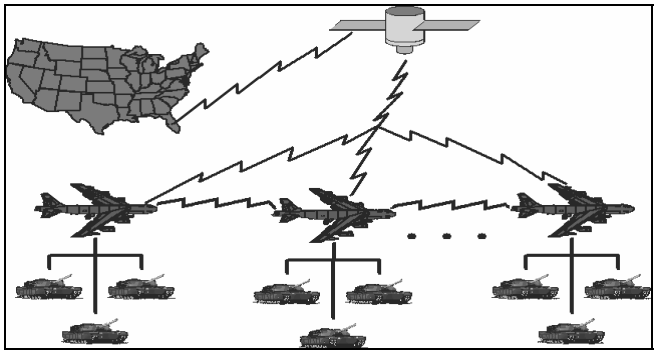

Figure 1. JEMPRS Flying LAN network for EMPR

In summary, the model consists of a collection of processing nodes and routers interconnected by different media at several bandwidths. The computer nodes broadcast packets at different rates. While it is possible to model the network traffic by creating an artificial packet generation that follows some probability distribution over time, a more representative approach was developed using the actual traffic generated by OTB during execution of the MR1 vignette. A log of the actual packets generated by OTB included the PDU timestamp which is used for packet generation in the traffic modeling studies in place of a random distribution function. This provides much more realism to the communication traffic analysis, especially for the bursty nature of traffic observed during EMPR execution.

Although DIS traffic was simulated in this research, other protocols like High Level Architecture (HLA) have been proposed as alternatives to DIS. The Defense Modeling and Simulation Office (DMSO) developed HLA as part of their goal to increase interoperability and promote reuse of simulations and their components. HLA defines the concept of a federation, which is a subset of interacting simulations. Many features not found in DIS are included in HLA to reduce traffic network, like Data Distribution Management that improves scalability by limiting the network traffic each federate has to process [Ceranowicz 2002]. The federation network traffic is segmented such that federates receive messages from only those segments that can affect them. However, currently fielded simulators used during EMPR training utilize the DIS protocol studied here.

The OMNeT++ discrete event simulator [Varga 2003] was used as a platform for communication assessment. Each of the elements shown in Figure 1 was realized as a C++ module in OMNeT++. Figure 2 shows one screenshot of the OMNeT++ tool. In this screenshot, neither the ground vehicles carrying computer nodes, nor the LAN links connecting them, nor the routers in each plane are depicted. Instead, three horizontal bars represent wireless channels. The upper bar is used for Wireless Plane-to-Plane communications (WPP link), the middle bar represents Wireless Satellite-to-Plane communications (WSP link), and the lowest bar represents Wireless Ground-Station-to-Satellite communications (WGS link).

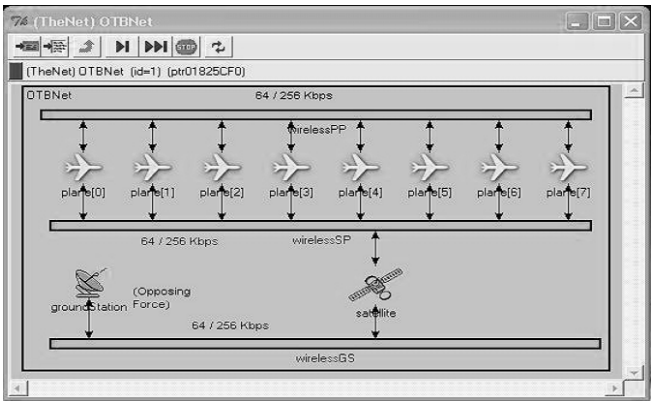

Figure 2. OMNeT window depicting JEMPRS network

The entities comprising the model include the four communication links (LAN, WPP, WSP, WGS), the computer nodes containing a generator and a sink for packets, routers connected to the LAN, WPP and WSP links, the satellite, and the ground station. The behavior of each of these objects is explained below.

Communication Channels

Starting with the simplest element, the *bus* object is used to model all communication links and is considered ideal, with FIFO channels and zero message loss. A bus contains input and output connectors separated by known distances. The bus length does not attenuate transmitted signals, and there is no interference or noise between bus channels. Each bus is configured to operate with a specific bandwidth and propagation delay. When a message enters through one of its input connectors, the bus delivers it to each of the output connectors at different times depending on the distance and propagation delay of the medium.

If (IC_i, OC_i) is a pair of input and output connectors located at a distance d_i from one of the bus endpoints, p is the propagation delay of the bus (in *seconds/meter*), b is the bus bandwidth (in *bps*), and a message of length n bits arrives into IC_i at time t, then at the time that message reaches an output connector OC_j the following relationships can be defined:

Distance traveled = $| d_i - d_j |$
Propagation delay = *Distance traveled* * p
Transmission time = n / b
Start time at OC_j = t + *Propagation delay* = $t + | d_i - d_j | * p$
End time at OC_j = *start* + *transmission time* = $t + | d_i - d_j | * p + n / b$

The start and end times at OC_j are useful to determine collisions between one or more PDUs. If a message has the same start and end times during the same interval as the start and end times at OC_j of any other message, then a collision occurs. We adopted a strategy where collided packets are discarded to obtain a reasonable estimate of traffic after observing the relatively low number of collisions produced during the simulations, less than 8% in the worst case, as indicated in the Section "Collision Analysis" below.

Routers and Satellite

There is a router onboard each airplane. A router is connected to the LAN, WPP and WSP links, as indicated in Figure 3. Because DIS traffic is broadcasted, a PDU coming from one input connector must be propagated to the other output connectors according to Table 1.

Table 1. Routing table in broadcast mode

Input link	Output link
LAN	WPP and WSP
WPP	LAN
WSP	LAN

Routers maintain an M/M/1 queue of input messages. Every time a new message arrives, the router records statistics about the number of messages in the queue at that time. The message length, the IFS gap, and the output bandwidth determine the service time.

The satellite behaves like a router with only two links, namely the WSP and WGS links. The satellite also maintains a queue of messages and calculates statistics as does the other routers, for messages at the ground station.

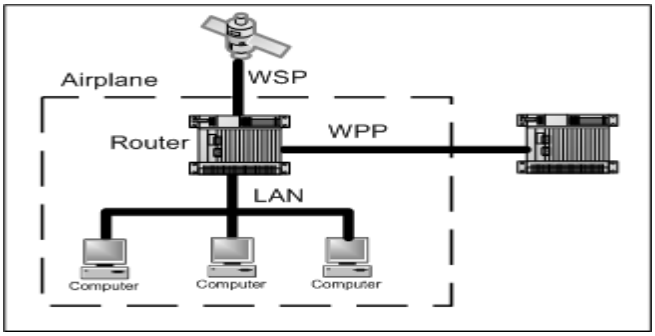

Figure 3. Router onboard a plane and its connections to the LAN, WPP and WSP links

Flying LAN Nodes and CONUS Ground Station

Each workstation in the model consists of a computer node that contains two other sub modules: the *generator* and the *sink* of PDUs. These computer nodes are connected to the LAN link.

Packets to and from the CONtinental U.S. (CONUS) ground station are handled identically as the simulation stations aboard the flying LAN, except that links from the ground station are connected to the WGS channel only.

PDU Generator

This module reads in the PDUs from a summary file containing the type, length and timestamp. It outputs packets to the LAN or WGS link depending on the location of the workstation. After sending a packet, an inter-frame space (IFS) or time gap of 50 μs is added, in accordance with the ANSI/IEEE protocol 802.11 (1999). Also, a 5 μs delay corresponding to the generator service time is added to the transaction.

Generators do not use random numbers drawn from a particular distribution to simulate traffic. Instead, the original type, length and timestamp of PDUs logged by OTB are applied, giving more realism to the simulation and making them readily repeatable and also allowing them to be correlated with events in the EMPR training scenario. This approach is more representative for intervals when bursts of transmissions occur in real OTBSAF traffic, which cannot be modeled using any pre-existing normal or Poisson distribution function.

PDU Sink

The sink consumes PDUs and maintains statistics about the number of frames received, the latency of each frame, and number of collisions detected at each node.

ANALYSIS OF LOGGED PDUs

The PDUs produced by OTB during the execution of the MR1 vignette represent 27 different types of varying lengths. Table 2 lists the PDU types, number of bytes sent, and the percentages of each one in the total sample, including the generators. It can be observed that Entity State PDUs (ESPDUs) make up the majority of the network traffic, reaching almost half (47.3%) of the total number of PDUs. This observation has been pointed out in [Bassiouni et al. 1997] where ESPDU traffic reached as high as 90% of the PDU traffic in some cases. OTB also uses a variety of Persistent Object (PO) PDUs. For the purpose of this research regarding communication traffic loading, the fields of interest in PO_PDUs are sufficiently comparable to IEEE standard PDUs so that the IEEE description can be applied.

Table 2. PDU frequency and transmission volume

PDU Type	# PDUs	# Bytes	% PDUs	#% Bytes	#
Laser	3	264	0.005	0.002	
start_resume	3	132	0.005	0.001	
stop_freeze	3	120	0.005	0.001	
po_task_authorization	6	388	0.010	0.003	
po_minefield	14	5384	0.023	0.043	
fire	23	2208	0.038	0.018	
detonation	25	2550	0.041	0.021	
acknowledge	36	1152	0.060	0.009	
po_delete_objects	110	4216	0.182	0.034	
minefield	117	42120	0.194	0.339	
po_message	119	69020	0.197	0.556	
signal	237	19896	0.393	0.160	
aggregate_state	256	37888	0.424	0.305	
po_simulator_present	370	34040	0.613	0.274	
po_task_frame	382	87984	0.633	0.709	
mines	386	396088	0.640	3.19	
po_point	659	55356	1.09	0.45	
po_objects_present	682	577952	1.13	4.65	
po_fire_parameters	713	376464	1.18	3.03	
iff	851	51060	1.41	0.41	
po_line	912	115524	1.51	0.93	
po_parametric_input	1196	165440	1.98	1.33	
po_unit	1793	1161864	2.97	9.36	
po_task	2274	399744	3.77	3.22	
transmitter	8642	898768	14.3	7.24	
po_task_state	11960	3052824	19.8	24.6	
entity_state	28569	4857328	47.3	39.1	
Totals	60341	12415774	100%	100%	

Each type of PDU has its own internal structure consisting of fields and values of different sizes. A study of all the PDUs logged in the vignette indicated that if two PDUs are of the same type and length then they have consistent field types, but with different data contents in selected fields.

Another observation from the logged PDUs is the fact that OTB schedules some sequences of consecutive PDUs using timestamps over very narrow intervals, as in the sample sequence shown in Table 3. This can cause congestion from sending several packets within a short time interval, creating negative spikes in the slack time of the routers. In most cases, consecutive PDUs of equal type and length differed in a only few fields, yielding the possibility of bundling them in a single PDU to improve transmission characteristics.

Table 3. Sample sequence of PDUs scheduled by OTB within a narrow time interval (shaded). Although the rightmost digit of time stands for milliseconds, OTB has microsecond timestamping resolution internally.

PDU Type	Byte	Timestamp
po_task_state	152	33:18.776
entity_state	176	33:18.821
entity_state	144	33:18.825
transmitter	104	33:18.825
signal	72	33:18.825
transmitter	104	33:18.825
iff	60	33:18.825
po_task_state	48	33:18.825
po_task_state	1016	33:18.825
po_task_state	1016	33:18.825
po_task_state	1016	33:18.825
entity_state	176	33:18.878
po_task_state	480	33:18.894
po_task_state	64	33:18.894
po_task_state	64	33:18.996
…	…	…
po_task_state	64	36:38.043
po_fire_parameters	528	36:38.043
po_fire_parameters	528	36:38.043
po_fire_parameters	528	36:38.043
po_fire_parameters	528	36:38.043
po_fire_parameters	528	36:38.043
po_fire_parameters	528	36:38.043
po_fire_parameters	528	36:38.043
po_fire_parameters	528	36:38.043
po_fire_parameters	528	36:38.043
po_fire_parameters	528	36:38.043
po_fire_parameters	528	36:38.043
po_fire_parameters	528	36:38.043
po_fire_parameters	528	36:38.043
po_task_state	840	36:38.043

REPLICATION OF PDUs

The observations and analysis of those PDUs participating in negative spikes lead to the proposal of a new scheme for bundling PDUs that can be seen as a kind of high-level lossless compression because the resulting block still conserves the characteristics of a PDU, perhaps of a different type, and so it is subject to further compressions. In fact, the proposed bundling does not remove the redundancy within the same PDU: fields filled with zeros in the reference PDU will continue being the same length of zeros. Only the redundancy resulting from the similarities between consecutive PDUs is removed. Therefore, other

traditional compression mechanisms are applicable and recommended after bundling.

Replication is the inverse procedure of bundling. When a bundled block arrives to a destination, the individual PDUs are extracted or replicated from it. Replication is independent of other data compression techniques because it is targeted at the PDU level and the resulting traffic is of PDU type. Therefore, even if there are no plans to modify the transport protocol in effect (by compressing TCP/IP headers, for instance), the reduction of PDU packets to increment the bandwidth availability by using replication is still applicable.

Numerous results for latency and delay under similar configurations for 64 Kbps, 128 Kbps, 200 Kbps, 256 Kbps, 512 Kbps, and 1024 Kbps are presented in [Vargas et al. 2004]. A straightforward solution to the negative spikes in the slack time is to bundle PDUs of the same type and length into longer ones, thus eliminating redundancy in corresponding fields of consecutive PDUs.

Specification of Bundling

Given a set $N = \{1, 2, \ldots, n\}$ of indexes and two consecutive PDUs $A = (a_1, a_2, \ldots, a_n)$ and $B = (b_1, b_2, \ldots, b_n)$, where A and B are of the same type and the a_i and b_i represent PDU fields, and $S \subseteq N$ is a subset such that $a_i = b_i$ for all $i \in S$, then the bundle of A and B is defined as $A\&B = (a_1, a_2, \ldots, a_n, [\ (b_j, j)\ _{j \in N-S}\ ])$. A is called the *base* PDU in the bundle. The definition can be extended to an arbitrary number of PDUs. For example, given the PDUs: $A = (a_1, a_2, a_3, a_4)$, $B = (b_1, b_2, b_3, b_4)$, and $C = (c_1, c_2, c_3, c_4)$, such that $a_2 = b_2 = c_2$, $a_3 = b_3$, $a_4 = c_4$, then the bundle $A\&B\&C$ is the aggregate PDU such that $A\&B\&C = (a_1, a_2, a_3, a_4, [(b_1, 1), (b_4, 4)], [(c_1, 1), (c_3, 3)])$.

From the information contained in the n-tuple it is straightforward to reconstruct the original PDUs A, B, and C. Each component (b_j, j) indicates that the value b_j replaces the field j in the base PDU. In a practical implementation, j could be a pointer or an offset into the base PDU.

Related Work

The above bundle differs from other proposed transmission aggregations in several ways. First, the resulting bundle conserves the basic characteristics of the base PDU and can be subject to further bundling and/or compression algorithms. In [Bassiouni et al. 1997] consecutive PDUs are concatenated in a single packet even if their types are different, and field redundancy is not eliminated. A delta-PDU encoding technique is mentioned in [OTA 1995] consisting of PDUs that carry changes respect to a reference PDU initially given. [Wills et al. 2001] describes several bundling techniques generally applicable to Web pages under the TCP/IP protocol suite, but none are specific to the DIS protocol. A protocol called DIS-Lite developed by MäK Technologies [Taylor 1995, Taylor 1996a, Taylor 1996b, Purdy and Wuerfel 1998] splits the Entity State PDU into static and dynamic data PDUs, so that the static information is sent once and the changes (dynamic PDUs) are subsequently transmitted as separate PDUs. According to [Fullford 1996] by eliminating redundancy, DIS-Lite can perform between 30% and 70% more efficiently than DIS. DIS-Lite includes also several other improvements not related to the combination of individual fields from a set of similar PDUs. These improvements complement related predictive strategies developed for conserving simulation bandwidth [Bahr 96] [Henninger 01]. DIS-Lite was designed to take optimize ESPDUs. Our bundling approach can optimize any consecutive PDUs of equal type and length. Also, the reference PDU is included in the bundle in our approach, so that the delta-PDUs are not sent separately incurring additional header overhead.

Input Data

When OTB is executing a vignette, its logger records all the generated PDUs into an output file for analysis. The OMNeT model reads relevant PDU data from a text file created from the OTB transmission log. AWK scripts were written to read this file and extract the type, length and timestamp of each PDU into a summary PDU file, along with two counters. One counter represents a local PDU identifier for the generating site, and the other is a global identifier from among all the participating sites. Figure 4 depicts a general view of the steps involved in the traffic modeling process.

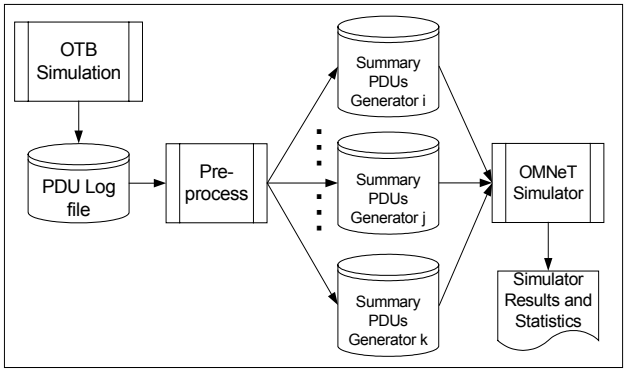

Figure 4. Overview of the simulation process

For the particular vignette used in this report, six summary files were generated, having 7382, 1056, 483, 553, 637 and 50230 PDUs, respectively. The largest file was assigned to the generator at the ground station, because the generator is directly connected to a slow wireless link, imposing a high load to it, and permitting a more realistic study of the performance of congested wireless channels under different bandwidths. This also corresponds to the case where the opposing force in the OTB simulation is controlled at the CONUS ground station.

Besides the PDU counters, two other characters, *S* for send and *W* for wait, were appended to the summary PDUs to provide information about the strategy to follow after processing each PDU. Four possible strategies are *Neural*

Network prediction and three variants of *Ideal Prediction* which will be described below. Table 4 shows a sample of summary PDUs.

Table 4. Sample of summary PDUs

Hex time	Size	Decimal Time	Local ID	PDU Type	Global ID	Prediction
59e1a736	100	21:03.957	15	objects_present	4721	SSSS
5a8738fc	92	21:13.052	16	simulator_present	5039	SWWS
5c357768	92	21:36.686	17	simulator_present	5728	SSSS
5c357768	100	21:36.686	18	objects_present	5729	SSSS
5ca513f0	84	21:42.817	19	point	5972	WWWW
5ca513f0	84	21:42.817	20	point	5973	WWWW
5ca513f0	84	21:42.817	21	point	5974	WWWW
5ca513f0	84	21:42.817	22	point	5975	WSSS
5ca513f0	80	21:42.817	23	task_state	5976	SSSS
5ca513f0	96	21:42.817	24	task	5977	SSSS

Implementation of PDU Bundling

The basic idea behind the PDU bundling strategy is that if consecutive PDUs of the same type and length are scheduled within some predefined time interval, they can be bundled and delivered as a single packet. This timeout value was set to 0.1 seconds after an analysis of traffic bursts to be optimized indicated a burst duration less than 0.1 seconds. Analysis of PDUs in the log file for a given vignette also indicated that the type and the size of a PDU are good indicators of feasibility for bundling. In other words, assume that PDUs $A = (a_1, a_2, a_3, \ldots, a_n)$ and $B = (b_1, b_2, b_3, \ldots, b_m)$ are such that *type(A) = type(B)* and *length(A) = length(B)*, then $n = m$ and *field-type*(a_i) = *field-type*(b_i), where *type*, *length* and *field-type* are functions that return the type, the length in bytes and the type of a field in a PDU, respectively. If two PDUs are of the same type and length, they are referred to as *compatible* and are candidates to be bundled.

The pseudo-algorithm of this PDU bundling strategy can be described as follows:

1) Wait until next PDU is ready for delivery. Let *A* denote that PDU.
2) *Bundle* = *A*. This is the first PDU (base PDU) in the bundle.
3) Set *timeout* = maximum time *A* will wait in the bundle (default is 0.1 seconds).
4) While (*timeout* not expired) {
   a. If next PDU is ready for delivery, let *B* be that PDU, otherwise repeat the while-loop;
      i. If *A* and *B* are compatible PDUs {
         Bundle = Bundle & B;
         B = ∅;}
      else break the while-loop}
5) Transmit *Bundle* as a single packet.
6) If *B* = ∅ then repeat from step 1)
   else *A* = *B* and repeat from step 2).

This algorithm is called *Always-Wait* because after processing a PDU, the algorithm waits for the next PDU unless a timeout is detected. When the next PDU is obtained, its type is inspected and if it is different from the type of the base PDU then the time waited was wasted. From this reasoning we can conclude that *Always-Wait* is not optimal. If there were a means to accurately predict the type and length, or at least the type, of the next PDU and the prediction indicates a type different from the type of the current base PDU, then the current bundle could be sent immediately, saving the waiting time. Such a variation of the above algorithm is introduced below.

One way to predict the next PDU based on the recent history is by using a Neural Network (NN) approach. A neural network was used to find patterns in sequences of PDUs that were observed to occur during negative spikes. These patterns can be used as a basis for predicting the type of the next incoming PDU. In this research, we set up a gradient descent back-propagation neural network that predicts the next type based on the types of the previous 44 PDUs. The neural network predicted the next PDU type with a certainty of near 70 %. Considering that there are 27 different PDU types, this percentage is significant. If NN prediction indicates that the next PDU type is the same as the current one then a "W" (for *Wait*) character is appended to the summary file, otherwise an "S" (for *Send*) is appended to it.

In this research, the comparison took place in the pre-processing stage shown in Figure 4 using actual PDU fields, which resulted in three *ideal prediction* methods in addition to the NN-based prediction. The ideal prediction methods calculate the next PDU type with 100 % certainty because they know the PDUs in advance and are referred to as *off-line algorithms*. Off-line algorithms have the luxury of making bundling decisions based on future packets that have not yet arrived by knowing the entire packet stream during off-line analysis. *On-line* algorithms see only the dynamic stream of incoming packets in the order of transmission. The first ideal prediction method we compared to considers the PDU type only, the second one considers the type and length, and the last one considers the type, length and timestamp. Given any PDU, if the next one can be bundled to it because its type (method 1), and length (method 2), and timestamp (method 3) are equal then it is held for bundling. Table 4 shows four columns with the "S" and "W" characters resulting from the application of the bundling strategies. The first column corresponds to the neural network prediction and the other three correspond to each of the perfect prediction methods. Only one column is processed during each run by the OMNeT simulator.

In addition to the four predictive methods already described, there are two others called *Always-Wait* and

Always-Send, realized by filling the decision column with all "W" or "S" characters, respectively.

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 t_{PDU} represents the timestamp of a PDU and t_{read} represents the time when the PDU was read, then $t_{slack} = t_{PDU} - t_{read}$. If the difference is positive ($t_{slack} > 0$) then the router or transmitter is ahead of the planned schedule, otherwise it is behind it. Thus, a negative slack time indicates that the channel bandwidth is insufficient to transmit the required PDUs without incurring delay.

Figure 5 shows the slack time of the generator at the ground station for different predictive algorithms. The

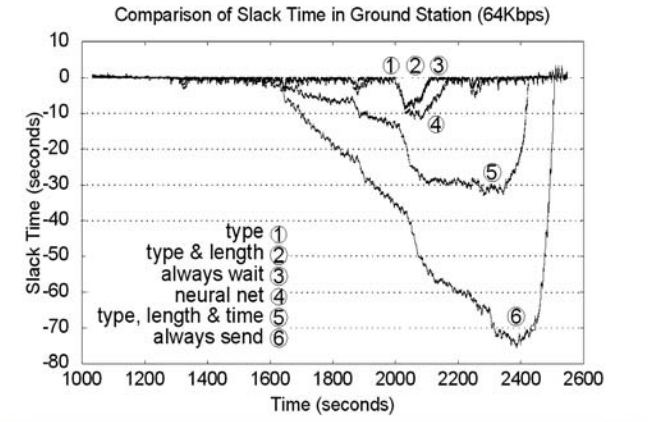

Figure 5. Slack time at ground station for several predictive strategies (64 Kbps)

graph was created assigning 64 Kbps to all wireless links and 100 ms to the timeout period. As seen in the diagram, up to second 1600 of the simulation, all of the algorithms behaved similarly, but at that point negative slack started to accumulate. The *Always-Send* algorithm incurred in the largest slack, followed by *Type*, *Type-Length,* and *Type-Length-Time* predictions. Curve 6 shows that the generator in ground station could not accomodate the traffic demands with only a 64 Kbps channel. Increasing delay in timeliness to send PDUs builds up such that over 75 seconds of latency are encountered. However, latency of more than just a few seconds would make a distributed training exercise unusable. The neural network approach performed relatively well, although not optimally because its predictions are not entirely correct. The other algorithms are among the best in this simulation, and so a close-up of their performance is shown in Figure 6. Results indicate that the neural network approach could be improved by using a better learning mechanism and/or neural network configuration. Another observation is that the decision of sending the current bundle based solely on the upcoming PDU type, is as good as the one that considers the type and the length of each PDU. Therefore, a NN approach could benefit from this observation by concentrating the effort in predicting the type only, instead of the type and the length. However, the most interesting observation comes from the fact that the *Always-Wait* algorithm is almost as good as those that consider PDU type and length, yet it is the simplest of the strategies. Because *Type* and *Type-Length* strategies are offline algorithms, they are not applicable during actual OTB simulations to predict future PDUs. On the other hand, *Always-Wait*, *Always-Send* and *Neural-Network* prediction could be used because they are online algorithms.

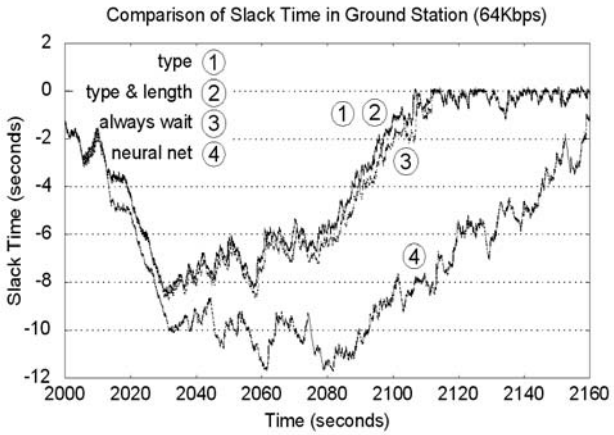

Figure 6. Comparison of negative slack for the four best algorithms

Table 5 shows the average and standard deviation in slack time for all of the combinations of algorithms and bandwidths measured at the ground station. The larger the average slack is, the better the algorithm is performing. The average was calculated considering all of the PDUs generated during the EMPR training exercise.

Table 5. Slack time average and standard deviation for all the studied algorithms and bandwidth combinations measured at the ground station

Avg: Std. Deviation:	64 Kbps	128 Kbps	256 Kbps	512 Kbps
Type	– 0.758 1.600	– 0.017 0.109	0.015 0.073	0.024 0.066
Type-Length	– 0.760 1.601	– 0.018 0.110	0.015 0.073	0.024 0.066
Always-Wait	– 0.802 1.689	– 0.017 0.109	0.016 0.073	0.024 0.066
Neural Net.	– 1.579 2.638	– 0.044 0.162	0.008 0.085	0.022 0.069
Always-Send	– 26.181 26.033	– 0.054 0.176	0.006 0.085	0.021 0.069

The *Always-Send* algorithm, which is used in DIS, is the worst of the five algorithms, and *Always-Wait* is among the best. Because, *Always-Send* corresponds to the non-

bundling algorithm case, we can infer that the type of bundling proposed here can be applied advantageously to DIS traffic. Another observation is the fact that at 64 Kbps and 128 Kbps, the average slack time was negative for all the algorithms, and for 256 Kbps and above it is positive. A negative average indicates that the corresponding bandwidth is insufficient to handle the PDU traffic. Therefore, for the EMPR vignette being studied, the wireless bandwidth should be at least 256 Kbps in this simulation to avoid incurring detectable transmission delays.

Travel Time Analysis

The travel time is the difference between the sending time of a PDU from a generator and the arrival time at the sink. All the transmission times, propagation times, and waiting times in router queues contribute to the travel time. If t_s, t_a and t_t represent the sending time, the arrival time and the travel time of a given PDU, then the travel time is defined as $t_t = t_a - t_s$. Every time a bundle is sent, the current time (t_s) is associated with it, allowing the destinations to calculate t_t.

Figure 7 shows the travel time as calculated at one of the sinks (sink # 0 onboard plane # 0), using 64 Kbps and 128 Kbps wireless links. It is clear that 64 Kbps is insufficient to handle the traffic required during EMPR simulation. As seen, during the interval from second 2000 to second 2400, many of the PDUs took almost 40 seconds to arrive at their destinations, which is completely unacceptable for maintaining the required fidelity during the training simulation.

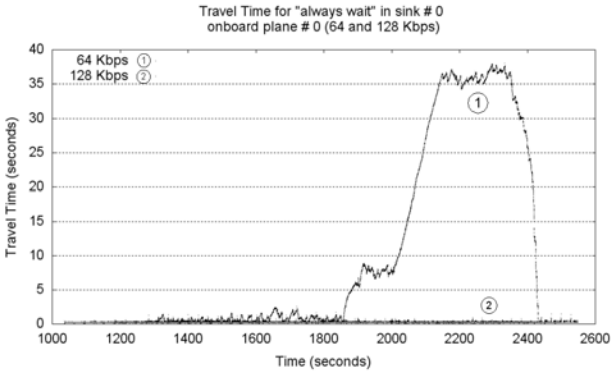

Figure 7. Travel time for the *Always-Wait* strategy, at destination 0 onboard plane 0, using 64 and 128 Kbps wireless links

However, a substantial improvement is obtained at 128 Kbps where the latency drops closer to 1 second. Figure 8 zooms in on the data shown in Figure 7, eliminating the 64 Kbps curve from the graph. It shows that most of PDUs take less than 0.4 seconds to reach their destinations. It is interesting to note the large concentration of PDUs near 0.25 seconds. This is because any message sent from the ground station to an airplane via satellite takes at least that duration. The distance traveled by the signals at the speed of light is near 76600 Km (2·[35800 Km of satellite height + 2500 Km of ground displacement]), giving a propagation delay of approximately 0.255 seconds. The graph also shows that some PDUs take less than 0.1 seconds of travel time. Those PDUs correspond to messages sent between simulation stations onboard en-route planes without passing through the satellite.

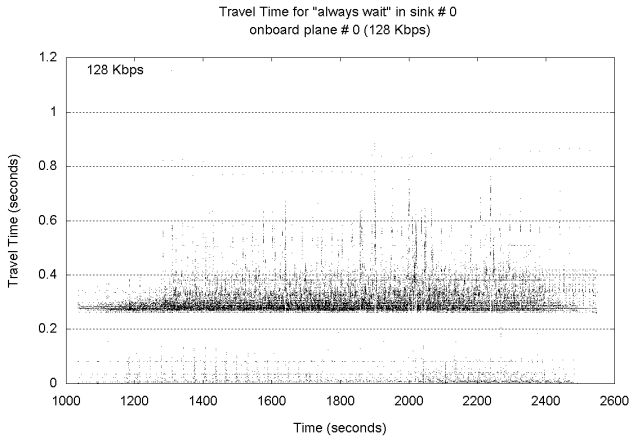

Figure 8. Close-up of travel time at sink 0, plane 0 (128 Kbps)

Table 6 shows the average and standard deviation of the travel time for each combination of algorithm and bandwidth, measured at sink #0 onboard plane #0. Considering that approximately 83 % of the PDU traffic received at sink #0 is transmitted by the ground station via satellite, and that for those PDUs 0.255 seconds is an unavoidable delay, the table shows, the table shows a very good behavior of the algorithms at 256 Kbps or more, giving a slight advantage to *Always-Wait* bundling.

Table 6. Average and standard deviation of travel time measured at sink #0

Avg: **Std. Deviation:**	**64 Kbps**	**128 Kbps**	**256 Kbps**	**512 Kbps**
Type	9.20 13.2	0.304 0.099	0.262 0.069	0.249 0.064
Type-Length	9.24 13.2	0.306 0.101	0.262 0.069	0.249 0.064
Always-Wait	9.43 13.5	0.303 0.099	0.261 0.069	0.249 0.064
Neural Net.	28.7 33.2	0.314 0.119	0.261 0.069	0.248 0.064
Always-Send	64.0 58.0	0.333 0.153	0.263 0.062	0.251 0.057

Queue Length Analysis

The satellite and the routers each contain a message queue to store incoming PDUs that are pending service. Every time a PDU arrives at a router or satellite, the number of other messages in the system is counted, including the PDUs already in the queue, plus any one being serviced. The value of this counter is recorded in an OMNeT statistics file along with the arrival time of the incoming PDU. When the simulation ends, a post-processing program reads the file to obtain the statistics on queue length distribution.

Due to the nature of the PDU traffic in the simulation, two queues to focus attention on are the router queue onboard any aircraft, for instance airplane #0, and the satellite queue. Figure 9 shows the satellite queue at 64 Kbps and 128 Kbps.

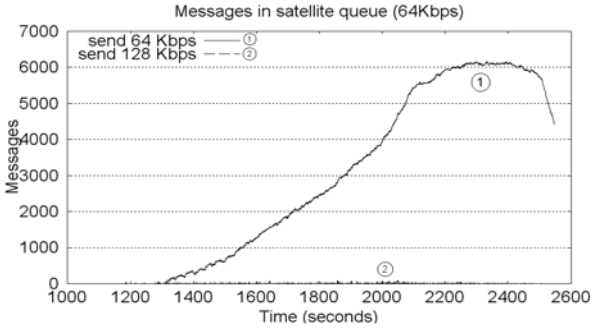

Figure 9. Messages in satellite at 64 and 128 Kbps showing the impact of a higher bandwidth on queue length

It is clear from the graph that 64 Kbps is an insufficient bandwidth from the perspective of queue length as well, causing the satellite queue to grow unbounded. The reason for its reaching a maximum of about 6000 messages, followed by a descent when the simulation is approaching its end, is that at that time no additional messages are sent from the generators. At 128 Kbps, a significant change in the queue length is observed, with it maintaining reasonably low values. Another observation is that the graph does not reach the zero axis. This occurs because the queue status is reported only if another message enters the queue. After the arrival of the last message to the queue, the messages already stored there are consumed without being reported.

Table 7 lists a summary of the average and standard deviation of the satellite queue length obtained for combinations of different algorithms and bandwidths.

Table 7. Average and standard deviation in the satellite queue length for combinations of algorithm and bandwidth

Avg: Std. Deviation:	64 Kbps	128 Kbps	256 Kbps	512 Kbps
Type	316.97 411.43	2.38 3.97	0.91 1.72	0.56 1.23
Type-Length	318.154 412.273	2.44 4.13	0.92 1.75	0.56 1.26
Always-Wait	327.278 421.161	2.30 3.88	0.85 1.69	0.49 1.16
Neural Net.	1028.47 1045.26	3.58 6.37	1.24 2.18	0.79 1.52
Always-Send	2962.94 2236.83	5.40 10.78	1.22 2.55	0.63 1.57

Collision Analysis

The satellite and routers keep separate counters of collisions on each of the links that they are connected to. The satellite is connected to two wireless links (WSP and WGS in Figure 2) and the routers are connected to one LAN and two wireless links (LAN, WSP and WPP in Figure 3). Each time a collision is detected, the corresponding counter is incremented and its new value along with the current simulation time is recorded for future processing. Again, *Always-Wait* performed well, closely followed by *Type* and *Type-Length* prediction bundling strategies.

The simulation results shown in Figure 10 indicate that at 64 Kbps the highest collision rate measured at the router aboard airplane #7 was close to 12 collisions per second and it occurred in WSP, the link connecting the satellite to the planes, during the time interval [2050, 2100]. At 64 Kbps, fewer than 4800 collisions were detected in total for the *Always-Send* algorithm, which represents less than 8% of the total number of PDUs. On the other hand, at 256 Kbps the total number of collisions for the *Always-Wait* algorithm was close to 2100, or 5.3 % of all transmissions. Collision accumulation in plane #7 at different bandwidth rates is given in Figure 10.

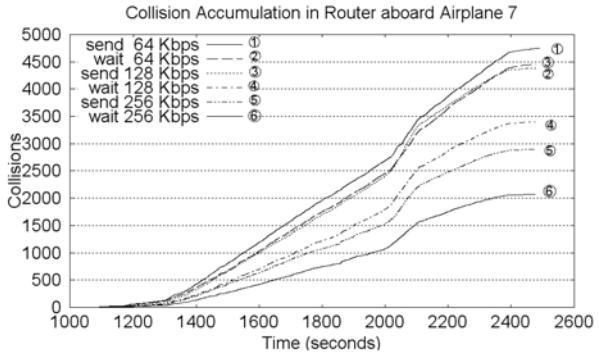

Figure 10. Collision accumulation as counted in router at airplane 7 for combinations of the *Always-Send* and *Always-Wait* algorithms at 64 and 128 Kbps

As Figure 10 shows, at 128 Kbps and 256 Kbps there is roughly a total difference of 1000 fewer collisions for the *Always-Wait* than for the *Always-Send* algorithm. As the bandwidth increases, the number of collisions decreases because at higher bandwidths the packets require less transmission time, and so the corresponding probability of a collision is reduced.

CONCLUSIONS

There are three sets of conclusions that can be drawn from the communication modeling study of DIS traffic under EMPR. The first set corresponds to observations about the required bandwidth in the wireless channels required to carry out an OTB simulation training exercise. The second set corresponds to conclusions about the effectiveness of the bundling techniques. The third set corresponds to research extensions not fully developed herein and thus potential future work.

Required Bandwidth

One goal of this research was to estimate the required bandwidth needed by wireless channels for communication

between the OTB stations in the JEMPRS flying network depicted in Figure 1. A vignette was prepared and executed in OTB, generating the traffic transmitted between the 24 flying sites plus the CONUS ground station. The PDU traffic generated was captured and used as input to the OMNeT discrete event simulator, preserving the original PDU timestamps, types, and lengths. The OMNeT model was executed under several combinations of wireless bandwidths (64K, 128K, 256K, 512K) and the bundling techniques developed (*Always-Send, Neural Network, Type, Type-Length, Always-Wait*).

From the results of the OMNeT traffic model, it can be concluded that 64 Kbps wireless links are not sufficient to handle the required PDU traffic for an EMPR vignette, due to the large negative slack reported in the generators (Figure 5), the large travel time latency for transmissions involving the ground station (Figure 7), and the excessive satellite queue length (Figure 9). However, at 128 Kbps the situation improves dramatically. It seems that an average of 0.26 seconds in the travel time and 2.3 messages in the satellite queue are good indicators of performance. However, the negative average slack time of –0.017 seconds indicates that 128 Kbps represents a slightly insufficient bandwidth to complete handle some traffic bursts during EMPR scenarios. Therefore, the conclusion is that to not incur unnecessary delays then the required bandwidth should be at least 256 Kbps in the JEMPRS network for this type of training exercise.

Effectiveness of Bundling

All of the statistics presented indicate that bundling can be effective for reducing PDU traffic and improving utilization of available bandwidth. The reductions in negative slack (Figures 5 and 6), travel time (Table 6), satellite queue length (Table 7), and number of collisions (Figure 10) are all indicators in that sense. Table 8 shows the total number of bundles and total number of bytes transmitted by the *Always-Wait* strategy at different bandwidths. In all cases, the number of PDUs read from the summary files is 60,341 PDUs equivalent to 12,415,774 bytes, which corresponds to the non-bundling strategy. Therefore, Always-Wait presents an approximate reduction of 35 % in the number of PDUs transmitted and 21 % in the number of bytes transmitted.

The replication of PDUs through bundling presented in this research differs from other proposals [ATO 1995, Taylor 1995, Taylor 1996a, Taylor 1996b, Fullford 1996, Bassiouni et al. 1997, Purdy and Wuerfel 1998, Wills et al. 2001] in several ways. Bundling strategies used here take into account the internal structure of each PDU; only PDUs of the same type and length are combined together into a bundle. The proposed bundling algorithms are straightforward to implement, as well as the extraction of individual PDUs at the destination. Bundles are independent of each other so the information required to extract the PDUs is contained in the same bundle. Thus, during packet switched transmission the bundles can arrive out-of-order without impacting the replication strategy at the receiver.

Table 8. Number of bundles and number of bytes transmitted by Always-Wait at different bandwidths

Kbps	# of Bundles	Number bytes
64	38,708	9,778,978
128	39,319	9,837,114
256	39,554	9,860,433
512	39,619	9,866,762

Future Work

Several possibilities for future work are open. The bundling algorithm could be extended to consider PDUs of different length as candidates to be included in the bundle. Also, the bundling of PDUs of different types might be investigated as long as the resulting bundle still maintains the underlying structure, or other extensions to publish-subscribe protocols similar to HLA.

Another branch of research could be based on further improvement of the prediction of the type for upcoming PDUs. The neural network developed in this study was rudimentary, having prediction accuracy close to 70 %. If an improved neural network is used, it could outperform the *Always-Wait* strategy, which is not necessarily optimal. However, a comparison of the CPU overhead incurred by both algorithms would be appropriate to justify if the use of improved prediction is warranted given the benefits.

ACKNOWLEDGEMENTS

The authors would like to thank the U.S. Army Program Executive Office for Simulation, Training, & Instrumentation (PEO STRI) for their support of this research.

REFERENCES

[Bahr and DeMara 96] Bahr, H. A. and DeMara, R. F., "A Concurrent Model Approach to Reduced Communication in Distributed Simulation," in *Proceedings of 15th Annual Workshop on Distributed Interactive Simulation*, Orlando, FL, U.S.A., Sept., 1996.

[Bassiouni et al. 1997] Bassiouni M., Chiu M., Loper M., Garnsey M., and Williams J. "Performance and Reliability Analysis of Relevance Filtering for Scalable Distributed Interactive Simulation," *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, Vol. 7, No. 3, pp 293–331, July 1997.

[Ceranowicz 2002] Ceranowicz A., Torpey M., Helfinstine B., Evans J., and Hines J. "Reflections on Building The Joint Experimental Federation", in *Proceedings of the 2002 I/ITSEC Conference*, Orlando, FL, U.S.A., Dec., 2002.

[Frontlines 2002] Microsoft Frontlines "JBC Initiative Delivers High-Bandwidth Collaboration Tools to Austere

Locations", Summer 2002. Available from http://www.larstan.net/Published_work/PDF_Q302_WP/Frontlines2%200702.pdf

[Fullford 1996] Fullford D. "Distributed Interactive Simulation: It's Past, Present, and Future." In *Proceedings of the 1996 Winter Simulation Conference*, Coronado, CA, U.S.A., Dec. 8 - 11, 1996. Available from http://portal.acm.org/ft_gateway.cfm?id=256601&type=pdf&coll=GUIDE&dl=ACM&CFID=17089419&CFTOKEN=3244421

[Henninger 01] A. E. Henninger, A. J. Gonzalez, M. Georgiopoulos, and R. F. DeMara, "Human Performance Models for Embedded Training: A Novel Approach to Entity State Synchronization," in *Proceedings of the '01 Advanced Simulation Technology Conference - Military, Government, and Aerospace Conference (ASTC-MGA),* Seattle, WA, U.S.A., April 22 - 26, 2001.

[IEEE/ANSI 1985] IEEE/ANSI STANDARD 8802/3. *Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specification*. IEEE Computer Society Press, 1985.

[IEEE/ANSI 1999] IEEE/ANSI STANDARD 8802-11. *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications*. IEEE Computer Society Press, 1999.

[Purdy and Wuerfel 1998] Purdy S., and Wuerfel R. "Comparison of HLA and DIS Real-Time Performance," In *Proceedings of 1998 SPRING SIW Conference*, Orlando, FL, U.S.A., March 9-13, 1998. Available from http://www.sisostds.org/doclib/doclib.cfm?SISO_FID_975

[Srinivasan 1996] Srinivasan S. "Efficient Data Consistency in HLA/DIS++," In *Proceedings of the 1996 Winter Simulation Conference*, Coronado, CA, U.S.A., Dec. 8 - 11, 1996., pp 946 – 951.

[Taylor 1995] Taylor, D. "DIS-Lite & Query Protocol," in *Proceedings of the 13th DIS Workshop on Standards for the Interoperability of Distributed Simulations,* Orlando, FL, U.S.A., Sept., 1995.

[Taylor 1996a] Taylor, D. "The VR-Protccol" in *Proceedings of the 14th DIS Workshop on Standards for the Interoperability of Distributed Simulations,* Orlando, FL, U.S.A., March, 1996.

[Taylor 1996b] Taylor, Darrin. 1996. "DIS-Lite and Query Protocol: Message Structures." in *Proceedings of the 14th DIS Workshop on Standards for the Interoperability of Distributed Simulations*, Orlando, FL, U.S.A., March, 1996.

[OTA 1995] Office of Technology Assessment - U.S. Congress, "Distributed Interactive Simulation of Combat," OTA-BP-ISS-151, Washington, DC: U.S. Government Printing Office, September 1995. Available at http://www.wws.princeton.edu/cgi-bin/byteserv.prl/~ota/disk1/1995/9512/9512.PDF

[Varga 2003] Varga, A. "OMNeT++" In the column *"Software Tools for Networking", IEEE Network Interactive*, Vol. 16, No. 4, July 2002.

[Vargas et al. 2004] Vargas J., DeMara R., Gonzalez A., and Georgiopoulos M. 2004 "Bandwidth Analysis of a simulated Computer Network Running OTB." In *Proceedings of the Second Swedish-American Workshop on Modeling and Simulation (SAWMAS 2004),* Cocoa Beach, FL, February, 2004.

[Wills et al. 2001] Wills C., Mikhailov M., and Shang H. "N for the Price of 1: Bundling Web Objects for More Efficient Content Delivery." in *Proceedings of the Tenth International Conference on World Wide Web*, ISBN 1-58113-348-0, Hong Kong, Hong Kong, 2001, pp 257-265.

[Witman 2001] Witman, R. and Harrison, C., "OneSAF: A Product Line Appraoch to Simulation," Technical Report, Contract Number DAAB07-01-C-C201. The Mitre Corporation, 2001.

BIOGRAPHY

Juan J. Vargas was born in Costa Rica in 1956. He has a BS (1977) in Computer Science and a BS (1988) in Mathematics from the University of Costa Rica (UCR), and an MS (1991) from the University of Delaware. Currently, he is pursuing Ph.D. studies in Computer Engineering at the University of Central Florida while employed as a visiting lecturer there. His research interests include parallel and distributed processing, computer networks and computer architecture. He has served as Chair and Program Coordinator of "Telematics," and MS program at UCR, and Vice Chair in the School of Computer Science at UCR.

Ronald F. DeMara is an Associate Professor in the Department of Electrical and Computer Engineering at the University of Central Florida. He earned a BS (1987) in Electrical Engineering from Lehigh University, MS (1989) in Electrical Engineering from the University of Maryland, and Ph.D. (1992) in Computer Engineering from the University of Southern California. Prior to joining UCF, he was an Associate Engineer at IBM Corporation in Manassas, Virginia. His research interests include design and performance analysis of Computer Architecture and Distributed Systems. He has published over 70 papers in these areas. He is a registered Professional Engineer in California, member of IEEE, ACM, and ASEE.

Michael Georgiopoulos received a diploma in Electrical Engineering from the National Technical University in

Athens in 1981. He also received his M.S. and Ph.D. degrees in Electrical Engineering from the University of Connecticut, Storrs, CT, in 1983 and 1986, respectively. In 1987, he joined the University of Central Florida, where he is currently a professor in the Department of Electrical and Computer Engineering. Dr. Georgiopoulos has been conducting research in the area of neural networks and applications for over 10 years now, and he has published 41 journal papers and over 120 conference papers/book chapters.

Avelino J. Gonzalez received his bachelors and masters degree in electrical engineering from the University of Miami in 1973 and 1974 respectively. He received his doctoral degree from the University of Pittsburgh in electrical engineering in 1979. He had held various administrative and technical positions in Westinghouse Electric Corp., until 1986 after which he joined the University of Central Florida where he is now a professor in the Department of Electrical and Computer Engineering. Dr. Gonzalez holds three patents and has co-authored two books and has published numerous articles in technical journals and conferences on artificial intelligence, context based behavior and representation, temporal reasoning intelligent diagnostics and expert systems.

Henry Marshall is the Principal Investigator for Mounted Embedded Simulation Technology at the Research, Development and Engineering Command (RDECOM) Simulation and Training Technology Center (STTC). Prior to this assignment he worked at the Simulation, Training and Instrumentation Command (STRICOM) where he spent 11 years as lead for the CGF/SAF, HLA and Linux Porting developments on the Close Combat Tactical Trainer (CCTT) system in addition to being a OneSAF team member. His twenty years with the Government have been mainly in CGF and Software acquisition. He received a BSE in Electrical Engineering and an MS in Systems Simulation from the University of Central Florida.