Simulation Models Verification for Resilient Communication on a Highly Adaptive Energy-Efficient Computer

Stefan Pfennig
Technische Universität Dresden
Faculty of Computer Science
01069 Dresden, Germany
stefan.pfennig@tu-dresden.de

Kim Feldhoff
Technische Universität Dresden
ZIH
01069 Dresden, Germany
kim.feldhoff@tu-dresden.de

Florina M. Ciorba
University of Basel
Department of Mathematics and Computer Science
4051 Basel, Switzerland
florina.ciorba@unibas.ch

Elke Franz, Tobias Reiher
Technische Universität Dresden
Faculty of Computer Science
01069 Dresden, Germany
{elke.franz, tobias.reiher}@tu-dresden.de

Mario Bielert, Thomas Ilsche, Wolfgang E. Nagel
Technische Universität Dresden
ZIH
01069 Dresden, Germany
{mario.bielert, thomas.ilsche, wolfgang.nagel}@tu-dresden.de

ABSTRACT

The utility of simulations depends on the confidence in the simulation implementation and its results. This study discusses the verification of the communication models in the parallel trace-driven simulation framework HAEC-SIM. As simulation input, a parallel application is executed and recorded on an existing HPC system. The simulation focuses on modeling the transfer times of point-to-point messages within the application and the indirect effects resulting in an output trace of application events of the simulated execution on a target platform. Consequently, via verification the message transfer times obtained with HAEC-SIM are compared with those of an independent implementation of the communication models. Both implementations consider the number of hops, the size and the target system parameters for each message. During verification the following factors are varied: application benchmarks, network topologies, mapping strategies, and resilient communication models.

Verification yields an almost perfect agreement: the transfer times differ for a tiny percentage (0.00019 %) of messages by a negligible deviation of one picosecond, which is the finest granularity of the time data type. This result strengthens the confidence in a correct implementation of the communication models in simulation.

Author Keywords

verification; resilient communication; network coding; trace-driven simulation; HAEC-SIM

ACM Classification Keywords

D.2 Software Engineering; D.2.4 Software/Program Verification; I.6 Simulation and Modeling; I.6.6 Simulation Output Analysis; I.6.8 Types of Simulation: Discrete Event, Parallel

1. INTRODUCTION

Motivation

A novel concept, namely the HAEC Box [4, 7], utilizes innovative ideas of optical and wireless chip-to-chip communication to allow high runtime adaptivity with the goal of achieving high performance and energy efficiency. HAEC-SIM[1] [3] is an integrated simulation environment designed for the study of the performance and energy costs of the HAEC Box running energy-aware applications.

Given the characteristics of the HAEC Box, any simulation of the execution of communication intensive benchmarks on this platform needs to account for packet loss due to errors, failures, or malicious attacks. Therefore, HAEC-SIM provides three communication models: resilient dimension order routing (DOR_r), resilient practical network coding (PNC_r), and resilient dynamic network coding (NCD_r) to support resilient communication within the HAEC Box.

Goal

Verification is an integral activity in the modeling and simulation process. The goal of this work is the practical verification of the code implementation of three resilient communication models [16] within HAEC-SIM.

Approach

Independent verification and validation phases consist of: requirements verification, design verification, code verification, and validation [13, p.463]. Our approach to achieving the above goal corresponds to code verification. We perform code verification by comparing the simulated point-to-point (unicast) transfer times produced by HAEC-SIM using the three resilient communication models with the individual message transfer times obtained from an independent implementation of these models. Via this approach, we were able to identify errors arising from a loss of precision due to type conversion and from insufficient precision using a certain data type.

Related work

Most of the related work on off-line (or trace-driven) simulators report validation studies [12, 18, 19, 20, 21] with only

[1] `https://tu-dresden.de/zih/haec_sim`

SpringSim–HPC 2016 April 3-6 Pasadena, CA, USA

very few describing the verification of their simulation models thoroughly [9]. Even though validation is not the focus of the present work, we briefly overview the most relevant validation results.

The network simulation component of BigSim has been validated in [19] for the PERCS network using MERCURY. For a small ping-pong test, MERCURYs results differed by 0.6 % to 1.1 % from BigSim's. For another all-to-all test, the results between the two simulators only differed by 0.5 %. In another effort, BigSim has been validated against a Power 775 prototype drawer using an all-to-all test. The BigSim simulation results were smaller and within nearly 10 % of those observed on the actual drawer.

SST/macro [21] has been validated against Hopper, a Cray XE6 with the Gemini interconnect [1]. Validation tests focused on communication-bound modeling using collective operations. The authors conclude that SST/macro correctly reproduces the performance trends and even the correct congestion behavior.

PSiNS [18] includes several built-in communication models (namely simple, resource contention, and PMaC) that can be used to investigate a target system. Each event can use a different communication model. PSiNS uses direct execution for MPI calls. It has been validated by comparing the simulated communication times for each task of a certain benchmark (HYCOM with 124 processors) against the measured times. Even though the simple built-in communication model was used, the average absolute error in predicting the communication times for all tasks was 17 %, whereas the error in predicting the total communication time was 14 %.

The network model of DIMEMAS [12] assumes two-level buses. A linear performance model is used for communication, but some non-linear effects such as network conflicts are taken into account. A qualitative validation of Dimemas[2] shows that the proposed model is simple enough to capture the performance of communication primitives acceptably well when compared to real executions of NAS parallel benchmarks. In the companion quantitative validation study it was found that most of the benchmarks are predicted by Dimemas with less than 10 % of errors.

SIMGRID [5] is designed as an on-line simulator with various bindings (including MPI), with a communication model based on flows which also assumes congestion. The flow-based network model has been validated for message sizes of 10 MB and 100 MB, respectively [20]. The authors state that having a more precise knowledge from the users of the acceptable range of platform parameters would enable to have much more reliable simulations and could also provide confidence intervals on the simulation results.

LogGOPSim [9] implements several LogP-derived network models. LogGOPSim has been successfully verified against analytical LogGPS models for different communication patterns (linear broadcast/scatter, linear gather, binomial tree, and dissemination). The simulator has also been validated against two real-world applications, which it represents well despite restrictions in its modeling (no congestion). The average simulation error was found to be below 2 % in all validation experiments.

Given that our modeling and simulation workflow is very specific to the HAEC Box characteristics, validation against existing similar platforms is not necessarily straightforward. Verification, however, is the prerequisite step to any successful validation endeavor [13, p.463]. Thus, in this work, we concentrate on verification of the communication models. Similar to the above related work on validation, we consider realistic applications and platforms. Moreover, we not only verify a small subset of our test cases output, but the entire output of all simulations conducted for the chosen test cases. We also emphasize the significance and benefits of high performance computing for conducting the simulations as well as their analysis and verification. Extending own prior work [4], in this paper we consider a communication-aware mapping strategy (namely 'minimum Manhattan distance'), an additional target system topology (3D mesh), and perform (and document) a more careful selection of model and simulation parameter values.

The remainder of this paper is organized as follows. The parallel applications of interest are discussed in Section 2 while the system targeted in the simulations is described in Section 3. A brief overview of the resilient communication models at the focus of our verification is given in Section 4. The verification workflow is presented in Section 5 together with the details of individual verification steps and results. Section 6 concludes the paper and outlines steps for future work.

2. APPLICATIONS AND THEIR EXECUTION TRACES

The code verification process (detailed in Section 5) requires that realistic simulations be performed. Thus, we chose the LU.D and BT.D applications of the NAS Parallel Benchmark Suite 3.3 [2] as example applications. These applications have been executed with a total of 4096 MPI processes on 256 Taurus[3] nodes, equally distributed as 16 processes per node. From the traces of the two communication intensive benchmarks (denoted as LU.D.4096 and BT.D.4096), we observed that they spend ≈68 % and ≈48 % respectively of the execution time in MPI functions.

Detailed running times and the percentage of time spent in the application and for the communication are shown in Table 1. In both cases, the communication matrices of both benchmarks are not dispersed. This means, that the processes communicate primarily with their neighboring ranks. The distribution of message sizes differs between the two benchmarks. In the LU.D.4096 trace, about 99 % of the total number of exchanged messages are of size 240 B and 280 B. In the BT.D.4096 trace, most exchanged messages are of size 11760 B and 1960 B, while other larger messages of size around 200 KiB represent at most 1 % of the number of exchanged point-to-point messages.

[2] http://www.bsc.es/computer-sciences/performance-tools/dimemas/validation

[3] https://doc.zih.tu-dresden.de/hpc-wiki/bin/view/Compendium/HardwareTaurus

Benchmark	Total Run Time (s)	Exclusive Application (%)	Exclusive MPI (%)
LU.D.4096	19.62	32.40	67.60
BT.D.4096	24.71	52.07	47.93

Table 1: Benchmark execution times and percentage of exclusive accumulated times for Application and MPI functions on Taurus.

3. SIMULATED SYSTEM SPECIFICATIONS

HAEC-SIM allows to configure the target platform topology and link specification. Previous simulations were focussed on the future platform specification described in [3]. However, in this work, parameters that characterize networks in current large-scale high performance computing (HPC) systems are used. Hence, we are combining considerations of the HAEC Box and of current HPC systems. This combination allows for evaluations of the future platform that are better comparable to existing HPC systems. The comparison of the HAEC Box to current HPC systems is reasonable since, in fact, the HAEC Box represents a small scale HPC environment. Conversely, not all assumptions for HPC systems hold true for the HAEC Box, what especially results in different assumptions regarding the packet loss probabilities (Section 4.1).

The target topology and interconnect are largely based on the Gemini interconnect [1]. Our target platform consists of $16 \times 16 \times 16$ computing nodes connected in a 3D mesh and a 3D torus. Each link is assumed to have bandwidth of 9.375 GB/s, which corresponds to the maximum bandwidth for Gemini. For the link latency, we assume 1.27 μs based on the Gemini interconnect latency [14]. The computing power of the target platforms is assumed equivalent to that of the execution platform where the application trace was generated.

4. COMMUNICATION MODELS

4.1 Unicast (Point-to-Point) Communication

In this work, we focus on unicast (point-to-point) communication where one sender transmits data over one or more forwarders to one receiver. The exception is an internode communication that takes place only locally. Further, we assume single-path communication where only one path from the sender to the receiver is used during the transmission of a message. Both, the mapping of processes to nodes and the applied routing strategy determine the number of physical links or hops (h) a message has to travel.

Since we assume that packet loss can occur due to errors, failures, or intended attacks, the sender needs feedback whether the transmission was successful. Therefore, the receiver sends back end-to-end acknowledgments.

A message that needs to be transmitted is issued at the MPI layer. Then, it has to be passed to the transport layer. We account for this task via an injection latency of $d_{mpi} = 500$ ns with unlimited injection bandwidth. At the transport layer, the message of length m is split into $n_p = \frac{m}{L_p}$ data packets of equal length ($s_p = 1500$ B). The payload per packet ($L_p < s_p$) depends on the communication model and is discussed in the next subsection.

We assume a packet loss probability per link of $p = 0.01$. The packet loss probability for a block of $n = s_p * 8$ bit/B $= 12,000$ bits can be computed from the bit error rate (BER) of the physical channel by $p = 1 - (1 - \text{BER})^n$. Hence, the assumed packet loss probability corresponds to a bit error rate of $\approx 10^{-6}$.

This packet loss probability value has been selected due to the fact that we also aim to achieve security against intentional attacks. In HPC systems, security issues are usually neglected since the system is assumed to be physically hosted in a secure environment. However, this assumption cannot be guaranteed for the HAEC Box due to its smaller size and portability. One cannot exclude attacks without the guarantee that attackers can never access the system – and this will not be possible. Particularly, one of the target uses of the HAEC Box is the efficient execution of big parallel applications. Hence, there must be a possibility for users to place their applications on the nodes of the HAEC Box and to execute them. Since it is not possible to decide a priori whether a candidate application is malicious or not, security aspects are important and must be addressed. Selecting a packet loss probability that is higher than usually assumed for HPC systems allows to evaluate how the system works even in case of the assumed attacks. If there is no attacker, the system's performance will be of course even better.

For simplicity, we consider a uniform packet loss probability for all links. This assumption must be refined in future work. In this work, it allows to evaluate whether a successful data transmission is still possible under a relatively high loss probability when resilient communication schemes are employed.

4.2 Resilient Communication Models

The resilient communication models we use within our evaluation ensure that a message can be transmitted even in case of packet loss. In order to prevent unrecognized modifications, forwarders and receivers can verify the validity of data packets and acknowledgments by means of digital signatures and discard manipulated packets. We assume the use of ECDSA [10] for the computation of digital signatures of size $s_{sig} = 32$ B. Packets are sent according to the following three communication models:

DOR$_\mathbf{r}$: The first model denotes the common store-and-forward approach of sending packets. Similar to TCP, packets are organized in windows of size s_w. The receiver confirms the successful receipt of each packet by means of acknowledgments. Upon successfully sending one window, the sender starts to send packets of the next window. Each packet contains a window identifier of size s_{wid}, a packet identifier of size s_{pid}, and a digital signature. Hence, the resulting payload per packet is given by $L_p = s_p - s_{wid} - s_{pid} - s_{sig}$. An acknowledgment of size s_a contains the window identifier, the packet identifier, a timestamp of size s_{tst}, and the digital signature.

PNC$_\mathbf{r}$: The second model refers to Practical Network Coding [6]. Packets comprise a number of elements of size

s_{ff} from a finite field to allow for the linear combinations. The sender organizes packets in matrices called generations (of size s_w), amends these packets by a global encoding vector of s_w symbols of size s_{ff}, and computes random linear combinations out of the packets of one generation. Each packet additionally contains a generation identifier of size s_{gid} and a digital signature of size s_{sig}. The resulting payload of a data packet according to the second model is $L_p = s_p - s_{gid} - s_w \cdot s_{ff} - s_{sig}$. An acknowledgment of size s_a contains the generation identifier, the rank of the matrix of received packets, a timestamp, and the digital signature. The receiver decodes upon receiving sufficient linear independent combined packets. It sends acknowledgments to confirm the current rank of the matrix of received packets. The sender proceeds to the next generation upon completing the sending of the current one.

NCD$_r$: The third model is similar to PNC$_r$. However, the sender estimates the delivery probability by means of the receiver acknowledgments. Thus, the sender can estimate the number of remaining packets to be sent such that the receiver will have full rank [15].

4.3 Model Parameters and Values

The resilient communication models were implemented using Sage [17] according to the models introduced in [16]. These models also comprise delays for sending a packet (d_s), receiving a packet (d_r), processing a packet at an intermediate node (d_i), and processing an acknowledgment (d_a). The delay for sending a packet over one hop is given by $d_{h,p} = d_{out} + l + \frac{s_p \cdot 8}{b} + d_{in}$, where l refers to the link latency, and b to the bandwidth. The delay for sending an acknowledgment over one hop is computed accordingly for s_a instead of s_p. The delays d_{in} and d_{out} describe the delays for sending data over the physical channel or for receiving data sent over the physical channel. Finally, we also have to include delays for the computation and verification of digital signatures (d_{sign} and d_{verify}), these delays are incorporated in d_s, d_r, d_i, and d_a.

The values for those processing delays are set in [8, Table 2] to achieve a certain targeted latency. Thus, we scaled up those parameters, as well as other parameters that refer to bandwidth, to comply with the Gemini link characteristics. The parameters used in this work are summarized in Table 2 herein.

Parameter	DOR$_r$	PNC$_r$	NCD$_r$
s_w	5	5	5
s_{wid}	4 B	—	—
s_{pid}	1 B	—	—
s_{gid}	—	4 B	4 B
s_{tst}	4 B	4 B	4 B
s_{sig}	32 B	32 B	32 B
s_{ff}	—	1 B	1 B
s_p	1500 B	1500 B	1500 B
s_a	41 B	41 B	41 B
L_p	1463 B	1459 B	1459 B
d_{mpi}	500 ns	500 ns	500 ns
$d_{out} = d_{in}$	127 ns	127 ns	127 ns
$d_{h,p}$	1.544 μs	1.544 μs	1.544 μs
$d_{h,a}$	1.525 μs	1.525 μs	1.525 μs
d_{sign}	40 ns	40 ns	40 ns
d_{verify}	100 ns	100 ns	100 ns
$d_a = d_i$	163.5 ns	163.5 ns	163.5 ns
d_s	294 ns	294.5 ns	294.5 ns
d_r	394 ns	396.5 ns	396.5 ns

Table 2: Resilient communication model parameters.

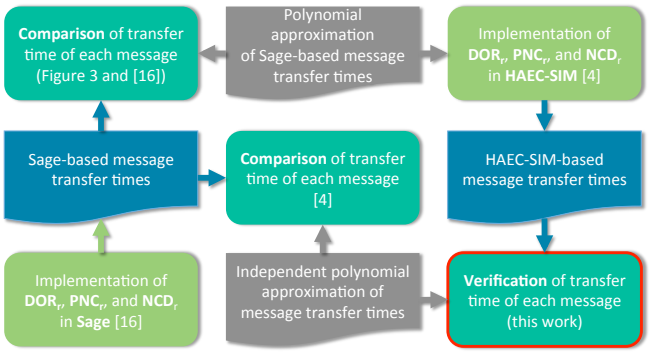


Figure 1: Workflow for the code verification of the implementation of resilient communication models in HAEC-SIM. The light green boxes denote the two distinct implementations of the resilient communication models. The darker green boxes denote the comparisons conducted in prior and present work between the message transfer times produced by the two model implementations.

5. VERIFICATION

5.1 Verification Workflow

The process of code verification of the implementation of the resilient network models in HAEC-SIM follows the workflow illustrated in Figure 1.

The first step was the implementation of the DOR$_r$, PNC$_r$, and NCD$_r$ communication models using Sage [16]. Polynomial approximations, described in Section 5.2, have been used to enable the use of the transfer times provided by these models in HAEC-SIM [4]. Subsequently, HAEC-SIM writes the transfer time of each message encountered in the input trace and the number of hops it traveled during simulation in the simulated output trace. This data is extracted with the help of a parallel parsing tool and used by another tool to verify in parallel each simulated message data against an independent implementation (last step of the illustrated workflow). Both HAEC-SIM and the independent implementation use polynomial approximation for approximating the message transfer times obtained from Sage-based simulations [16], as shown in Figure 1. The acceptability of the verification results is assessed in Section 5.5. It is important to note that the independent polynomial approximation focuses exclusively on the message transfer times, while in addition, the HAEC-SIM-based implementation of the models addresses and interleaves the exchanged messages with computational

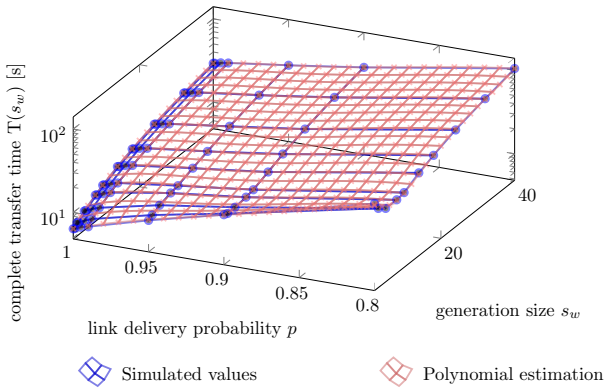


Figure 2: Comparison of simulated values to polynomial approximation for 6 hops.

aspects of the application. Thus, a direct comparison and verification between the two implementations is only possible at the message level.

During the verification process we were able to identify: (a) errors arising from a loss of precision due to type conversion and (b) insufficient precision in chosen data type. (a) *Precision loss*: The timestamps of events read from the input trace are internally represented in the simulator as integers. The message transfer times are calculated using floating point arithmetic. This leads to inevitable conversions from integers to floating point numbers and vice versa. Both types of conversions can, however, lead to loss of precision. Therefore, care must be taken when and where these conversions are performed, especially whenever arithmetics are involved, e.g., int(1.5) + int(1.5) equals 1 + 1 and the expression returns 2, while int(1.5 + 1.5) equals int(3.0) and therefore the expression returns 3. The aim is to minimize the amount of possible type conversion errors. (b) *Insufficient precision*: Certain modeling parameters (i.e., d_s, d_i, d_r, d_a, d_{h_p}, and d_{h_a}) corresponding to delays were represented as integers. The values of these delays are comparatively small and in some cases also obtained from fractions. Thus, a representation as integer results in insufficient precision. These delays are now represented as floating point numbers.

5.2 Polynomial Approximation of Communication Times

To provide meaningful communication times, especially the resilient models needs to be simulated multiple times. Just using one simulation run would yield to indeterministic or questionable times. However, this repeated simulation would take to much time in HAEC-SIM and thus, we do this prior to that in Sage [17]. The results of the low level point-to-point communication simulation are integrated into HAEC-SIM by means of a polynomial approximation. This has several advantages. We can derive the end-to-end communication times for arbitrary values of the packet loss probability, the generation size, and the number of hops even if the real simulation was done only for a limited number of parameter sets. Further, we save memory and computing time, since we do not store all results of the measurements, but only the polynomials and the times are very fast computable.

Msg. size	DOR_r	PNC_r	NCD_r
Mapping: xyz			
$\leq$ 200 B	36.68 μs	37.10 μs	36.17 μs
240 B	36.51 μs	36.95 μs	35.89 μs
280 B	32.91 μs	33.30 μs	32.38 μs
3,264 B	12.70 μs	11.90 μs	11.76 μs
$\approx$204,800 B	1180.68 μs	1083.49 μs	1068.08 μs
Mapping: min md			
$\leq$ 200 B	41.38 μs	56.64 μs	43.62 μs
240 B	61.15 μs	74.70 μs	67.48 μs
280 B	70.44 μs	69.24 μs	77.41 μs
3,264 B	6.99 μs	6.96 μs	6.96 μs
$\approx$204,800 B	1047.84 μs	991.15 μs	979.37 μs

Table 3: Average message transfer times for different message sizes for the simulated execution of **LU.D.4096** on the **3D mesh target platform** using DOR_r, PNC_r, NCD_r as communication models.

For the approximation we used a biquartic polynomial with 25 terms, which should provide low deviation from the real values [16]. To prove that claim, we compared the output of the Sage-based simulation with polynomial approximation of HAEC-SIM. In Figure 2, the approximation shows a good conformity with the simulation, e.g., we measured a mean relative error of 0.248 % for 6 hops. Hence, we verified the correlation between the results of the simulation and the polynomial approximation.

5.3 Simulation of Applications on the Target Systems

We conducted the simulations with a total of 4096 MPI processes on 256 Taurus nodes. The MPI processes were mapped to the compute nodes of the target system by using the *xyz* and the *min md* mapping strategies. Identifying known optimal mappings for each application on each topology is part of ongoing work. Therefore, in this work we consider a mapping that is oblivious to the communication patterns of the application, *xyz*, and a mapping that is aware of it, *min md*.

The mapping strategy *xyz* is application oblivious. It identifies the compute node to assign to an application process by first increasing the x coordinate of the last assigned node until every node along the x dimension is assigned, then by increasing the y coordinate and, finally, the z coordinate in the same manner. In contrast to xyz, the mapping strategy *min md* is application aware. It aims to minimize the Manhattan distance between pairs of highly communicating application processes that communicate over the network. The algorithm selects the process pair that exchanges the largest number of point-to-point messages and maps them to nodes with a Manhattan distance of 1 hop, starting at the origin of the system topology. The next such process pair is selected and mapped to system nodes with a Manhattan distance of 1 hop. If no free node within 1 hop radius is found, the search radius is increased to 2 hops. Both mappings strategies proceed in a round-robin fashion if the number of application processes is larger than the number of system nodes.

Msg. size	DOR_r	PNC_r	NCD_r
Mapping: xyz			
$\leq$ 200 B	12.09 μs	12.09 μs	12.09 μs
240 B	8.72 μs	8.72 μs	8.73 μs
280 B	8.28 μs	8.28 μs	8.29 μs
3,264 B	9.14 μs	9.14 μs	9.12 μs
$\approx$204,800 B	602.36 μs	602.36 μs	600.82 μs
Mapping: min md			
$\leq$ 200 B	23.15 μs	21.36 μs	21.32 μs
240 B	31.45 μs	28.65 μs	28.55 μs
280 B	32.12 μs	29.23 μs	29.08 μs
3,264 B	6.97 μs	6.95 μs	6.95 μs
$\approx$204,800 B	677.86 μs	648.65 μs	644.67 μs

Table 4: Average message transfer times for different message sizes for the simulated execution of **LU.D.4096** on the **3D torus target platform** using DOR_r, PNC_r, NCD_r as communication models.

Msg. size	DOR_r	PNC_r	NCD_r
Mapping: xyz			
1,960 B	135.39 μs	134.75 μs	127.33 μs
11,760 B	279.49 μs	284.53 μs	265.64 μs
$\approx$204,800 B	1353.89 μs	1323.39 μs	1259.74 μs
Mapping: min md			
1,960 B	135.32 μs	131.81 μs	126.46 μs
11,760 B	357.27 μs	337.60 μs	321.77 μs
$\approx$204,800 B	1798.28 μs	1771.92 μs	1713.67 μs

Table 5: Average message transfer times for different message sizes for the simulated execution of **BT.D.4096** on the **3D mesh target platform** using DOR_r, PNC_r, NCD_r as communication models.

Since we are interested in verifying the communication models used in the simulations, we concentrate on analyzing the message statistics of the simulated applications on the target simulated system using the performance analysis tool Vampir [11]. Vampir was launched in parallel using 2048 MPI processes due to the size of the simulated trace files. Each of the simulated trace files had a size of about 284 GB for LU.D.4096 and of about 92 GB for BT.D.4096.

The average message transfer times per each message size are shown for each application in Tables 3-6. LU.D.4096 contains 1,975,849,596 point-to-point messages, the majority of which are of 240 B (i.e., 1,231,836,480) and 280 B (i.e., 708,305,976). BT.D.4096 contains 394,813,440 MPI point-to-point messages, the majority of which are of 11,760 B (i.e., 194,310,144) and 1,960 B (i.e., 194,310,144).

The message transfer times in Tables 3-6 are comparable yet clearly distinguishable for each communication model: PNC_r and DOR_r result in the slowest transfer times for BT and LU, respectively. NCD_r achieves the fastest transfer times for both

Msg. size	DOR_r	PNC_r	NCD_r
Mapping: xyz			
1,960 B	30.93 μs	30.24 μs	30.19 μs
11,760 B	91.17 μs	88.87 μs	88.60 μs
$\approx$204,800 B	860.10 μs	828.86 μs	824.43 μs
Mapping: min md			
1,960 B	87.51 μs	80.78 μs	78.89 μs
11,760 B	232.19 μs	211.13 μs	204.17 μs
$\approx$204,800 B	1384.92 μs	1282.32 μs	1259.63 μs

Table 6: Average message transfer times for different message sizes for the simulated execution of **BT.D.4096** on the **3D torus target platform** using DOR_r, PNC_r, NCD_r as communication models.

applications. The interpretation of these results is in agreement with prior work [4, 15]. Thus, these results constitute suitable measurements for our verification process.

5.4 Number of Hops Traveled per Message

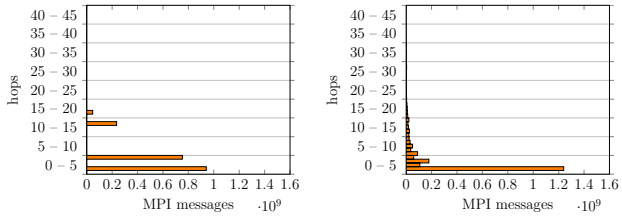
The number of hops traveled by each message is used as input to the verification. Together with the size of the messages, the verification tool can evaluate in parallel the expected message transfer times and compare it with the transfer times obtained by HAEC-SIM. Because the total number of hops and their distribution depends on the processes-to-nodes mapping and application communication patterns, these values are useful to compare different mapping strategies.

We analyzed the number of hops per message for both applications (LU.D.4096, BT.D.4096), both target HPC platform topologies (3D mesh, 3D torus), and both mappings strategies (xyz, min md). Figures 3 and 4 show the number of messages per hops traveled. The mapping strategy min md yields a smoother and wider distribution than xyz. Expectedly, fewer messages are traveling a higher hop count on the 3D torus topology than on the 3D mesh topology.

To compare the mappings numerically, we deduced the overall number of hops traveled by all messages. The total sums can be seen in the Tables 7 and 8. For BT.D.4096 running on a $16 \times 16 \times 16$ 3D torus, the xyz mapping is a naturally perfect fit for the communication pattern of the application. In contrast, the min md mapping for this case is less efficient with $\approx 50\,\%$ more hops traveled. In all other scenarios, min md mapping reduces the total number of hops traveled vs. xyz mapping.

5.5 Verification of Message Transfer Times

All message transfer times produced by the HAEC-SIM simulations were compared using the verification tool against those obtained from an independent implementation of the communication models (cf. Figure 1). As mentioned in Section 5.3, the two applications LU.D.4096 and BT.D.4096 have $1,975,849,596$ and $394,813,440$ MPI point-to-point messages, respectively. Since we used three different communication models, two different topologies, and two different mappings, we conducted 12 simulations for each ap-



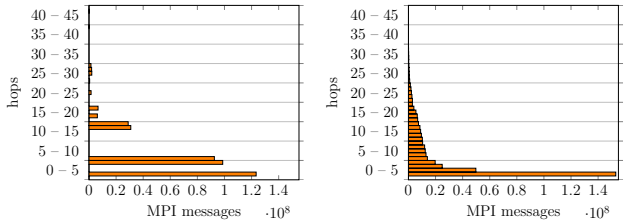
(a) 3D mesh, xyz (b) 3D mesh, min md

(c) 3D torus, xyz (d) 3D torus, min md

Figure 3: Number of hops traveled by the point-to-point messages for the **LU.D.4096** simulated onto the two target platform topologies 3D mesh (upper row) and 3D torus (lower row) using the two mappings xyz (left column) and min md (right column).

	LU.D.4096	BT.D.4096
xyz	7,762,266,270	2,208,487,680
min md	6,048,588,664	2,137,480,380

Table 7: Total number of hops traveled for all point-to-point messages on the **3D mesh target platform**.



(a) 3D mesh, xyz (b) 3D mesh, min md

(c) 3D torus, xyz (d) 3D torus, min md

Figure 4: Number of hops traveled by the point-to-point messages for the **BT.D.4096** simulated onto the two target platform topologies 3D mesh (upper row) and 3D torus (lower row) using the two mappings xyz (left column) and min md (right column).

	LU.D.4096	BT.D.4096
xyz	5,221,888,218	1,394,184,960
min md	3,766,463,068	2,091,663,000

Table 8: Total number of hops traveled for all point-to-point messages on the **3D torus target platform**.

plication. For 21 out of the 24 scenarios, we observed no deviation between the message transfer times obtained with HAEC-SIM and with the verification tool. However, for each of the remaining three scenarios exactly one type of message (same message size and number of hops traveled) caused a very small deviation of 1 picosecond. The picosecond range is, however, the finest time granularity used by HAEC-SIM. Deviations in this range may occur due to rounding errors. Moreover, these deviations were not very common. Overall, only 54,432 of all 28,447,956,432 point-to-point messages (or 0.00019 %) result in such a picosecond deviation. Furthermore, these (rounding) errors do not accumulate over simulated time, thus resulting in a negligible deviation in the simulation results.

6. CONCLUSIONS AND FUTURE WORK

Simulation results have little or no value if the modeling and simulation process is erroneous. Verification can detect errors in the concept, requirements, design, and coding of simulation models. In this work, we concentrated on the code verification of the implementation of three resilient communication models in HAEC-SIM. Comparison of the simulated results against those from the independent implementation shows consistency. Of the investigated models, PNC_r and NCD_r perform similar, and both outperform DOR_r. Thus, this work represents the first step towards sufficiently accurate HAEC-SIM-based simulations for studying the behavior of communication-intensive applications running on the HAEC Box using the verified resilient communication models.

In the present, we are implementing routing and path selection strategies to support the HAEC Box as a simulated target topology. Future work directions include developing communication models that account for link congestion and for collective communication (or multicast messages). Verification and validation of the models to be developed constitute important parts of our future work.

ACKNOWLEDGMENTS

This work is in part supported by the German Research Foundation (DFG) in the Collaborative Research Center 912 Highly Adaptive Energy-Efficient Computing. The authors thank Daniel Besmer of University of Basel for his contribution to the communication-aware mapping. The authors also acknowledge Carl Albing of US Naval Academy for providing the information on NERSC computing system [14]. The compute resources for the simulations were provided by ZIH, TU Dresden.

REFERENCES

1. Alverson, R., Roweth, D., and Kaplan, L. The Gemini System Interconnect. In *Proc. of IEEE 18th Annual Symposium on High Performance Interconnects (HOTI)* (2010).
2. Bailey, D., Barszcz, E., Barton, J., Browning, D., Carter, R., Dagum, L., Fatoohi, R., Fineberg, S., Frederickson, P., Lasinski, T., Schreiber, R., Simon, H., Venkatakrishnan, V., and Weeratunga, S. The NAS Parallel Benchmarks. RNR Technical Report RNR-94-007, NASA, 1994.
3. Bielert, M., Ciorba, F. M., Feldhoff, K., Ilsche, T., and Nagel, W. E. HAEC-SIM: A Simulation Framework for Highly Adaptive Energy-Efficient Computing Platforms. In *Proc. of the Conference on Simulation Tools and Techniques* (2015).
4. Bielert, M., Feldhoff, K., Ciorba, F. M., Pfennig, S., Franz, E., Ilsche, T., and Nagel, W. E. Verification of Resilient Communication Models for the Simulation of a Highly Adaptive Energy-Efficient Computer. Poster at SC15: The International Conference for High Performance Computing, Networking, Storage and Analysis, 2015.
5. Casanova, H., Giersch, A., Legrand, A., Quinson, M., and Suter, F. Versatile, Scalable, and Accurate Simulation of Distributed Applications and Platforms. *Journal of Parallel and Distributed Computing 74*, 10 (2014).
6. Chou, P. A., Wu, Y., and Jain, K. Practical Network Coding. In *Proc. of Annual Allerton Conf. on Comm., Control, and Computing* (2003).
7. Fettweis, G., Nagel, W., and Lehner, W. Pathways to Servers of the Future. In *Proc. of the Design, Automation Test in Europe Conference Exhibition* (2012).
8. Franz, E., Pfennig, S., Matthiesen, B., Scheunert, C., and Jorswieck, E. A. Energy Models for Communication of Future Computing Platforms. In *Proc. of Workshop on Broadband Wireless Communication between Computer Boards (Atto-Nets) at IEEE ICUWB* (2015).
9. Hoefler, T., Schneider, T., and Lumsdaine, A. LogGOPSim: Simulating Large-scale Applications in the LogGOPS Model. In *Proc. of the 19th ACM Intl. Symp. on High Perf. Dist. Comp.*, ACM (2010).
10. Johnson, D., Menezes, A., and Vanstone, S. The Elliptic Curve Digital Signature Algorithm (ECDSA). *Int. Journal of Information Security 1* (2001).
11. Knüpfer, A., Brunst, H., Doleschal, J., Jurenz, M., Lieber, M., Mickler, H., Müller, M. S., and Nagel, W. E. The Vampir Performance Analysis Tool-Set. In *Tools for High Performance Computing*, Springer (2008).
12. Labarta, J., Girona, S., and Cortes, T. Analyzing Scheduling Policies using Dimemas. *Par. Co. 23*, 1-2 (1997).
13. MITRE. Systems Engineering Guide. **`http://www.mitre.org/publications/systems-engineering-guide/systems-engineering-guide`**, 2014.
14. NERSC. Hopper Interconnect Description. **`https://www.nersc.gov/users/computational-systems/hopper/configuration/interconnect/`**, 2015.
15. Pfennig, S., and Franz, E. Adjustable Redundancy for Secure Network Coding in a Unicast Scenario. In *Proc. of International Symposium on Network Coding* (2014).
16. Pfennig, S., Franz, E., Ciorba, F. M., Ilsche, T., and Nagel, W. E. Modeling Communication Delays for Network Coding and Routing for Error-Prone Transmission. In *Proc. of the 3rd Intl. Conf. on Future Gen. Comm. Techn.*, IEEE (2014).
17. The Sage Developers. *Sage Mathematics Software (Version 6.9)*, 2015. `http://www.sagemath.org`.
18. Tikir, M., Laurenzano, M., Carrington, L., and Snavely, A. PSiNS: An Open Source Event Tracer and Execution Simulator for MPI Applications. In *Proc. of the 15th Intl. Euro-Par Conf. on Par. Proc.* (2009).
19. Totoni, E., Bhatele, A., Bohm, E., Jain, N., Mendes, C., Mokos, R., Zheng, G., and Kale, L. Simulation-Based Performance Analysis and Tuning for a Two-Level Directly Connected System. In *Proc. of IEEE 17th International Conference on Parallel and Distributed Systems (ICPADS),* (2011).
20. Velho, P., and Legrand, A. Accuracy Study and Improvement of Network Simulation in the SimGrid Framework. In *Proc. of the 2nd International Conference on Simulation Tools and Techniques*, Simutools '09, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering) (ICST, Brussels, Belgium, Belgium, 2009).
21. Wilke, J. J., Sargsyan, K., Kenny, J. P., Debusschere, B., Najm, H. N., and Hendry, G. Validation and Uncertainty Assessment of Extreme-scale HPC Simulation Through Bayesian Inference. In *Proc. of the 19th International Conference on Parallel Processing*, Euro-Par'13, Springer-Verlag (Berlin, Heidelberg, 2013).