Hybrid PDES Simulation of HPC Networks Using Zombie Packets

ELKIN CRUZ-CAMACHO, Computer Science, Rensselaer Polytechnic Institute, Troy, United States
KEVIN BROWN, Argonne National Laboratory, Lemont, United States
XIN WANG, Computer Science, University of Illinois Chicago, Chicago, United States
XIONGXIAO XU, Illinois Institute of Technology, Chicago, United States
KAI SHU, Illinois Institute of Technology, Chicago, United States
ZHILING LAN, University of Illinois Chicago, Chicago, United States
BOB ROSS, Argonne National Laboratory, Lemont, United States
CHRISTOPHER CAROTHERS, Computer Science, Rensselaer Polytechnic Institute, Troy, United States

Although high-fidelity network simulations have proven to be reliable and cost-effective tools to peer into architectural questions for high-performance computing (HPC) networks, they incur a high resource cost. The time spent in simulating a single millisecond of network traffic in the highest detail can take hours, even for static, well-behaved traffic patterns such as uniform random. Surrogate models offer a significant reduction in runtime, yet they cannot serve as complete replacements and should only be used when appropriate. Thus, there is a need for hybrid modeling, where high-fidelity simulation and surrogates run side-by-side. We present a surrogate model for HPC networks in which: packets bypass the network, while the network state is left untouched, *i.e*, suspended. To bypass the network, we use historical data to estimate the arrival time at which every packet should be scheduled at; to suspend the network, all in-flight packets are scheduled to arrive at their destinations, and are kept in the system to awaken as zombies when switching back to high-fidelity. Speedup for a hybrid model is relative to the proportion of surrogate to high-fidelity. This light-weight surrogate obtained up to 76× speedup. Keeping the zombies in the network showed an increase in the accuracy of the high-fidelity simulation on restart when compared to restarting the network from an empty state.

CCS Concepts: • **Networks → Network simulations**; **Network performance modeling**; • **Computing methodologies → Parallel computing methodologies**.

Additional Key Words and Phrases: Parallel Discrete-event Simulation, HPC networks, Surrogate simulation

1 Introduction

Parallel discrete event simulations (PDES) are used to accurately model complex HPC networks and drive important co-design studies for emerging systems, including, quality of service [2], congestion control [17], performance analysis [23], and application interference [11]. To speed up high-fidelity simulations, which might take hours to run a single millisecond of virtual time, we can make use of low-fidelity equivalents [8, 13, 15], also

Authors' Contact Information: Elkin Cruz-Camacho, Computer Science, Rensselaer Polytechnic Institute, Troy, New York, United States; e-mail: cruzce@rpi.edu; Kevin Brown, Argonne National Laboratory, Lemont, Illinois, United States; e-mail: kabrown@anl.gov; Xin Wang, Computer Science, University of Illinois Chicago, Chicago, Illinois, United States; e-mail: xwang823@uic.edu; Xiongxiao Xu, Illinois Institute of Technology, Chicago, Illinois, United States; e-mail: xxu85@hawk.iit.edu; Kai Shu, Illinois Institute of Technology, Chicago, Illinois, United States; e-mail: kshu@iit.edu; Zhiling Lan, University of Illinois Chicago, Chicago, Illinois, United States; e-mail: zlan@uic.edu; Bob Ross, Argonne National Laboratory, Lemont, Illinois, United States; e-mail: rross@mcs.anl.gov; Christopher Carothers, Computer Science, Rensselaer Polytechnic Institute, Troy, New York, United States; e-mail: chrisc@cs.rpi.edu.

ACM 1558-1195/2024/7-ART
https://doi.org/10.1145/3682060

called surrogates. Hybrid simulations can drive up significant savings on time. These combine the best of both worlds: surrogates and high-fidelity simulation, both phases coexisting. Yet in hybrid simulations, the state of the surrogate phases can veer away from high-fidelity simulations if left unchecked for long enough.

We propose a hybrid strategy for network simulations that allows for a reduction in runtime while maintaining the high-fidelity phases close to those of the ground truth, the high-fidelity exclusive simulations. Our strategy consists of three phases: (i) a high-fidelity phase, we run a simulation until we reach a steady state and we stay in the steady state for some time collecting packet-latency data to train a surrogate (a packet-latency predictor); then, we transition to (ii) a surrogate phase, where instead of simulating all network activities, the surrogate predicts how long a packet will take to arrive at its destination and schedules its arrival for the predicted time; and, finally we transition back into (iii) a high-fidelity phase, where the state of the high-fidelity network is reconstructed so that the high-fidelity does not start from an inaccurate network state.

To make our strategy a success, there are three major challenges to be tackle: first, how to implement the switch between high-fidelity and surrogate modes when running on a parallel discrete-event simulation (PDES); second, how to track packet-latency data and train the surrogate online; and, third, how to bring the state of the network post-surrogate close to that that network would be if no surrogate had been used.

We maintain simulation consistency by suspending the network at a stable state, which ensures a higher accuracy for the high-fidelity simulation upon restart as opposed of ignoring the state of the network. Suspending means that we stop processing any in-flight packets in the network until re-animation. Suspended in-flight packets are tagged as *zombies* while their copies are sent directly to their destinations, thereby bypassing the network and completing each packet's journey. When the high-fidelity packet routing is resumed, re-animated zombie packets are treated as normal network traffic by routers but are discarded at their destination computing nodes. In this way, zombies keep the network in a hot state when switching back to high-fidelity packet simulation.

We propose a packet-latency predictor based on the average packet-latency collected at the steady state for each source to destination pair. Computing the average requires a constant amount of memory and basic arithmetic operations which make for a lean and quick predictor. Given enough packet-latency data within the time window of collection, the average packet-latency predictor should reproduce the throughput of the underlying network simulation.

Suspending the network on a PDES-based simulation requires careful manipulation of events due to the optimizations used by highly efficient PDES frameworks such as ROSS [1]. Under parallel optimistic simulations, our suspension strategy requires augmenting the rollback machinery and various optimistic computation optimizations. We have extended the CODES [18] simulation toolkit and its underlying PDES engine, ROSS, with the ability to pause a simulation at an arbitrary point in time on sequential or parallel execution. We pause the simulation at a steady state, during which, we switch the underlying behavior of the simulation, from one mode to another.

We demonstrate our approach on three dragonfly network configurations: 72-, 1056- and 8448-nodes, and on three synthetic traffic patterns: uniform random, all-to-all and bisection. We obtained up to 76× speedup using hybrid modeling on surrogate mode. Furthermore, using re-animated zombies to restart the PDES network model increased the accuracy of the post-surrogate phase of hybrid simulations compared to cases without zombies. For example, we found that on a 72-node network congested with uniform random traffic, the mean absolute percentage error (MAPE) of the average packet-latencies for the post-surrogate high-fidelity simulation with zombies outperformed by 10% that of a simulation without them, with the MAPE values of 15.656% to 5.600%, respectively.

Our current use case is simple yet it shows the potential gains of surrogates for PDES network simulations. Although, the traffic patterns we used converge into a stable state permanently, and thus there is no need to switch back to high-fidelity, we force a switch to high-fidelity for to reasons: to evaluate the impact of re-enabling the network with its challenges, and to pave the way for realistic traffic patterns which exhibit alternating behavior. To find out when to switch, we ran a high-fidelity simulation and found a good place to force a switch. This renders the efforts of speedup nil. Further work has to be done in order to find good places to switch as the simulation progresses.

The key contributions of this work are:

(1) We present a framework for hybrid simulation of computer networks.
  - A hybrid simulation starts in high-fidelity mode, and can switch back and forth to surrogate mode.
  - We define a simple surrogate model based on the average packet latency seen in high-fidelity.
  - We propose a strategy to maintain the network consistency between high-fidelity runs with surrogate transitions via the use of suspension and zombie packets.
  - We modify the PDES engine, ROSS, to allow for a consistent switch at arbitrary points in a simulation.

(2) We run a suite of experiments showing the speedup gains and accuracy of the hybrid model.
  - The surrogate performs well over a varied number of synthetic workload conditions (random uniform, all to all, and bisection).
  - We show that relatively large networks (1056 and 8448 virtual computing nodes) can be effectively sped up.
  - We present evidence for the accuracy improvement made by suspending the network using zombie packets.
  - We found and fixed a limiting factor when scaling up the number of nodes in the simulation. The routing algorithm contained an inneficiency on a table look-up that was replaced with the use of cache.

2 Background and Related Work

2.1 HPC Network Traffic

Distributed HPC applications transmit messages among processes distributed across the network. Messages are broken into packets, which are further broken down into *flits* at link-level for better utilization of link-level resources [5, pg. 224]. A packet, for example, might be of size 4096 bytes, while flits are often small and in the order of dozens of bytes (*e.g*, 48 bytes). Network often bundle and manage flits in groups, called *chunks* in CODES parlance, where a chunk represents one or multiple flits. The time that takes to move a packet from its source to its destination is called its end-to-end packet-latency. This time is dependent on how many other packets/flits are already in the network, also called *in-flight* to their destination. In certain common scenarios, the network can enter a steady state where the number of in-flight packets and the resulting packet-latencies are relatively stable. It is relatively easier to train a surrogate model to accurately approximate stable network behaviors under these steady states than during more dynamic states.

The dragonfly network architecture is one of the most common architectures for HPC given its low cost and good performance [12]. A key feature of dragonflies is their hierarchical structure with groups connected via high-speed global connections. Routers within groups are fully connected (intra-group connections), while inter-group connections are such that the number of hops going across groups is minimized to one global hop.

In CODES, a simulation is broken down into three components: workloads, terminals and routers. Workloads simulate the computing nodes, the terminals simulate their network interface controllers (NICs) and the routers are routers.

2.2 Accelerating Discrete Event Simulations

Multi-fidelity (as well as multi-scale) modeling came to be from the necesity of running large, complex models faster [4, 7]. Portions of large models can be carved out and approximated by analytical or data-driven tools, thus speeding up simulation times. For PDES, we aim at reducing the work done per unit of virtual time via the reduction of number of events generated. This can can be achieved in two different ways: making a simulation more granular (*e.g*, decreasing the number points in a mesh), or changing how in a layer of LPs interact (*e.g*, routing packets directly to the computer node destination, thus skipping all in-between links).

Various multi-resolution and hybrid PDES models focused on reducing the number of events per link have been proposed. Liu [15] and Gu et al. [8] demonstrated one such approach by combining a fluid flow model with an event-based model to simulate network traffic. Both models predict the hop-by-hop activity of the flows and packets as well as the updated buffer states at each hop. Li et al. [14] aggregate multiple packets into one and approximate the per hop latency through analytical models for transition rate. Making CODES' *chunks* larger is an approximation similar to that presented in their work.

He et al. [9] switches between an end-to-end TCP flow model and a packet-based discrete event model for simple TCP networks. To ensure the discrete event model remains statistically accurate throughout the simulation, packet buffers are checkpointed (frozen) when switching to the flow model, and buffers are restored when switching back to the event-based model. The simulation is broken into epochs, where each epoch is delimited by transitions between flows. A flow starts when a TCP connection is attempted and it ends when the TCP closes. A flow model is trained during the first fraction of an epoch, and then a transition is triggered to rely only on the flow model. Because the switch to the flow model happens at a stage in the TCP connection where data is being transmitted orderly, the receiving node can easily update the state of the transmission without fussing for the in-flight packets. We do need to take care of in-flight packets as there is no order guarantee. It is uncertain how this framework performs when epochs are too small to allow the flow model to make progress in the transfer. While this approach successfully maintains statistically accurate buffer states, the work does not explore the challenges in the context of parallelized simulation execution, which is required for simulating large-scale HPC networks.

Lavin et al. [13] implemented a strategy to accelerate the simulation of a memory subsystem in the SST simulator. They propose a strategy to have multi-fidelity simulations, where easily-predicted stable phases of the simulation are replaced by a low-fidelity surrogate. They determine when to switch to low-fidelity based on statistical technique to find the stable phases. A key difference between their work and ours is the component being abstracted by the surrogate: theirs is a surrogate for the synthetic workload generator while ours is for the computer network. This difference presents unique challenges since maintaining and restarting the network in a consistent state requires manipulating the flow of events across all routers in the simulated network.

Another closely related work in goals and spirit is Rao and Wilsey [21] work on Asynchronous Transfer Mode (ATM) simulations. They built a multi-resolution PDES computer network simulator with the express purpose of accelerating simulation times. Their key insight is that a hierarchical structure, like that of a computer network, can be abstracted into black boxes with a clearly defined API that defines how each component interacts with others. Through this strategy, they were able of neatly defining all links as a single gigantic box. This box can be swapped for another simpler box dynamically. Additionally, they propose two strategies for the dynamic switch of components: proactive, defined by the user before starting the simulation, and reactive, rollbacking the simulation to a previous state to switch a component after finding out that it had to be switched. In contrast with our work, they swap a component composed of multiple LPs (the network) for a single LP. Besides the simple difference on the kind of computer network simulated (ATM, theirs, vs. Dragonfly, ours), Rao and Wilsey's work differs from ours in two important ways: their theoretical largest performance improvement was 10×, ours in practice has shown to be 76×, and they allow packets to go missing when switching, which is inadmissible for us. We solve the problem of packets being dropped at switch time via either suspending the network or allowing the network to run concurrently with the surrogate.

3 Hybrid Modeling using *Zombie* Events

To demonstrate the feasibility of our hybrid modeling approach, we implemented a packet-level surrogate and the mechanism to switch back and forth between high-fidelity mode and surrogate mode in the CODES simulation toolkit. CODES supports high-fidelity flit-level network simulations that enable investigating low-level activities such as adaptive routing [10] and quality of service [2]. A single millisecond of network traffic can take hours to simulate in high-fidelity mode. Our goal is to train a surrogate model that approximates the behavior of the network such that it can replace the slower PDES network model with accurate, lightweight predictions. To this end, we extended both CODES and its underlying PDES engine, ROSS, to pause and continue the simulation at user-defined timestamps. During these periods, the underlying network is swapped for a surrogate and back. The critical requirement of maintaining consistent PDES network model states during the surrogate phases is achieved by using zombie packets to keep the PDES network populated with in-flight packets.

Notice that "surrogate" in our context means a data-driven approximation of a high-fidelity model. Our high-fidelity model and the surrogate are contained within the same data structure, namely, an LP. The dragonfly terminal LP can switch between two modes: high-fidelity, where it routes packets downstream, or surrogate, where it asks a predictor to estimate the arrival of a packet and sends it directly to the destination. A predictor is an opaque object which is fed packet-latency data and returns estimations. On our setup, each terminal LP has one predictor which only gets fed packet-latency information from its parent terminal, so that both terminal LP and predictor live in the same computing core.

Further details on the multiple interactions between the terminal, the predictor and the PDES engine can be found in the subsection below.

3.1 Implementation details

The switch from high-fidelity to surrogate mode and back to high-fidelity mode is shown in Figure 1. Our surrogate causes packet events to bypass the network routers in the PDES network model. A terminal behaves differently depending on the simulation mode: high-fidelity or surrogate. When we switch to surrogate mode, the terminal does not inject the packet into the network, but instead, it asks a *predictor* to estimate the end-to-end latency for the packet and schedules its arrival at its destination. During surrogate mode, the predicted packet can be seen bypassing the network. Switching back to high-fidelity mode requires only switching how the terminal handles packets so that it can inject them into the network again.

During the surrogate mode, if the routers continue processing in-flight packets and progressing them to their destinations, the network will eventually become vacant. If this happens, the network will be empty when we switch back to high-fidelity mode. In some cases, like when the network is barely utilized, this will not be a problem. However, if the network is *hot* (*i.e*, congested with packets in-flight) prior to the switch to surrogate mode, we have to keep the network *hot* on restart in order to preserve high-fidelity model accuracy. This is accomplished by suspending the network model state and re-animating it on the switch back. To make the hybrid modeling accurate, we propose the following solutions to the challenges that parallel hybrid modeling for networks presents.

3.1.1 Challenge 1. Switching parallel simulation modes.

Parallel discrete-event simulation (PDES) partitions the simulated environment across multiple physical compute cores. Under the paradigm of optimistic PDES, each compute partition ignores what the others do for some period of time while it simulates its own partition. Once a given time or number of steps has passed, all computing nodes coordinate with each other and determine how far they have to rollback individually to stay in synch with the others. The nodes rollback to a common agreed virtual time when all their activities have synchronized. This time is called global virtual time (GVT). After rollback at GVT, the state of the whole simulation is guaranteed to be consistent. We take advantage of this consistent global state point to trigger a *director function* call, which will trigger a switch if it deems it so.

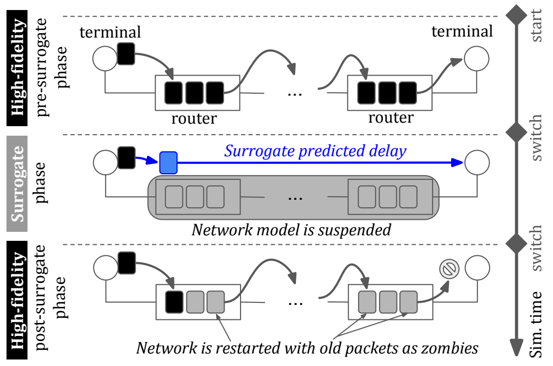

Fig. 1. Hybrid simulation's modeling phases.

The director function is also in charge of setting the global variable that informs the terminals of the current simulation mode (either high-fidelity or surrogate).

To make our approach useful for a range of PDES use cases, we made it so that we can trigger any arbitrary function at GVT and not only a director function. Triggering the function only at GVT guarantees that no causality links are broken as all computing cores are in sync. Furthemore, any events processed pre-GVT have already happened and will not be rollbacked. This means that at GVT, we can schedule new events and they cannot be cancelled. We use this later on to notify destination terminal LPs of zombies in the network.

A particular feature of ROSS is its ability to run deterministically [16]. More precisely, the order in which events are processed is deterministic, but not the number of GVT operations or when they happen. Running a director only at non-deterministic GVTs breaks all determinism. To prevent this from happenning, we can force a GVT at some deterministic virtual time and only at those special GVTs will the director trigger a switch. There are two strategies to force a GVT at a specific virtual time: at every GVT check if you have passed your forced GVT time and trigger a user-directed (user-level) rollback [6], or insert a check in the event loop process to stop right before you hit the forced GVT. Either strategy will be able to position us at precisely an arbitrary GVT of our choice. A user-level rollback might have a lower impact than a check per event, sadly this strategy is not feasible in ROSS due to some lower-level optimizations.

Our function call at GVT implementation is flexible and can be used for many other tasks in need of a consistent state across the whole simulation. It can be used for example to, implement checkpoint/restart, gather statistics on PDES, and even switch global PDES parameters on-the-fly (*e.g* restrict optimistic execution [22], batch event processing size, or resize the preallocated space for events).

3.1.2 Challenge 2. Packet-latency predictor. In order to bypass the network, we need to estimate the end-to-end latency for every packet on each node. To acquire data on which to base these estimates, we record the individual packet-latencies during the first phase of the hybrid simulation, *i.e* when running in high-fidelity mode. We then compute the average packet-latency per source-destination pair for all node pairs in the system and use this information to populate look-up tables. In our implementation, each computing node has a table containing the average packet-latency of transmitting a packet to every destination. Estimating the packet-latency of a new packet is a simple look-up on a table.

Activating the surrogate, and thus using the packet-latency predictor, is as simple as switching the behavior of each computing node individually from packet injection into packet-latency prediction and arrival scheduling. This means that a single global variable per PE can control whether every virtual computing node runs in high-fidelity or surrogate mode.

Table 1. Hybrid-lite vs hybrid execution strategies. See Figure 1 for the hybrid strategy in action. Although, the hybrid-lite strategy forgoes network model suspension and zombification of packets, it separates the simulation on pre-surrogate, surrogate and post-surrogate phases, as hybrid does.

What is being turned on / activated	Hybrid-lite	Hybrid
		Network surrogate
network suspension	×	✓
zombies	×	✓
packet-latency predictor	✓	✓

Solving these two challenges (1 and 2) leaves us with our first complete surrogacy strategy, which we nickname: ***hybrid-lite***. To compare the capabilities of hybrid-lite against the other strategy we implement in this work see Table 1.

3.1.3 Challenge 3. Network suspension and reanimation. Hybrid-lite (no network suspension) suffers from low network accuracy because we are allowing the network to continue processing in-flight packets during the surrogate phase. Once we switch back from surrogate mode into high-fidelity, the network will most likely be vacated and in a state that is inconsistent with its expected high fidelity state. Thus, we need network suspension. However, to properly suspend the network, we have to inspect the network state in detail which in turn requires a more involved director function. We implemented a director function able to read all scheduled events, so that it can find and process all in-flight packets properly.

Suspending the network means freezing all in-flight packets in place, which in turns means that those in-flight packets will be stuck until restart. However, this would interfere with the simulated workflow. To allow the simulation to continue without unduly perturbing the flow of active traffic, we have to schedule a copy of each in-flight packet to arrive at their destinations within an appropriate time. We make use of the same predictor strategy used by the surrogate to estimate the arrival time for all in-flight packets. To determine what packets are in-flight, we have two alternatives: keep a tally of all sent packets or inspect every router for the in-flight events. The former methodology is far simpler but not massively scalable. The latter requires each router to trigger an event to the source terminal for it to handle it. Either way, once the terminal knows of an in-flight packet, the terminal predicts its packet-latency and schedules its arrival at its destination. In this work, we have stick with the simpler approach as it allows us for a bit more modularity (the underlying network can be easily abstracted away).

Finally, to prevent packets from being delivered a twice after the network is restarted and frozen in-flight packets are reanimated, we tag suspended in-flight packets as zombies. Zombie packets behave like any other packet/flit at the link-level. This allows the network to restart in a *hot* state. However, zombies are discarded on arrival at their destination computing node. This is occurs transparently to the workload to prevent the workload from seeing duplicated traffic.

When we turn on the predictor and suspend the network, we are turning on a *network surrogate*. We call this strategy of turning on and off the surrogate ***hybrid***. As a comparison of the differences between hybrid-lite and hybrid see Table 1.

3.1.4 Bonus Challenge. Finding and fixing inneficiencies on top of the network layer. By ignoring the network, the speedup gained by the surrogate only depends on two factors: the remaining simulation components (workload generation and some pieces of the terminal), and the packet-latency estimation. We theorized that the speedup would be bounded by the packet-latency estimation procedure. After all, the network on a network simulator should take the bulk of the computation. When we scaled up to larger networks though, *e.g*, 8448-nodes, we found that we are mostly bounded by the scalability on the number of virtual computing nodes. In fact, we found that the bulk of the simulation time would be spent at the terminal in high-fidelity, and thus the gains of ignoring the network were negligible. The issue came from a table look-up that was triggered each time a new packet was received. The once, not too expensive, solution of looking-up on a table via iterating element by element was no longer viable. To fix it, we duplicated some data in order to create a cache that is $O(1)$ at look-up. This simple change, on a heavily used function, made for a substantial speedup of high-fidelity on its own without any changes on the simulation results. The change also allowed for a substantial speedup in surrogate mode because the network was again the main resource consuming factor.

3.2 Implementation Validation

To validate our implementation we performed three types of tests: model determinism, correct application termination and throughput consistency.

A simple and powerful test of integrity is checking for deterministic model execution. ROSS and CODES have previously been enhanced with a tie-breaker mechanism that allows for deterministic executions of the same experiment [16]. This means that running the same experiment multiple times should produce the same result. Otherwise, there is a potential problem with the event rollback mechanism. In our validation efforts, we found no differences in the net number of events processed or other metrics between the high-fidelity exclusive and hybrid simulations.

In the process of implementing the surrogate network model, we found it useful to test the surrogate against an application sensitive to packets being dropped. Dropping packets should never happen in the scenario where no network error occurs. Ping-pong with a set number maximum pings will always terminate with the same number of packets sent and received (given the same random seed). In ping-pong, every computing node will send a ping to a random computing node, not itself. Any node that receives a ping, responds to the sender with a pong, to which the sender sends another ping to a random computing node if it has not reached its limit of pings. Keeping the same random seed, we saw no difference between ping-pong running on high-fidelity exclusive, hybrid-lite or hybrid.

Throughput depends on the network configuration, the speed of links, and the rate at which packets are injected into the network. We saw no significant difference between high-fidelity exclusive and hybrid-lite or hybrid on any of our experiments as presented in Section 4.

4 Evaluation

In this section, we explain the experimental setup, the experiments and the results to showcase the usefulness of hybrid modeling for HPC networks and the impact that the network suspension approach with zombies has on the accuracy of the simulation.

4.1 Experimental setup

We ran a suite of synthetic traffic pattern workload experiments on three dragonfly network configurations. The synthetic traffic patterns were:
Uniform random – each computing node sends each message to some other computing node at random.
All-to-all – nodes send messages to all the other nodes at the same time.
Bisection – nodes send messages to nodes that are on the opposite side of the network (for the 72-node network, node 0 sends to 36, 20 sends to 56, and 57 to 21).

All synthetic traffic patterns generate messages of size 1024 at a frequency high enough to achieve 100% injection rate and congest the network, except for the bisection traffic pattern which does not easily stabilize

Table 2. Network configuration parameters for each 1D Dragonfly case.

Total nodes	72	1056	8448
Total routers	36	256	1056
Nodes per router	2	4	8
Routers per group	4	8	32
Total groups	9	33	33
Packet size	1024	1024	1024
Chunk size	64	64	64
Local link bandwidth	2 GB/s	5.25 GB/s	5.25 GB/s
Global link bandwidth	2 GB/s	4.7 GB/s	4.7 GB/s

at high enough frequencies. We include bisection for completeness although it does not overload the network. All network parameters can be seen in table 2. Preliminary experiments showed us that most synthetic traffic patterns stabilize within the 10 ms mark. For all 10 ms simulations, we chose to switch to surrogate mode at 2 ms, while we start tracking the packet latencies for estimation at 1 ms.

Experiments were conducted on the AiMOS supercomputer at CCI (Center for Computational Innovations) [3]. A node on AiMOS consists of a 20-core IBM Power9 processor with a 3.15 GHz clock and 512GiB of RAM.

4.2 Methodology.

For each set of experiments, we ran three simulation modes: high-fidelity exclusive, hybrid-lite, and hybrid. The high-fidelity exclusive mode is our baseline to which we compare the effectiveness of our hybrid modes. We collected snapshots of the router port buffer occupancy, total packets sent per compute node, individual packet latencies, total time spent in the simulation, and time spent in surrogate mode only. Aggregated buffer occupancy is a measure of the total number of in-flight packets. This is used to track the state of the network and determine if it is congested – we determine that the network is congested when the buffer occupancy remains continuously high without monotonically increasing or decreasing. Since the objective of common simulation-based network studies is to predict the packet latencies for different traffic patterns being routed across the network, the accuracy of the packet latency predictions of our hybrid simulations is one of our figures of merit. We focus on the post-surrogate phase of the hybrid simulations since this is the phase impacted by the use of zombies when suspending the network. We compare our the post-surrogate phase of our hybrid simulation against the high-fidelity exclusive simulation using mean absolute percentage error (MAPE). We can also compute the speedup of the surrogate alone on the whole simulation by isolating the time that it skipped in high-fidelity exclusive and comparing it against the time spent in surrogate mode.

4.3 Results.

4.3.1 Maintaining High-fidelity Network State. Contention for network channels and port buffers impact the latency of packets moving through the network. This contention is captured by packets waiting in port buffers on routers and must be maintained throughout the simulation for the accurate prediction of packet latencies. During the surrogate phase when packets are not injected into the network, buffers vacate and the network's state becomes inconsistent with the traffic being generated by the workload. Zombies allow us to maintain this network state during the surrogate phase when packets are not injected into the network. Hence, the simulation can resume during the post-surrogate phase in a relatively consistent state. The results in this section compare the buffer occupancy hybrid simulations with and without zombie packets.

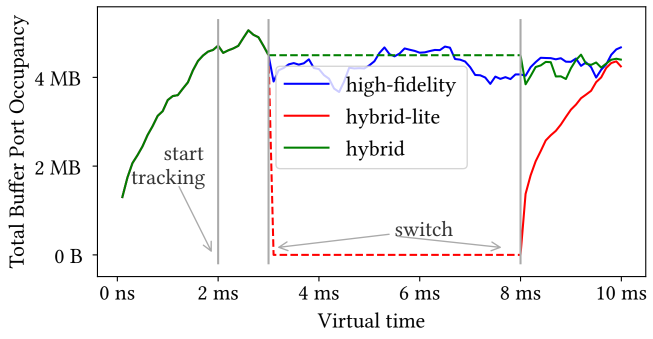

Fig. 2. Aggregated router port buffer occupancy for uniform random traffic pattern. 72-node dragonfly network, using minimal routing, total simulation time of 10 ms.

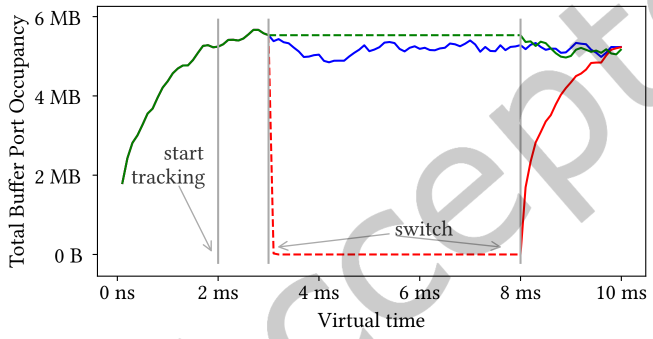

Fig. 3. Aggregated router port buffer occupancy for uniform random traffic pattern. 72-node dragonfly network, using progressive adaptive routing, total simulation time of 10 ms.

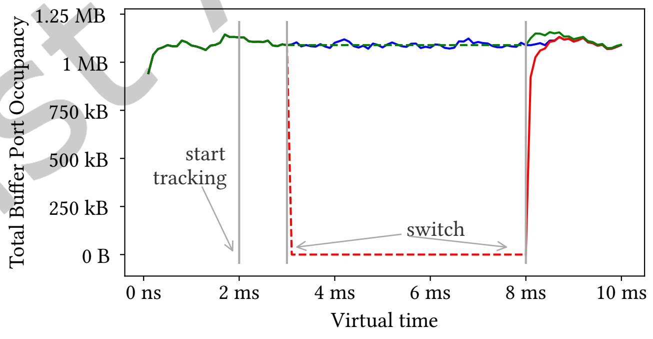

Fig. 4. Aggregated router port buffer occupancy for all-to-all random traffic pattern. 72-node dragonfly network, using minimal routing, total simulation time of 10 ms.

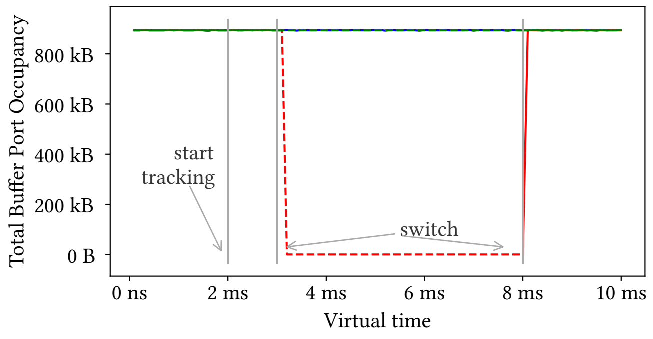

Fig. 5. Aggregated router port buffer occupancy for bisection random traffic pattern. 72-node dragonfly network, using minimal routing, total simulation time of 10 ms.

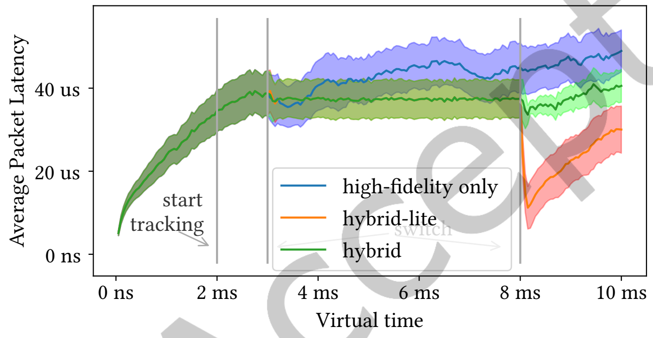

Fig. 6. Average global packet latency with a window of 50 us for uniform random traffic. Error bars represent 20% of the standard deviation. MAPE for hybrid-lite is 52.078% and hybrid is 18.788%. 72-node dragonfly network, using minimal routing, total simulation time of 10 ms.

Figure 2 shows the aggregated buffer occupancy as it changes in time for uniform random using minimal routing. The network vacates quickly in hybrid-lite as it can be seen by the sudden drop to 0 B in the occupancy. Post-surrogate, the network takes approximately 2 ms to arrive at similar state it was before the switch. Suspending the network allows for a network state post-surrogate much closer to what it would have been in high-fidelity exclusive, as seen by hybrid. Although the state of the network between high-fidelity exclusive and hybrid at the switch back is not the same, they lead to closer final states when compared to ignoring hybrid. This result is maintained even when changing the routing algorithm, see Figure 3 for progressive adaptive, and the synthetic traffic pattern, see Figure 4 for all-to-all traffic and Figure 5 for bisection traffic.

4.3.2 Impact of Zombies on Simulation Accuracy. As demonstrated in the previous section, using zombie packets impact the state of the network in the hybrid simulation during the post-surrogate phase. This section shows maintaining the network state with zombie packets impact the accuracy of packet latencies during the post-surrogate phase.

Figure 6 presents the average packet latency for uniform random with minimal routing. Although the average packet latency during the surrogate phase does not represent an accurate latency for either hybrid-lite or hybrid,

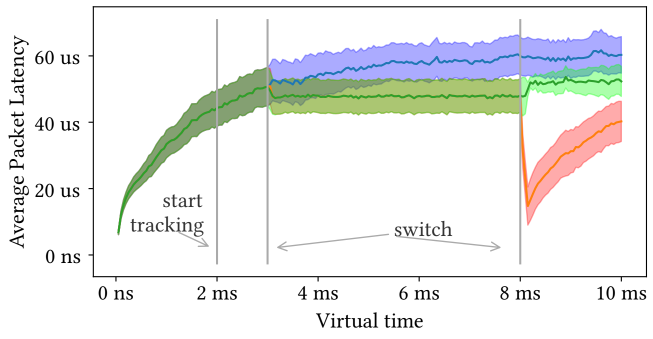

Fig. 7. Average global packet latency with a window of 50 us for uniform random traffic. Error bars represent 20% of the standard deviation. MAPE for hybrid-lite is 48.507% and hybrid is 13.023%. 72-node dragonfly network, using progressive adaptive routing, total simulation time of 10 ms.

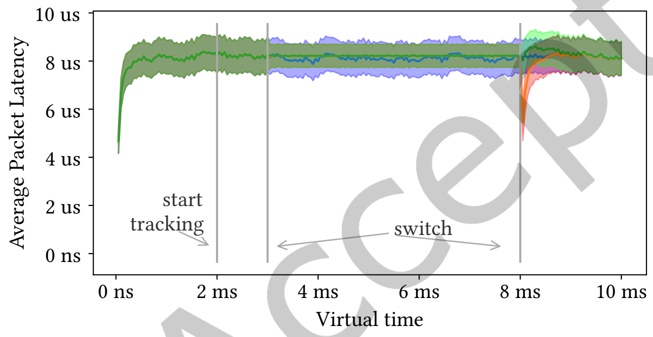

Fig. 8. Average global packet latency with a window of 50 us for all-to-all traffic. Error bars represent 20% of the standard deviation. MAPE for hybrid-lite is 2.104% and hybrid is 1.583%. 72-node dragonfly network, using minimal routing, total simulation time of 10 ms.

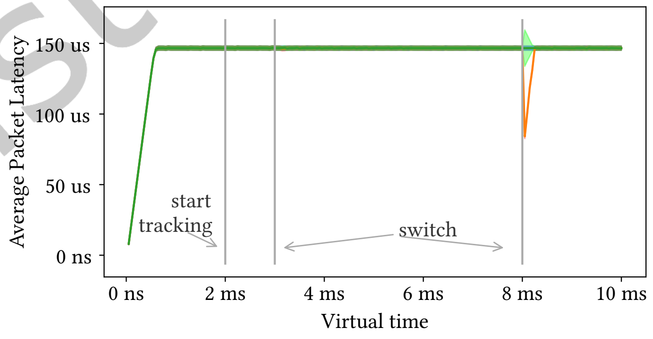

Fig. 9. Average global packet latency with a window of 50 us for bisection traffic. Error bars represent 20% of the standard deviation. MAPE for hybrid-lite is 2.615% and hybrid is 0.037%. 72-node dragonfly network, using minimal routing, total simulation time of 10 ms.

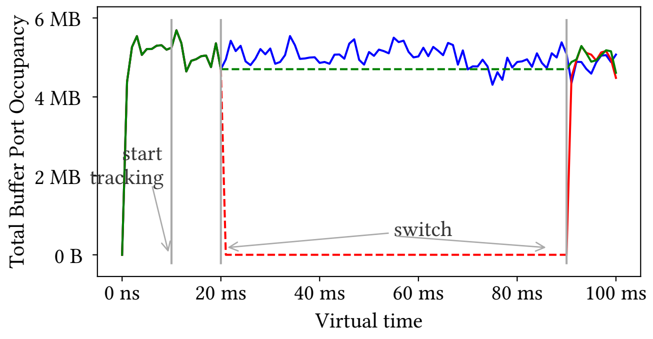

Fig. 10. Aggregated router port buffer occupancy for uniform random traffic pattern. 72-node dragonfly network, using progressive adaptive routing, total simulation time of 100 ms.

the average packet latecy post-surrogate for hybrid does not deviate as far as hybrid-lite does. To quantify these differences, we measured the mean absolute percentage error (MAPE) between hybrid-lite/hybrid and high-fidelity exclusive. We report these MAPE values in Table 3. The lower the MAPE, the closer to high-fidelity the hybrid-lite/hybrid model is. Hybrid outperforms hybrid-lite in all cases including different routing strategies, see Figure 7, and different traffic patterns like all-to-all (Figure 8) and bisection (Figure 9).

4.3.3 Impact of Zombies over Time Scales. The longer we can skip ahead using surrogates, the higher the speedup we can gain from them. A point of concern with running surrogates for long periods of time is the accumulation of error leading to radically different results. As an exemplary case of longer time scales and how suspending the network can lead to more faithful network behavior, we present a 100 ms run of uniform random with progressive adaptive routing in Figures 10-11. In this case, hybrid is 10% more accurate than hybrid-lite, as measured by the difference on MAPE for packet latency predition (5.600% vs 15.656%, Table 3).

Table 3. Validating surrogacy against high-fidelity simulation. On the switch back, the high-fidelity simulation is more accurate for hybrid than for hybrid-lite, as shown by MAPE. *Time between switches* was extracted from high-fidelity runs at the time that the simulation took between the two points in which the hybrid simulation goes into surrogate and back.

Synthetic Workload	Routing Algorithm	Network Size	Virtual Time (and V. Time in Surrogate)	Mode	MAPE (compared to high-fidelity)	MPI Ranks	Total Wallclock-time	Simulation Speed-up	Time Betw. Switches / Time in Surrogate	Surrogate Speed-up
Uniform random	Minimal	72-node	10 ms (5 ms in surrogate)	High-fidelity	-	9	162.53 s	-	81 s / 1.47 s	55.0×
				Hybrid-lite	52.078%		80.90 s	2.00×		
				Hybrid	**18.788%**		80.66 s	2.01×		
Uniform random	Progressive adaptive	72-node	10 ms (5 ms)	High-fidelity	-	9	223.71 s	-	113 s / 1.48 s	76.4×
				Hybrid-lite	48.507%		111.56 s	2.00×		
				Hybrid	**13.023%**		112.88 s	1.98×		
Uniform random	Progressive adaptive	72-node	100 ms (70 ms)	High-fidelity	-	9	2254.49 s	-	1128 s / 20.55 s	76.8×
				Hybrid-lite	15.656%		688.56 s	3.27×		
				Hybrid	**5.600%**		683.41 s	3.30×		
All2all	Minimal	72-node	10 ms (5 ms)	High-fidelity	-	9	158.99 s	-	80 s / 1.26 s	63.5×
				Hybrid-lite	2.104%		79.53 s	1.99×		
				Hybrid	**1.583%**		79.18 s	2.01×		
Bisection	Minimal	72-node	10 ms (5 ms)	High-fidelity	-	9	37.52 s	-	18 s / 0.45 s	40.0×
				Hybrid-lite	2.615%		19.71 s	1.90×		
				Hybrid	**0.037%**		19.07 s	1.97×		
Uniform random	Minimal	1056-node	10 ms (5 ms)	High-fidelity	-	33	1921.14 s	-	964 s / 13.52 s	71.3×
				Hybrid-lite	0.588%		951.47 s	2.02×		
				Hybrid	**0.036%**		957.54 s	2.01×		
Uniform random	Minimal	8448-node	10 ms (5 ms)	High-fidelity	-	33	4875.76 s	-	2476 s / 85.40 s	29.0×
				Hybrid-lite	4.693%		2527.47 s	1.93×		
				Hybrid	**1.594%**		2469.90 s	1.97×		

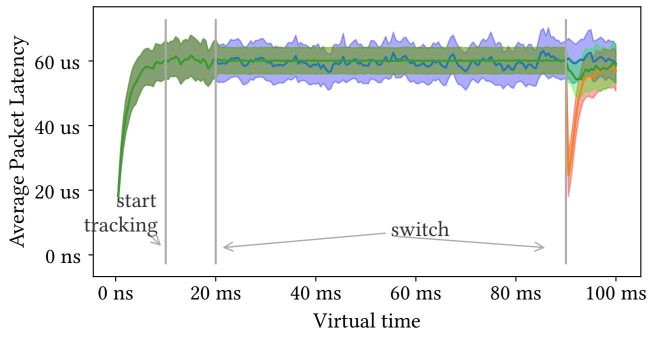

Fig. 11. Average global packet latency with a window of 0.5 ms for uniform random traffic. Error bars represent 20% of the standard deviation. MAPE for hybrid-lite is 15.656% and hybrid is 5.600%. 72-node dragonfly network, using progressive adaptive routing, total simulation time of 100 ms.

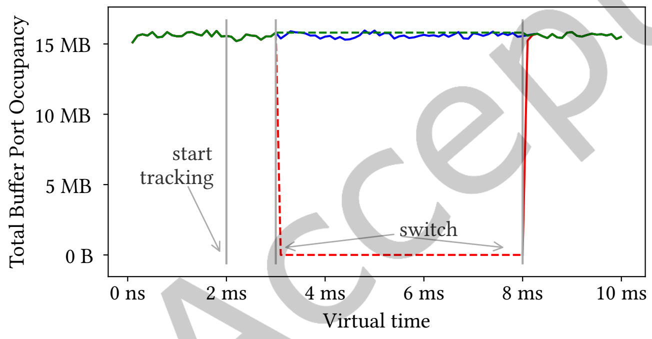

Fig. 12. Aggregated router port buffer occupancy for uniform random traffic pattern. 1056-node dragonfly network, using minimal routing, total simulation time of 10 ms.

4.3.4 Impact of Zombies over System Scales. The 72-node dragonfly network is a small and useful network model. However, given its small size, it is fair to ask whether suspending the network matters at larger scales when more nodes and routers are involved. As it can be seen in Figures 12 and 13, the aggregated buffer occupancies post-surrogate for 1056- and 8448-node network cases show little difference except for that small window of time when the network is warming up in hybrid-lite. That small difference is reflected as a substancial MAPE difference, as it can be seen on Table 3 represented as a small dip on packet latency on Figure 14. The same results can be seen with smaller networks and different synthetic workloads (Figure 9), as well as on larger networks (Figure 15).

5 Discussion.

The effects of hybrid-lite and hybrid on the state of the network are significantly different. The main difference occurs in the post-surrogate phase, packets in hybrid-lite will experience fewer queuing delays while the network fills up again and thus report a lower latency than packets injected at the same virtual time in the hybrid or

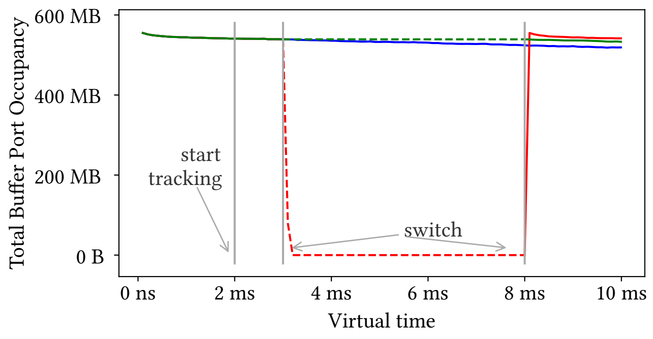

Fig. 13. Aggregated router port buffer occupancy for uniform random traffic pattern. 8448-node dragonfly network, using minimal routing, total simulation time of 10 ms.

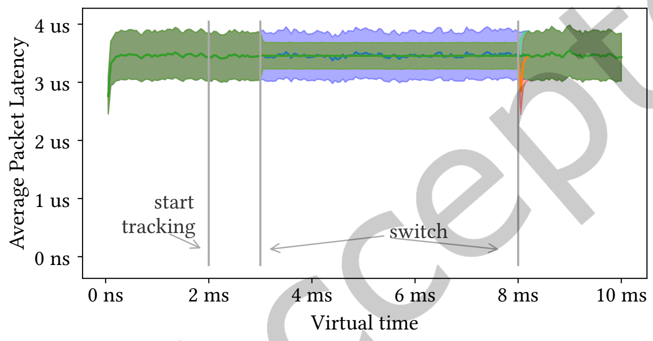

Fig. 14. Average global packet latency with a window of 50 us for uniform random traffic. Error bars represent 20% of the standard deviation. MAPE for hybrid-lite is 0.588% and hybrid is 0.036%. 1056-node dragonfly network, using minimal routing, total simulation time of 10 ms.

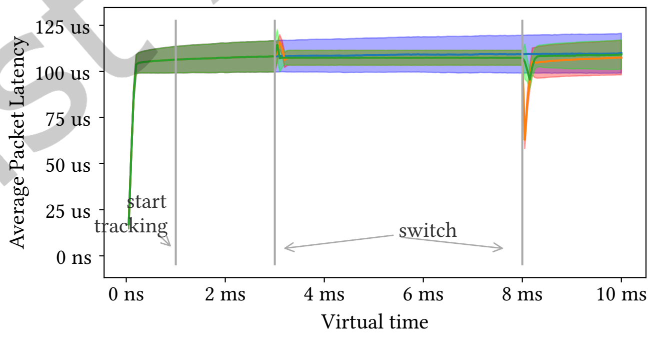

Fig. 15. Average global packet latency with a window of 50 us for uniform random traffic. Error bars represent 20% of the standard deviation. MAPE for hybrid-lite is 4.693% and hybrid is 1.594%. 8448-node dragonfly network, using minimal routing, total simulation time of 10 ms.

high-fidelity cases. Suspending the network and re-animating it with zombies provides a more consistent state as captured by the accuracy of average packet-latency.

We determined when to switch to surrogate mode by looking at the aggregated buffer port occupancy as seen in Figure 2. The network seems to reach steady-state congestion at about 3 ms. We theorized that, a stable enough buffer occupancy can inform us when to automatically switch. However, the average packet-latency trend for high-fidelity exclusive in Figure 6 demonstrates that the traffic performance does not stabilize at 3 ms as the latency trends upward. This discrepancy in buffer occupancy and latency motivates the need for a more insightful approach to identify when the network reaches steady-state using more than just buffer occupancy measurements.

Regardless of the scenario, suspending the network shows better accuracy of buffer port occupancy and, in nearly all cases, more accurate packet-latency predictions without increased computational cost. In other words, even though more sofisticated statistical approaches, such as Nikoleris et al. [19], could drive even larger improvements in network accuracy, zombies provide a zero-operational-cost approach to critically improving simulation accuracy when used to restart the high-fidelity simulation.

The speedup gain obtained by a network surrogate can be bounded by the chosen packet size and chunk size combination—the smaller the packet, less flits that will be skipped, and so the speedup gain will be smaller. Additionally, we anticipate that higher speedups could be gained if instead of predicting individual packet-latencies, individual message latencies would be used instead. That is, the more lower-level, intermediate events that can be skipped, such as packetization, the greater the speedup that can be achieved. Gains in network surrogacy can only go so far; we have seen a 76× speedup with a surrogate, with higher speedups possible if routing algorithms at the terminals are optimized. Thus the need for higher-levels of surrogacy for greater speedup.

Although there is a need for higher-level surrogates, these would not make network level surrogates obsolete. The network level surrogate can be used to keep the consistency of the network simulation. Just the same way that zombies keep the router level data consistent for the network-level surrogate.

Even though this average packet-latency-based surrogate model is very simple, it has two key benefits: it uses few resources, thus very fast, and it proved to be very robust on a variety of traffic patterns we encountered. For more complex scenarios where the average packet-latency predictor is less suited, more complex surrogates such as neural networks maybe be deployed using an interface to connect to PyTorch [20] models via its C++ API.

6 Conclusion

We described a hybrid parallel simulation strategy for computer networks where the high-fidelity simulation is used to inform a fast surrogate that can then replace the high-fidelity network model. The hybrid model starts off in high-fidelity mode and then can switch to and from surrogate mode. To maintain a relatively consistent network state and increase the accuracy of the post-surrogate phase, the network state can be suspended with the help of zombie messages and re-animated to a near accurate state.

We showed that a surrogate which uses the historical average packet-latency leads to up to 76× speedup. Furthermore, suspending the network incurred negligible computational cost overhead. However, due to optimizations required for parallel high-fidelity modeling, special considerations should be taken to implement the suspension strategy for parallel execution.

We found that replacing the network by a surrogate speeds up the simulation but it is limited to the non-network portion of the simulation. The workload generator, on the simulated computing node, becomes a burden as it grows larger and hungrier, thus driving the need for a higher-level surrogate.

6.1 Future work

Network suspension via zombie packets leads to accurate network states, but it requires a state to suspend to start with. We plan to implement other strategies to re-animate or "re-heat" the network.

The (average packet-latency-based) network surrogate shows great performance, but it is limited by the number of packets it can process. An application-level surrogate would abstract away packets altogether, leading to much higher speedups. We plan on integrating application-level surrogates as well as running experiments with more realistic synthetic traffic patterns such as MILC and Jacobi3D, and later with workload traces. Another weakness of the average-based surrogate is on handling complex and non-static traffic patterns. For those, we plan on leveraging our previous work on machine learning on PyTorch [24] in an online learning setting.

Lastly, we plan on finding a strategy to determine when and how a network state is in steady state, using both statistical methods and ML tools. Conversely, during surrogate-mode, we plan to investigate when the steady state will stop in order to resume the high-fidelity simulation.

Acknowledgement: We thank the reviewers for their kind and helpful feedback. This work is supported by the Director, Office of Advanced Scientific Computing Research, Office of Science, of the U.S. Department of Energy under Contract No. DE-AC02-06CH11357 and the "Kronos: Enabling Long Timescale PDES Simulations via Multi-Resolution Methods" project.

References

[1] P. D. Barnes, C. D. Carothers, D. R. Jefferson, and J. M. LaPre. 2013. Warp speed: Executing Time Warp on 1,966,080 cores. In *Proc. of the 2013 ACM SIGSIM Conf. on Principles of Advanced Discrete Simulation (PADS)*. 327–336.

[2] Kevin A. Brown, Neil McGlohon, Sudheer Chunduri, Eric Borch, Robert B. Ross, Christopher D. Carothers, and Kevin Harms. 2021. A Tunable Implementation of Quality-of-Service Classes for HPC Networks. In *ISC High Performance 2021: High Performance Computing (ISC)* ((Virtual) Frankfurt, Germany).

[3] Center for Computational Innovations. [n.d.]. Artificial intelligence multiprocessing optimized system (AiMOS). cci.rpi.edu. (accessed December 13, 2023).

[4] Bastien Chopard, Jean-Luc Falcone, Pierre Kunzli, Lourens Veen, and Alfons Hoekstra. 2018. Multiscale Modeling: Recent Progress and Open Questions. *Multiscale and Multidiscip. Model. Exp. and Des.* 1, 1 (March 2018), 57–68. https://doi.org/10.1007/s41939-017-0006-4

[5] William James Dally and Brian Patrick Towles. 2004. *Principles and Practices of Interconnection Networks.* Elsevier.

[6] Samir R. Das and Richard M. Fujimoto. 1993. A Performance Study of the Cancelback Protocol for Time Warp. In *Proceedings of the Seventh Workshop on Parallel and Distributed Simulation.* ACM, San Diego California USA, 135–142. https://doi.org/10.1145/158459.158476

[7] M. Giselle Fernández-Godino. Fri Dec 01 00:00:00 EST 2023. Review of Multi-Fidelity Models. *ACSE* 1, 4 (Fri Dec 01 00:00:00 EST 2023), 351–400. https://doi.org/10.3934/acse.2023015

[8] Yu Gu, Yong Liu, and D. Towsley. 2004. On integrating fluid models with packet simulation. In *IEEE INFOCOM 2004*, Vol. 4. 2856–2866 vol.4. https://doi.org/10.1109/INFCOM.2004.1354702

[9] Q. He, M. Ammar, G. Riley, and R. Fujimoto. 2002. Exploiting the predictability of TCP's steady-state behavior to speed up network simulation. In *Proceedings. 10th IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunications Systems.* 101–108. https://doi.org/10.1109/MASCOT.2002.1167066

[10] Yao Kang, Xin Wang, and Zhiling Lan. 2022. Study of Workload Interference with Intelligent Routing on Dragonfly. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis* (Dallas, Texas) *(SC '22)*. Article 20, 14 pages.

[11] Yao Kang, Xin Wang, Neil McGlohon, Misbah Mubarak, Sudheer Chunduri, and Zhiling Lan. 2019. Modeling and Analysis of Application Interference on Dragonfly+. In *SIGSIM PADS.*

[12] John Kim, Wiliam J. Dally, Steve Scott, and Dennis Abts. 2008. Technology-Driven, Highly-Scalable Dragonfly Topology. In *Proceedings of the 35th Annual International Symposium on Computer Architecture (ISCA '08).* IEEE Computer Society, Washington, DC, USA, 77–88. https://doi.org/10.1109/ISCA.2008.19

[13] Patrick Lavin, Jeffrey Young, and Richard Vuduc. 2023. Multifidelity Memory System Simulation in SST. In *9th International Symposium on Memory Systems.* Washington, DC, USA.

[14] Ting Li, Nathanael Van Vorst, and Jason Liu. 2013. A Rate-Based TCP Traffic Model to Accelerate Network Simulation. *SIMULATION* 89, 4 (April 2013), 466–480. https://doi.org/10.1177/0037549712469892

[15] Jason Liu. 2007. Parallel Simulation of Hybrid Network Traffic Models. In *21st International Workshop on Principles of Advanced and Distributed Simulation (PADS'07).* 141–151. https://doi.org/10.1109/PADS.2007.26

[16] Neil McGlohon and Christopher D. Carothers. 2021. Toward Unbiased Deterministic Total Ordering of Parallel Simulations with Simultaneous Events. In *Proceedings of the Winter Simulation Conference* (Phoenix, Arizona) *(WSC '21)*. IEEE Press.

[17] Neil McGlohon, Christopher D. Carothers, K. Scott Hemmert, Michael Levenhagen, Kevin A. Brown, Sudheer Chunduri, and Robert B. Ross. 2021. Exploration of Congestion Control Techniques on Dragonfly-class HPC Networks Through Simulation. In *2021 International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS)*. 40–50. https://doi.org/10.1109/PMBS54543.2021.00010

[18] Misbah Mubarak, Christopher D Carothers, Robert B Ross, and Philip H Carns. 2017. Enabling Parallel Simulation of Large-Scale HPC Network Systems. *IEEE Trans. Parallel Distrib. Syst.* 28, 1 (2017), 87–100.

[19] Nikos Nikoleris, Lieven Eeckhout, Erik Hagersten, and Trevor E. Carlson. 2019. Directed Statistical Warming through Time Traveling. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO '52)*. Association for Computing Machinery, New York, NY, USA, 1037–1049. https://doi.org/10.1145/3352460.3358264

[20] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, Soumith Chintala, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. Curran Associates, Inc.. , 8024–8035 pages.

[21] Dhananjai M. Rao and Philip A. Wilsey. 2006. Accelerating ATM Simulations Using Dynamic Component Substitution (DCS). *SIMULATION* 82, 4 (April 2006), 235–253. https://doi.org/10.1177/0037549706067271

[22] Sudhir Srinivasan and Paul F. Reynolds. 1998. Elastic Time. *ACM Trans. Model. Comput. Simul.* 8, 2 (April 1998), 103–139. https://doi.org/10.1145/280265.280267

[23] Noah Wolfe, Misbah Mubarak, Nikhil Jain, Jens Domke, Abhinav Bhatele, Christopher D. Carothers, and Robert B. Ross. 2017. Preliminary Performance Analysis of Multi-Rail Fat-Tree Networks. In *Proceedings of the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing* (Madrid, Spain) *(CCGrid '17)*. IEEE Press, 258–261. https://doi.org/10.1109/CCGRID.2017.102

[24] Xiongxiao Xu, Xin Wang, Elkin Cruz-Camacho, Christopher D. Carothers, Kevin A. Brown, Robert B. Ross, Zhiling Lan, and Kai Shu. 2023. Machine Learning for Interconnect Network Traffic Forecasting: Investigation and Exploitation. In *Proceedings of the 2023 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*. 133–137.

Received 22 December 2023; revised 16 May 2024; accepted 10 July 2024