Fit Fly: A Case Study on Interconnect Innovation through Parallel Simulation

Neil McGlohon
mcglon@rpi.edu
Rensselaer Polytechnic Institute
Troy, New York

Noah Wolfe
woflen@rpi.edu
Rensselaer Polytechnic Institute
Troy, New York

Misbah Mubarak
mmubarak@anl.gov
Argonne National Laboratory
Lemont, Illinois

Christopher D. Carothers
chris.carothers@gmail.com
Rensselaer Polytechnic Institute
Troy, New York

ABSTRACT

To meet the demand for exascale-level performance from high-performance computing (HPC) interconnects, many system architects are turning to simulation results for accurate and reliable predictions of the performance of prospective technologies. Testing full-scale networks with a variety of benchmarking tools, including synthetic workloads and application traces, can give crucial insight into what ideas are most promising without needing to physically construct a test network.

While flexible, however, this approach is extremely compute time intensive. We address this time complexity challenge through the use of large-scale, optimistic parallel simulation that ultimately leads to faster HPC network architecture innovations. In this paper we demonstrate this innovation capability through a real-world network design case study. Specifically, we have simulated and compared four extreme-scale interconnects: Dragonfly, Megafly, Slim Fly, and a new dual-rail-dual-plane variation of the Slim Fly network topology.

We present this new variant of Slim Fly, dubbed Fit Fly, to show how interconnect innovation and evaluation—beyond what is possible through analytic methods—can be achieved through parallel simulation. We validate and compare the model with various network designs using the CODES interconnect simulation framework. By running large-scale simulations in a parallel environment, we are able to quickly generate reliable performance results that can help network designers break ground on the next generation of high-performance network designs.

CCS CONCEPTS

• **Networks** → **Network simulations**; **Network topology types**; *Network performance analysis*; • **Computing methodologies** → **Discrete-event simulation**.

SIGSIM-PADS '19, June 3–5, 2019, Chicago, IL, USA

ACM ISBN 978-1-4503-6723-3/19/06…$15.00
https://doi.org/10.1145/3316480.3325515

KEYWORDS

Interconnection Networks; High Performance Computing; Parallel Discrete Event Simulation; Modeling

ACM Reference Format:
Neil McGlohon, Noah Wolfe, Misbah Mubarak, and Christopher D. Carothers. 2019. Fit Fly: A Case Study on Interconnect Innovation through Parallel Simulation. In *SIGSIM Principles of Advanced Discrete Simulation (SIGSIM-PADS '19), June 3–5, 2019, Chicago, IL, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3316480.3325515

1 INTRODUCTION

When designing a new high-performance computing (HPC) system, the choice of the underlying interconnection network is not to be taken lightly. Performance and cost must be balanced appropriately to prevent large bottlenecks. Moreover, different configurations of the same topology can have varying levels of performance depending on the types of communication featured in a given workload. Simple synthetic workloads cannot reproduce all of the complexities found in real-world applications [14]. Thus, HPC system integrators must rely on exhaustive and fine-grained simulations, testing various configurations, topologies, and workloads to pick the right interconnect for their system. As network features such as routing algorithms, congestion-abatement schemes, and fault tolerance mechanisms get more complicated, predicting full-scale, real-world performance of prospective system designs becomes even more challenging.

We leverage the CODES (Co-Design of Exascale Storage) storage and network simulation toolkit to develop, test, and evaluate exascale topologies and technologies. CODES is a high-fidelity network simulator built on top of the Rensselaer Optimistic Simulation System (ROSS), which is a parallel discrete event simulation (PDES) framework. Previous works have implemented and tested performance of routing schemes such as On-the-Fly Adaptive Routing (OFAR) [11] or Universal Globally-Adaptive Load-balanced (UGAL) routing [29] on networks featured in the CODES toolkit [21, 23, 24, 33]. We show that this framework can be used to evaluate not only existing networks but also newly proposed networks that have not yet been built, in a quick and efficient time frame.

We propose Fit Fly, a variant of the Slim Fly network topology [3], and compare it with state-of-the-art hierarchical networks such as Dragonfly [18] and Megafly [28]. Fit Fly has the same basic

topology and construction as Slim Fly but features a dual-rail-dual-plane configuration. Fit Fly is a product of "what-if?" questions: What if we could (1) double the network's overall bandwidth, (2) further reduce the average number of hops between two terminal nodes in the network, and (3) double the number of routers to reduce possibility of congestion?

Because of its second, independent plane of routers, Fit Fly does incur twice the router cost, but with this cost comes twice the overall bandwidth. Because of the way Slim Fly networks are constructed, they already have high path diversity and low diameter. If we add a second plane of routers "on top" of the first but with the terminal nodes mapped in a mirrored fashion, we can further increase the path diversity and reduce the average number of hops between two nodes. Moreover, with the second injection rail and increased overall bandwidth, packets can be injected into the network at a rate much higher than a single-rail-single-plane Slim Fly network. Additionally, we found that even at a reduced link bandwidth a second plane of routers greatly reduced the impact of interference traffic on application performance.

The main contributions of this work are as follows:

- A variant of the Slim Fly topology and ROSS+CODES network model that enables multiple independent router planes with multiple injection rails at each network terminal.
- An example methodology for how hypothetical questions can be quickly answered through rapid development and experimentation with the CODES framework. This is demonstrated through a series of experiments on four state-of-the-art network designs of approximately 3,000 nodes.
- A corroboration of the conclusions found in [3, 13] relating low-diameter networks to increased performance as well as the benefits of multiple router planes in combating network congestion.
- Evaluation of Slim Fly and Fit Fly with comparable overall bandwidth showing that the increased performance of Fit Fly over Slim Fly can be maintained even with reduced-cost hardware.

2 RELATED WORK

Work on simulating dual-rail dual-plane fat tree networks [13] identified many potential benefits to the extra bandwidth and router-buffer space supplied by a second, independent, plane of routers. The authors did mention, however, that whether any performance benefits were realized depended on the supplied workload.

Fit Fly is a variant of the Slim Fly topology [3], which is a network class that features a low diameter. In their work, the authors show that lowering the diameter of a network can reduce not only the latency of operations on the network but also the cost of the network in terms of both construction and operating energy.

Being able to answer hypothetical "what-if?" questions was a motivating point for the CODES toolkit [1], and evidence supporting the value of these types of ventures has been presented in several works. In [24] the authors describe tests of various networks on varying workloads including synthetic and HPC network traces. They show that CODES can be used by network designers to gain crucial insight about prospective network designs in an efficient manner.

The authors of [14] argue that testing networks with a diverse set of workloads and configurations is important to obtaining reliable predictions of real-world network performance. They introduce TraceR and use it to add support for more types of HPC traces into CODES.

In addition to the CODES simulator used by this work, there are many other network simulation tools with similar goals and uses.

BigNetSim [6], a PDES system based on the Pose simulation environment provides packet level simulation with sequential, conservative parallel, and optimistic parallel execution modes. It features several synthetic traffic generators as well as application trace support.

INSEE [26], is a network simulation and evaluation environment that joins a functional simulator and traffic generator into one simulation system. While it only operates under sequential execution it consumes very little memory due to its functional nature. The authors note that a 64,000 node simulation was run while consuming only 2GB of memory.

BookSim [16], is a sequential-execution-only network simulator with similar traffic generation and application trace features. While it only works sequentially, it is a *cycle-accurate* flit-level simulator making its results very reliable. This does, however, limit the overall size of simulatable networks.

More information on various tools and systems for simulating interconnection networks, as well associated challenges, can be found in [2].

3 PARALLEL SIMULATION BACKGROUND

Accurately measuring performance metrics on varying network models with multiple workloads and configurations requires fine-grained simulation. The more detail that a simulation has, the more complicated the model and the more technical overhead required to run it.

To complete all necessary simulation runs in a timely manner, one would like to be able to run simulations spread across multiple processing nodes. PDES is one possible approach to parallelizing a simulation model. In PDES, any interaction between entities is considered an *event*. Every event has a timestamp representing the time within the global simulation that the event occurred.

We utilize ROSS [4] as the core simulation framework. ROSS abstracts a simulation into logical processes (LPs). Each LP represents an entity in the simulation, for example, a network switch, a network terminal, or a workload server. The set of all LPs is divided among processing elements (PEs), which process events scheduled by their respective LPs in timestamp order. There is one PE per physical MPI rank for the execution of the simulation. Within each PE is an additional organizational abstraction called kernel processes (KPs). KPs are a sort of container for LPs and aid in organization and synchronization of LPs during simulation.

ROSS allows for sequential, conservative, and optimistic execution. *Sequential* execution maps all LPs to a single PE, and events are deterministically executed in exact timestamp order. Since all LPs process and schedule events using the same clock, causality is never infringed upon.

Conservative execution is a type of parallel execution that has additional synchronization overhead to prevent events from ever

being processed out of order across all PEs. Because of this layer of synchronization, causality of events is always correct in this mode.

Optimistic execution allows each LP to schedule events according to its own local clock. Therefore, PEs could process events in an order that disagrees with causality on other PEs. Given relatively infrequent causality errors, optimistic execution is faster than conservative since the latter has a much higher synchronization overhead. Performance of optimistic execution is hindered by the frequency of causality errors, because these must be handled before the simulation can progress. ROSS handles causality errors through reverse computation [5] as an optimization of the *Time Warp* protocol [15].

When out-of-order events are detected, all events resulting from such a conflict must be undone, in other words, reverse computed, until the state of the simulation is what it was just before the incorrect event was scheduled. Each LP is programmed by the model developer with a reverse-computation handler that gives the LP the functionality required to undo an event. This reverse handler typically just undoes any LP state changes made by the forward event handler.

ROSS gives each LP its own set of random number generators (RNGs) that can be independently rolled back any number of times to deterministically reproduce the same sequence of random numbers after being rolled back.

A simple example of a ROSS PDES simulation is a toy random-packet-routing model with each LP in the simulation representing a single router. Routers forward packets to each other randomly through *events* and keep track of how many packets they received. When the simulation is forward progressing, router LPs will receive a packet from a neighboring router. They must then (1) increment its received packet count, (2) randomly pick a new neighbor to forward the packet to, and (3) schedule and send the packet in a new event to the chosen neighbor. When ROSS detects that an event was processed in an order that conflicts with causality, router LPs then have to undo the conflicting events by (1) decrementing its received packet count; (2) rolling back the RNG used to pick a neighbor; and (3) sending an anti-message to the previously chosen neighbor, signifying that the resulting message was sent in error.

The Fit Fly model, as well as the Slim Fly model that it is derived from, was developed by using CODES [1]. CODES is a simulation toolkit built on top of ROSS to enable easier development of HPC network and storage system models [30]. With CODES, users can evaluate features and behavior of network models with varying configurations and HPC workloads. The feature set of CODES is wide ranging. It includes data collection services, a reverse computation stack to simplify complex LP state rollback behavior, and tools for generating synthetic traffic as well as replaying HPC application traces. The supported workloads can be run on a given network independently or simultaneously with arbitrary endpoint allocation mapping to help simulate real-world usage and performance.

The CODES framework maps MPI workload LPs, each representing the MPI ranks that would be operating on physical compute nodes of a given network, to *terminal node* LPs in the network model. The workload LPs operate as the origin and final termination of messages in the network. When a workload LP generates a message, CODES passes the message to the mapped terminal node for injection into the network.

4 FIT FLY NETWORK MODEL

The Fit Fly model is an enhanced version of the Slim Fly model previously added to the network set of CODES [33]. The topology of a Fit Fly network is thus similar to that of Slim Fly, with a lot of overlapping details. Unless otherwise specified, any aspect that applies to Slim Fly also applies to Fit Fly.

Table 1: Description of symbols used to define Slim Fly and Fit Fly networks

h	Nodes connected per router
P	Number of independent router planes
N_{r_p}	Total routers per plane
N_r	Total routers in network ($N_r = N_{r_p} \cdot P$)
N_h	Total nodes in the network ($N_h = N_{r_p} \cdot h$)
k'	Router network radix
k	Router radix ($k = k' + h$)
q	Prime power

The most critical difference between Slim Fly and Fit Fly is that the latter has at least two independent router planes and rails whereas the default Slim Fly model only has one. Each plane contains N_{r_p} routers, which are connected only to other routers in the same plane. While Fit Fly has multiple router planes, the total number of nodes in the network remains the same. Nodes, or terminals, within the network are mapped to routers in each plane by using a scheme detailed in Section 4.2.2.

4.1 Slim Fly Background

In a work that introduced the Slim Fly topology, Besta et al. [3] argued that lowering the network diameter could be beneficial in several ways. First, the lowered network diameter could translate to reduced latency because, on average, fewer hops would be traversed by packets. As a result, packets would be less likely to interfere with each other. Second, fewer routers are necessary to create a connected network, so there is a reduced cost in terms of up-front construction as well as maintaining energy costs. Besta et al. showed that a low-diameter network could be designed without sacrificing high overall bandwidth and while simultaneously reducing costs.

The main objective in the design of the Slim Fly topology was to maximize the number of endpoints N_h with a given network diameter D and vertex (router) radix k. (For notation, see Table 1.) Besta et al. [3] drew parallels between this objective and that of the *degree-diameter problem* [20]. They thus decided to use graphs related to that problem for the backbone of Slim Fly. In order to maximize the number of endpoints but also guarantee the low diameter property, the network is constructed based on a class of diameter-2 graphs commonly referred to as MMS graphs. MMS graphs of diameter-2 are close to the optimum size noted in the degree-diameter problem as they approach what is referred to as the Moore bound [19].

4.2 Slim Fly Topology

Slim Fly network routers are divided into local groups, each with a certain degree of local connectivity. The set of all local groups is

divided between two subgraphs that, for ease of reference, we will call the α and β subgraphs.

Each router in a local group also has a certain degree of global connectivity. We define a global connection as any router-router connection that spans between two local groups. Similarly, local connections are defined as any router-router connections that fall within a local group. *In Slim Fly, there are no connections between groups within the same subgraph.*

The result of these specifications is a bipartite graph with global connections spanning the two subgraphs and several local groups within each subgraph with local connections between routers in each group.

Each router also has a certain number of terminal nodes connected to it. Figure 1a shows an example Slim Fly layout. Labeled are the α and β subgraphs, each with four router groups of four routers. Each router in this example has a single terminal host attached. The exact connections from router to router, both within local groups and globally, are not shown in detail because the construction is not trivial. Details of this construction are given in Section 4.2.1.

4.2.1 *Constructing a Slim Fly-Class Network.*

Given the correct parameters for generation of the topology, the simulation model will calculate the nontrivial link structure and connect the router LPs to their corresponding neighbors. But finding the correct parameters to input is itself nontrivial. To determine the correct input parameters to define the network, we follow the construction-simplified methodology for Diameter-2 MMS graphs laid out in greater detail in [3]. This process comprises five steps.

1. Choose the number of planes p that will exist in the network. Standard Slim Fly has a single plane of routers, whereas Fit Fly has at least two.

2. Find a prime power $q = 4\omega + \delta$, where $\delta \in \{-1, 0, 1\}$, such that $N_{r_p} = 2q^2$ is satisfied for the desired number of routers per plane.

3. Construct a Galois field of order q. Let $\mathbb{F}_q$ be such Galois field. Also, find the primitive element ξ that generates it. ξ is an element that *generates* all other elements of the set. More formally, all nonzero elements of $\mathbb{F}_q$ can be written as ξ^i, where $i \in \mathbb{N}$.

4. Utilizing ξ, construct sets X and X', known as *generator sets*. These generator sets specifically will be used to determine router-router connections within a single network plane using Equations 1–3.

5. Connect terminals sequentially to routers, reversing the order for each consecutive plane.

We can assign each router in the network with four coordinates (s, x, y, p), where $s \in \{\alpha, \beta\}$ indicates which of the two subgraphs the router is in. $x \in \{0, \ldots, q-1\}$ and $y \in \{0, \ldots, q-1\}$ represent the local group that the router resides in and its position within the group, respectively. The fourth coordinate, $p \in \{0, \ldots, P\}$ represents the network plane. Since each plane's router connections are independent and identically computed, this process is repeated for each plane p.

$$\text{router}(\alpha, x, y, p) \text{ connected to } (\alpha, x, y', p) \text{ iff } y - y' \in X \quad (1)$$

$$\text{router}(\beta, m, c, p) \text{ connected to } (\beta, m, c', p) \text{ iff } c - c' \in X' \quad (2)$$

$$\text{router}(\alpha, x, y, p) \text{ connected to } (\beta, m, c, p) \text{ iff } y = mx + c \quad (3)$$

We utilize Equation 1 to compute the intragroup (local group) connections of subgraph α. Similarly, Equation 2 computes the intragroup connections within subgraph β. Equation 3 determines the connections between the two subgraphs. A toy example of how the connections can be visualized is presented in Figure 2.

4.2.2 *Fit Fly: Additional Rails and Planes.*

Unless otherwise noted, Fit Fly refers to a Slim Fly network with two independent planes of routers as well as two injection rails on each terminal for introducing and receiving traffic to/from the network. This makes Fit Fly a dual-rail-dual-plane network by default, but it can also be generalized to refer to a multi-rail-multi-plane Slim Fly network of arbitrary order.

As mentioned in Section 4.2.1, Equations 1–3 are used to determine the connectivity within a single network plane. Since Fit Fly has at least two planes, we must perform the same calculations for each plane. We emphasize that there are no connections between any two routers on separate planes.

What joins the planes together to form a single cohesive network are the terminals. The multiple planes share the same set of terminal nodes. Figure 1 shows a general layout of both Slim Fly and Fit Fly. In Figure 1a each terminal has a single connection to a router, following the standard Slim Fly design. Figure 1b shows each terminal having a connection to a single router on each plane.

While the router connections within each plane are identical, the mapping of terminals to routers in each plane is not. When connecting terminals to a Fit Fly network, the terminals are connected via two schemes depending on the plane being linked to. If we number the planes with the ID p in the range [0,P), the mapping is as follows. In even-numbered planes, terminals are mapped with the default Slim Fly scheme; this means that terminal node 0 is connected to router (α, 0, 0, p) and so on. In odd-numbered planes, however, terminals are mapped in reverse; terminal 0 is connected to router (β, $q-1$, $q-1$, p) and so on in those planes. This scheme increases the overall path diversity and reduces the average hop count.

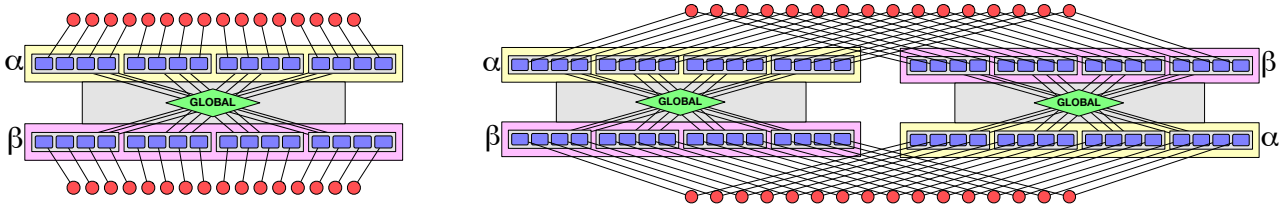
4.3 Routing and Planar Selection

Currently three algorithms for packet routing within a plane are implemented in the Slim Fly and, by extension, Fit Fly models: minimal, nonminimal, and adaptive. These routing algorithms remain unchanged from previous work [33].

While Fit Fly utilizes the same routing algorithms as does Slim Fly on a per plane basis, an additional scheme must be in place in order to determine which rail, and consequently which plane, a packet will be passed through at the originating terminal. Three different schemes have been implemented for this selection: `PATH`, `CONGESTION`, and `RANDOM`. The best choice of planar selection algorithm depends on what type of experiments are to be performed.

4.3.1 `PATH` *Planar Selection.*

When the network simulation is configured to utilize the `PATH` selection algorithm, the originating terminal evaluates the minimal path length to the destination terminal along each rail. The shortest path is then chosen. In the event of a tie for the shortest path, the `CONGESTION` algorithm is used.



(a) Slim Fly (Single-Rail-Single-Plane)

(b) Fit Fly (Dual-Rail-Dual-Plane)

Figure 1: General layout of the Slim Fly and Fit Fly topologies. Networks shown are simplified for ease of understanding; normal networks will have far more routers and terminal nodes per router. Subfigure (a) shows an example of the Slim Fly topology. Each terminal has a single connection to a router (single-rail) in either subgraph α or subgraph β. Subfigure (b) shows an example of the Fit Fly topology. Each terminal has two connections (dual-rail) to routers, one connection to a router in each of two planes of independent routers (dual-plane). One distinction of Fit Fly is that the second plane of routers is identical to the first but mirrored, increasing the total path diversity further.

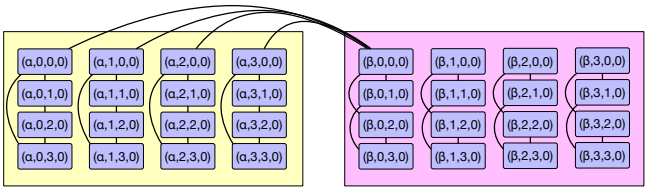


Figure 2: Example visualization showing the three distinct ways connections could be determined. Many global connections are not shown for simplicity. Note that this is for a single plane, denoted by the fourth coordinate. The same process is used for all planes, and there are no connections between routers on different planes.

4.3.2 CONGESTION *Planar Selection.*

The CONGESTION scheme evaluates the number of packets set for injection along each rail at the originating terminal. Whichever rail has the fewest number of packets awaiting injection will be chosen. This method is intended to increase the load-balancing properties of the network and prevent performance degradation from a single rail becoming overly congested.

4.3.3 RANDOM *Planar Selection.*

A more trivial way to balance load across the various planes is to allow for the originating terminals to select the rail of injection uniformly at random.

5 VALIDATION

The Slim Fly CODES network model has previously been validated against an independently developed Slim Fly simulation for minimal, nonminimal, and adaptive routing [17, 33]. Fit Fly utilizes the same base model as Slim Fly with minimal changes to enable links to the additional rails/planes. Because no real-world Fit Fly model or other simulation exists, validation can be performed only in comparison with the Slim Fly model itself.

Since Fit Fly's main feature is its increased overall bandwidth, we have performed a set of offered versus accepted load experiments. These experiments show that the additional rails and planes do, in fact, yield a greater accepted load amount that is proportional to the number of rails in the network. While Fit Fly generally refers to dual-rail-dual-plane Slim Fly, we have expanded the model to allow for arbitrary number of rails/planes to show flexibility. Figure 3 shows the results of these runs. Accepted load was measured by counting the number of bytes received by terminals and dividing by the duration that the count was measured. Additional rails give the networks greater capacity and overall bandwidth and are therefore able to accept greater amounts of offered load.

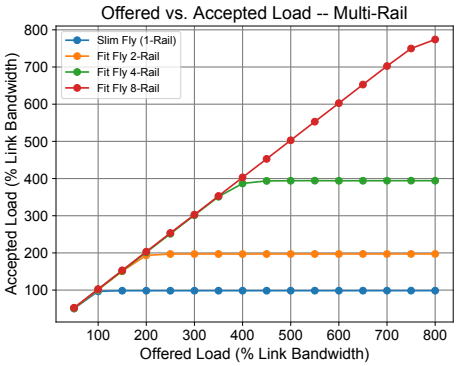


Figure 3: Slim Fly, dual-rail, quad-rail, and octo-rail Fit Fly networks were offered varying loads by sweeping the mean amount of time between when packets of fixed size were injected into the networks. The number of router planes in each network strictly equals the number of rails. Here load is shown as a percentage of the link bandwidth (12.5 GiB/s).

6 EXPERIMENTS

In this section we describe the various experiments that we have performed to compare the Fit Fly network model with others of similar size, in terms of the number of both terminals and routers.

For all experiments in this work, we allocate terminals for a primary workload and secondary background workload with the exception of zero interference (baseline) runs. The same random allocation map was utilized for each network.

Table 2: Configurations of Slim Fly, Fit Fly, Dragonfly, and Megafly networks used for performance comparison

	Slim Fly	Fit Fly	Dragonfly	Megafly
Router Radix	28	28	36	36
Planes	1	2	1	1
Rails	1	2	1	1
Groups	26	52	19	10
Node Count	3042	3042	3078	3240
Router Count	338	676	342	360
Global Connections	4732	9464	3078	3240
Nodes/Group/Rail	117	117	162	324
Global Connections / Group	169	169	162	324
Link Bandwidth	12.5 GiB/s	12.5 GiB/s	12.5 GiB/s	12.5 GiB/s
Nodes per Router	9	9	9	18 (Leaves only)
Routing Algorithm	Adaptive (UGAL)	Adaptive (UGAL)	Adaptive (PAR) [35]	Adaptive (PAR)
Planar Selection Scheme	n/a	`CONGESTION`	n/a	n/a

6.1 Workloads

The networks in this paper are evaluated with two separate HPC network traces provided by the Design Forward Program [7]. These traces are given as a DUMPI MPI trace format [31], which lays out the specific details of MPI messages sent during the course of real-world HPC applications.

We used traces from the following applications: AMG and MultiGrid. Note that each of these traces are sourced from applications with multiple ranks per host but are replayed on our network models with a 1:1 rank/host ratio to spread out communication across the network, forcing hosts from all parts of the network to participate in the primary workload.

6.1.1 Algebraic MultiGrid Solver (AMG) Workload.
AMG is an algebraic Multigrid solver miniapp for unstructured mesh physics packages. The specific trace used in our experiments has 1,728 MPI ranks. We map each rank to its own terminal node in the given network. The application from which this trace was created had a runtime of 2.2 seconds, including computation time, with 1,728 MPI ranks on 72 hosts and 52% of the time spent on communication. [7]

6.1.2 MultiGrid (MG) Workload.
MultiGrid is a geometric multigrid cycle miniapp from the adaptive mesh refinement application framework BoxLib [10]. The specific trace used in our experiments has 1,000 MPI ranks each of which, similarly to our AMG workload, is mapped to its own terminal node in the simulated network. The application from which this trace was created had a runtime of 1.4 seconds, including computation time, with 1,000 MPI ranks on 42 hosts. The application spent 3.7% of the time on communication [7].

6.1.3 Synthetic Workload.
In addition to the primary workloads (AMG and MG traces), we use a synthetic uniform random workload to simulate various levels of induced traffic load on the network. With a mean interval of 100μs, we vary the payload size to alter the intensity of interference.

6.2 Evaluation Metrics

The three primary metrics that we analyze are the maximum communication time of the primary workload, the average packet latency of all packets in the network, and the average number of hops traversed by all packets in the network. The maximum communication time is the total amount of time spent by any one rank of the primary workload from the first MPI message it sends to the final. The average packet latency is important for qualifying how long packets are spending in transit and in router buffers. This metric is correlated with the communication time but is application agnostic. The total number of hops traversed can give insight into how much deviation packets experience as a result of the adaptive routing algorithm due to congestion.

6.3 Other Evaluated Networks

We evaluate four state-of-the-art networks: Slim Fly, Fit Fly, Dragonfly, and Megafly. Since in Section 4 we already discussed the specifics of how Slim Fly and Fit Fly topologies are designed, this section covers some background of the latter two networks.

6.3.1 Dragonfly. The specific type of Dragonfly network that we simulate in this work is a 1D Dragonfly topology introduced in [18]. It features a hierarchical design of groups of routers. Routers within groups are locally connected in an all-to-all pattern. In addition to local and terminal node connectivity, routers have some degree of global connectivity, or links to routers in other groups. *Every group must have at least one global connection to every other group in the network.*

This type of Dragonfly network is not to be confused with the 2D-Dragonfly topology featured in the Cray XC systems known as Cray Cascade [8]. While there are similarities, the local groups of the 2D-Dragonfly feature a grid type connectivity instead of all-to-all.

6.3.2 Megafly. The network type called Megafly, also known as Dragonfly+, is a derivative of the 1D Dragonfly network introduced in [28]. Whereas Dragonfly has all-to-all local connectivity, Megafly features a two-level fat tree for each local group. The result is that within each group of routers is a fully connected bipartite graph.

On one half are routers, called *leaf routers*, that have terminal node connections and local connections. On the other half of the bipartite graph are routers, called *spine routers*, that have global connections to other groups and local connections. Spine routers have no terminal connections, and leaf routers have no global connections. Similarly to Dragonfly, every group must have at least one global connection to every other group in the network.

6.4 Cross-Network Comparison

In this section we show a direct comparison of the Fit Fly model not only to Slim Fly but also to a couple of dragonfly-class networks. The intention is not to argue that one network is definitively better than another but instead to show how flexible the CODES framework is for quickly testing multiple types of networks with varying workloads and interference intensities. We feel that these networks also give some added context for the power of Slim Fly-class networks.

Table 2 gives the parameters of the four networks tested in this section. Because of the difference in the guidelines or restrictions for generating these networks, a true comparison with all aspects of the networks being the same is difficult. As a result, our Slim Fly network has many more global connections than either Dragonfly or Megafly has. Fit Fly, because of its second router plane, has even more.

The link bandwidth that we have chosen for our network configurations is to match the Mellanox InfiniBand EDR specifications. However, we have used 12.5 GiB/s instead of 12.5 GB/s for configuring our networks in order to match previous experiments. As a result, our bandwidth configurations slightly exceed their respective Mellanox InfiniBand counterpart by approximately 7% but should negligibly affect overall observed trends and final results.

Figure 4 shows the results of the experiments performed on the AMG1728 workload.

In Figure 4a we note that the networks are closely matched at low levels of background interference but start to exhibit some performance loss at higher levels of traffic injection. We can see that Fit Fly well outperforms each network, which is expected by looking at the statistics from Table 2. We reason that Fit Fly performed so well because of its increased overall bandwidth: it has twice the routers and thus twice the number of links for packets to be transmitted across and therefore less opportunity to encounter interfering congestion.

Comparing the average number of hops traversed by packets in the network can also give insight into how well each network handles congestion. Figure 4c plots the average number of hops traversed by all packets in the network, not taking into account the different workloads. Slim Fly and Fit Fly show great advantage over the Dragonfly and Megafly networks in this regard as well, because of the low diameter of the Slim Fly-based topologies. Fit Fly again takes the lead, notably because the extra routers give it a distinct advantage for packets to be routed more directly to their destination. With a constant packet injection in a load-balanced network, the likelihood that any two packets will find themselves queued on the same router decreases as the number of routers increases.

Figure 5 shows similar results from the MultiGrid1000 trace workload. In Figure 5a we observe that each network appears to handle increasing levels of background traffic up to 15% of link bandwidth unperturbed, with Fit Fly narrowly outperforming them all. Fit Fly remains unimpeded even at the highest level of background interference. Figure 5c shows that Fit Fly did not need to reroute many packets, on average, away from their minimal paths.

We note that the convention that CODES follows for counting hops is that the count is incremented when received by a router in the network—not on receipt by a terminal. Thus, a packet that originates at a terminal node, visits two routers, and then is received by the destination terminal will have a path hop count of two.

6.5 Equalized Bandwidth Comparisons

The configuration of Fit Fly given in Table 2 was chosen to change as few variables as possible from the chosen Slim Fly configuration. Thus, since our Slim Fly network was configured with link bandwidth speeds close to Mellanox InfiniBand (IB) EDR 100 Gb/s specifications, so too was Fit Fly.

This decision, however, gives Fit Fly a distinct advantage. The amount of overall bandwidth, the total amount of bandwidth available to link any two terminals in the network, is far greater in the case of Fit Fly. Additionally Fit Fly has a higher number of routers than do the other networks by an order of 2.

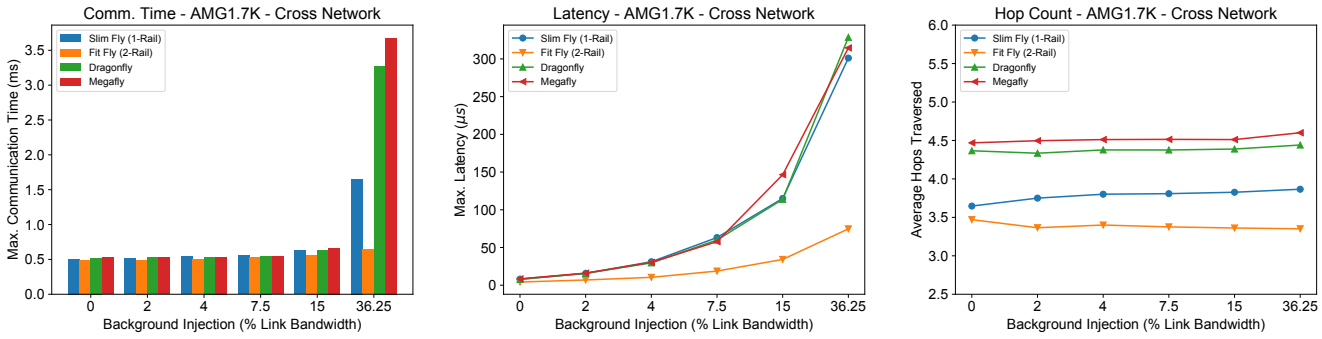
To provide a different perspective, we perform additional experiments where the bandwidths of the two networks are configured to be mutually comparable. In one set we halve Fit Fly link bandwidths to match Mellanox IB FDR 56 Gb/s link specifications as well as experiments where the bandwidth of Slim Fly is instead doubled to match Mellanox IB HDR 200 Gb/s link specifications. The result is two pairs of Slim Fly and Fit Fly networks where the total overall bandwidth between networks in each pair is of a comparable magnitude.

The first comparison uses the same Slim Fly network described in Table 2 but with a modified Fit Fly network where all link bandwidths are nearly halved to match the 56 Gb/s link speed specification. The results of these experiments are shown in Figures 6 and 7. In these figures we observe that while the networks are uncongested, they have similar application performance in each trace. When the network experiences higher levels of injection traffic, however, Fit Fly is able to handle larger levels of interference, even with a reduced bandwidth.

Similarly, if instead of halving the bandwidth of Fit Fly we double the bandwidth of Slim Fly to the level of Mellanox IB HDR 200 Gb/s, we observe the same phenomenon. Normalizing for bandwidth shows that bandwidth is only part of the story. With the total amount of bandwidth between the two networks kept at comparable levels, the only true difference between the two networks is the additional plane and rail that Fit Fly has over Slim Fly.

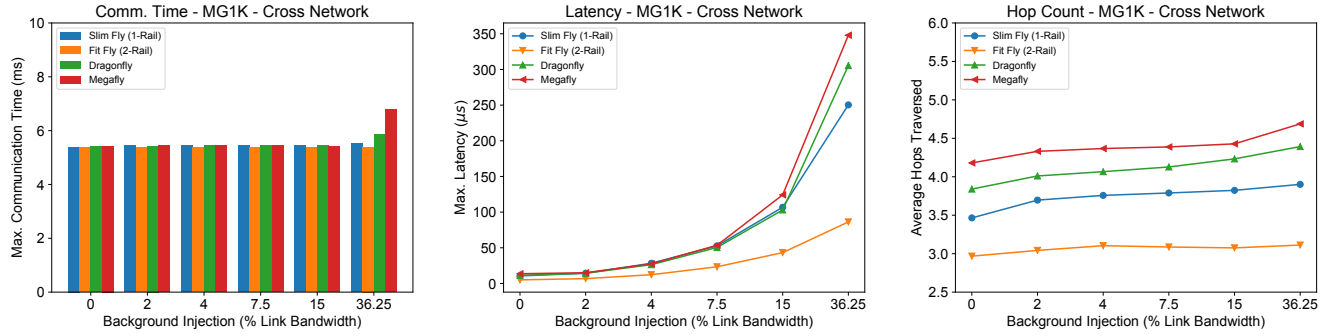
7 DISCUSSION

The demand for stronger performance from HPC systems continues to grow. Building exascale level systems and overcoming the challenges associated with that endeavor will require a lot of innovative techniques and ideas.



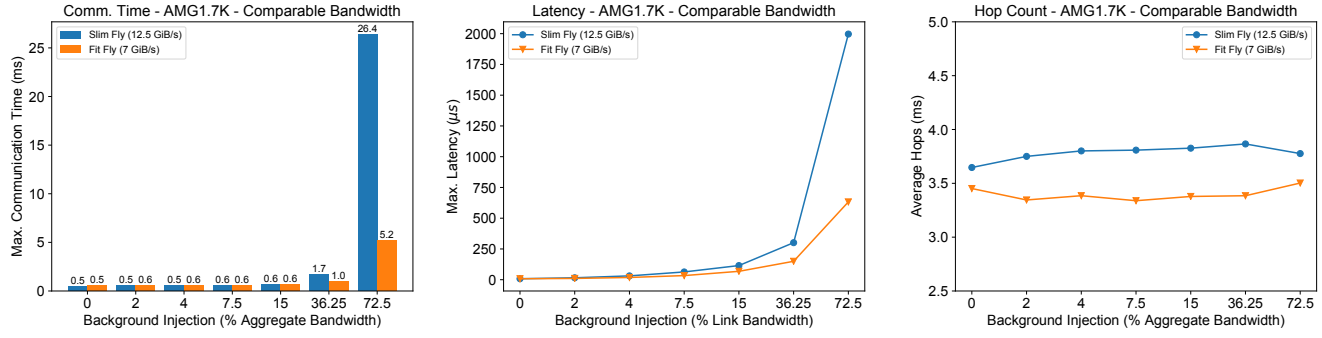
(a) Application Communication Time (b) Latency (c) Hop Count

Figure 4: Synthetic interference experiments on the AMG1728 trace workload with 1,000 synthetic background ranks. Each synthetic rank injects packets into the network at a fraction of the total link bandwidth of 12.5 GiB/s (≈InfiniBand EDR). The same workloads and allocations were used across four networks.



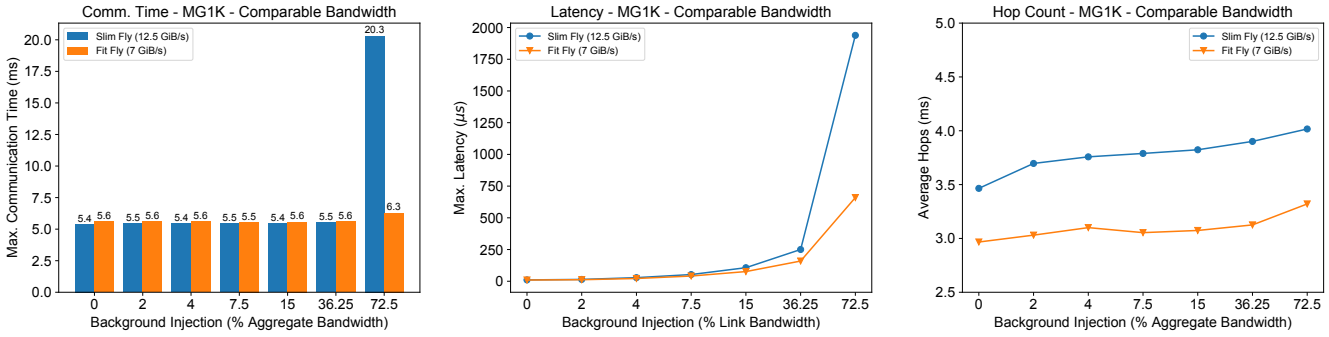
(a) Application Communication Time (b) Latency (c) Hop Count

Figure 5: Synthetic interference experiments on the MultiGrid1000 trace workload with 1,000 synthetic background ranks. Each synthetic rank injects packets into the network at a fraction of the total link bandwidth of 12.5 GiB/s (≈InfiniBand EDR). The same workloads and allocations were used across four networks.



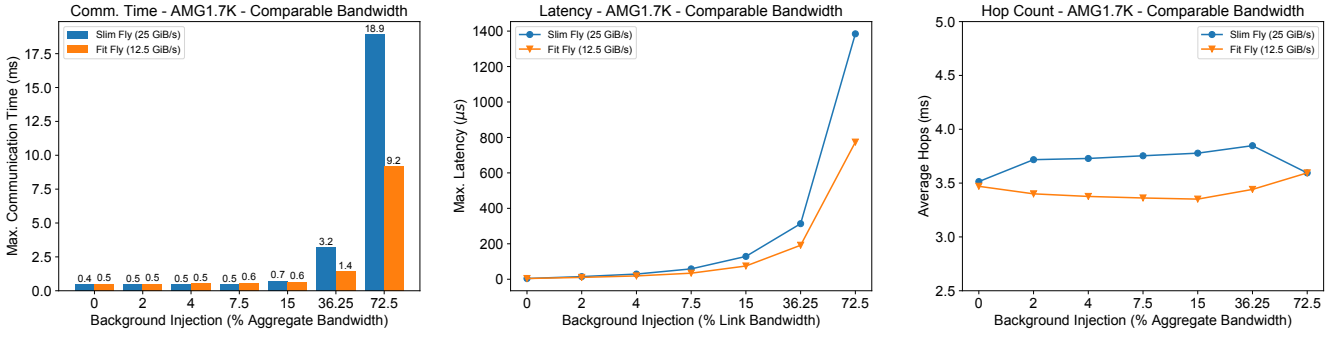
(a) Application Communication Time (b) Latency (c) Hop Count

Figure 6: Synthetic interference experiments on the AMG1728 trace workload with 1,000 synthetic background ranks. Link bandwidth of Slim Fly in this case is 12.5 GiB/s (≈InfiniBand EDR) while Fit Fly is 7 GiB/s (≈InfiniBand FDR). Total aggregate bandwidth is calculated by $B_L \cdot P$, where B_L is the bandwidth of each link in the network.



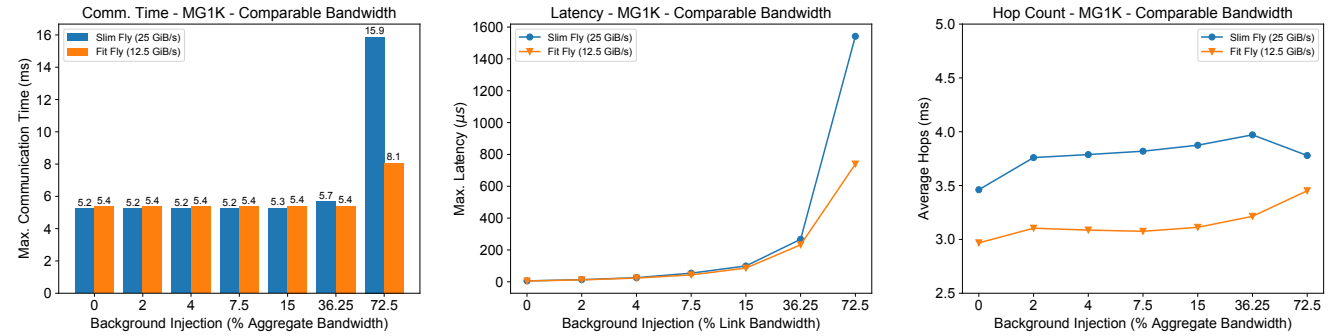
(a) Application Communication Time (b) Latency (c) Hop Count

Figure 7: Synthetic interference experiments on the MultiGrid1000 trace workload with 1,000 synthetic background ranks. Link bandwidth of Slim Fly in this case is 12.5 GiB/s (≈InfiniBand EDR) while Fit Fly is 7 GiB/s (≈InfiniBand FDR). Total aggregate bandwidth is calculated by $B_L \cdot P$, where B_L is the bandwidth of each link in the network.



(a) Application Communication Time (b) Latency (c) Hop Count

Figure 8: Synthetic interference experiments on the AMG1728 trace workload with 1,000 synthetic background ranks. Link bandwidth of Slim Fly in this case is 25 GiB/s (≈InfiniBand HDR) while Fit Fly is 12.5 GiB/s (≈InfiniBand EDR). Total aggregate bandwidth is calculated by $B_L \cdot P$, where B_L is the bandwidth of each link in the network.



(a) Application Communication Time (b) Latency (c) Hop Count

Figure 9: Synthetic interference experiments on the MultiGrid1000 trace workload with 1,000 synthetic background ranks. Link bandwidth of Slim Fly in this case is 25 GiB/s (≈InfiniBand HDR) while Fit Fly is 12.5 GiB/s (≈InfiniBand EDR). Total aggregate bandwidth is calculated by $B_L \cdot P$, where B_L is the bandwidth of each link in the network.

We have performed several experiments in order to answer a "what if?" question. These include comparing the performance of our new implementation with state-of-the-art networks and determining what makes this new model so high-performing.

7.1 Comparing With Other Networks

It is valuable to see how a new idea might compare with old ideas. To do so, we compared Fit Fly with its originating model, Slim Fly, as well as two similarly sized Dragonfly-class networks.

The Dragonfly network, for example, is currently slated as the underlying model for the upcoming Cray Slingshot interconnect [12]. Megafly, or Dragonfly+, is also a contender as the skeleton for future exascale-level HPC systems [9, 25]. The results of these experiments showed that the Slim Fly model itself is competitive with both Dragonfly and Megafly. It matched their performance in both the AMG and MultiGrid trace workloads in low interference studies and outperformed them in higher interference cases. Overall we observed a reduction in maximum communication by to up to 50%.

Fit Fly, by adding an additional plane of independent routers, showed great resilience to high levels of interference. A 72.5% interference traffic run was performed on the Slim Fly and Fit Fly networks to extend Figures 4 and 5 but was not included because the Dragonfly-class networks had such difficulty completing the workloads with that level of interference.

We note that occasionally the Fit Fly average hop count for the baseline 0% background traffic was actually higher than subsequent, more intense, interference runs. This is counterintuitive and is likely a result of an overly aggressive `CONGESTION` planar selection scheme choosing a worse path to avoid small levels of congestion.

Because the Dragonfly-class networks had such difficulty handling the high levels of interference, we stopped the simulations at 1 second (simulation time). Even if we let the simulations complete, it would be difficult to represent the accurate performance metrics while also maintaining the clarity needed to compare Fit Fly with the other networks. For transparency we have included the results of those runs in Table 3.

Table 3: Maximum communication time with synthetic interference injected at 72.5% of link bandwidth (extension of Figures 4 and 5)

Trace Workload	Slim Fly	Fit Fly	Dragonfly	Megafly
AMG1728	28.4 ms	**1.4 ms**	>1 s	>1 s
MultiGrid1000	20.3 ms	**5.4 ms**	>1 s	>1 s

7.2 Comparing with Slim Fly

As noted in Section 6.5, the comparisons made between the Slim Fly and Fit Fly networks following the configurations in Table 2 give Fit Fly a distinct advantage in total bandwidth and router count.

The results of the experiments between Slim Fly and Fit Fly with comparable aggregate bandwidths show that bandwidth alone is not why Fit Fly performs so well compared with its single-rail-single-plane counterpart. Looking at the average number of hops traversed by packets in the networks of Figures 6, 7, 8, and 9, we observe that Fit Fly consistently has a lower hop count than its Slim Fly counterpart has.

With aggregate bandwidth made comparable between the two networks, the only other difference is the total number of routers from the second plane. The higher number of routers mean that there is a lower chance of any two packets sharing a buffer on the same router. Because of the increased number of routers, Fit Fly is able to keep the average number of hops that packets traverse lower, typically, than that of Slim Fly and thus is capable of handling greater levels of traffic even with slower, less expensive links.

We note that the performance of the reduced link bandwidth Fit Fly is far better than that of Slim Fly at high levels of interference in these experiments but is beaten, slightly, by Slim Fly at lower levels of interference. We suspect that this is a result of CODES using link bandwidth for the speed of transmission of data from the workload server and the terminal node instead of the total injection bandwidth.

We also note that at extreme interference workloads, the average hop count sometimes decreases. We believe this is a result of the primary workload finishing and the remaining queued packets from the background traffic working their way through the network without as much contention.

7.3 Cost Comparison

Cost is an important consideration when comparing Fit Fly with the other networks. To give an approximation of how much each of the networks tested in this work would cost to build, we have created a virtual shopping cart of network switches and links using Mellanox's online store [32]. The bandwidths configured for the networks in Table 2 equate to Mellanox IB EDR-class interconnect hardware. Additionally, we want to consider whether the reduced-bandwidth Fit Fly network could be a decent compromise between expense and performance using cheaper Mellanox IB FDR-class hardware.

In Table 4 we give the estimated cost to construct the various networks in this paper—excluding a Slim Fly HDR system since the Mellanox online store had no official quote for that hardware at the time of writing.

Slim Fly, with its lower number of links and routers, wins the "least-expensive" award, followed by Dragonfly, Megafly, and finally Fit Fly. It is unsurprising that Fit Fly loses a flat cost comparison. But even the less expensive Fit Fly FDR system outperformed all the other networks—enough so that even cheaper interconnect hardware could still potentially outperform the state of the art without incurring too much increased expense.

With the strong performance of Fit Fly taken into consideration, the additional costs of Fit Fly (which can be reduced while maintaining good performance by using less expensive hardware) could be worth considering for many HPC system integrators.

7.4 Parallel Simulation Advantages

All experiments in this work were performed by using the ROSS and CODES simulation frameworks. A primary feature of this simulation system is its built-in capability for parallelization. Large simulations stand to benefit greatly from the added processing power to handle the increased number of events.

Table 4: Cost breakdown of the various networks tested in this paper.

	Slim Fly EDR	Fit Fly EDR	Fit Fly FDR	Dragonfly EDR	Megafly EDR
Links	6,253	12,506	12,506	7,524	8,100
Routers	338	676	676	342	360
Cost per Link	$134	$134	$86	$134	$134
Cost per Router	$25,633	$25,633	$19,803	$25,633	$25,633
Total Link Cost	$837,902	$1,675,804	$1,075,516	$1,008,216	$1,085,400
Total Router Cost	$8,663,954	$17,327,908	$13,868,828	$8,766,486	$9,277,880
Total Network Cost	**$9,501,856**	**$19,003,712**	**$14,462,344**	**$9,774,702**	**$10,313,280**

To qualitatively show the type of speedup gained from running simulations in parallel, we performed the simulations in Section 6.4 in both sequential and parallel across 24 processing cores on our 32-core Intel Xeon E5-2640 v3 CPU server at 2.60 GHz with 110 GB of total RAM. Full scaling analysis of ROSS, CODES, and the CODES base Slim Fly model which Fit Fly is iterated from have been performed in previous works [4, 22, 27, 33, 34] and was not performed for this work due to time and space constraints.

The results of this comparison are shown in Figure 10. Optimistic execution granted great speedup (≈7x) in these simulations, which allows for rapid feedback. The longest-running simulation overall was the 72.5% interference run on the 25 GiB/s configured Slim Fly network in Figure 8a, which came in at 99,478 seconds in optimistic mode. For the sake of time, a sequential run was not attempted for comparison.

Parallel simulation even has an advantage over running multiple sequential simulations at once in the form of reduced overall memory consumption. A single parallel simulation largely needs only to allocate extra memory for event buffers for each PE while multiple sequential simulations reinstantiate the entire simulation for each process.

ROSS+CODES is flexible and allows researchers to choose the execution mode that best suits their needs.

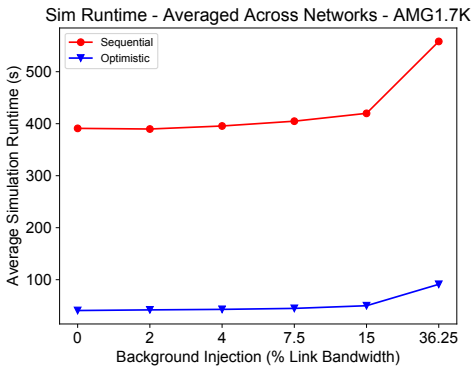


Figure 10: Simulation runtime comparison between sequential and optimistic (parallel) execution. Times are averaged across the four networks tested in Section 6.4. Sequential simulations are executed on a single PE while the optimistic simulations were spread across 24 PEs.

8 CONCLUSION

To summarize, we present Fit Fly, an enhanced Slim Fly simulation model that enables use of multiple independent router planes and multiple rails for packet injection. To facilitate proper utilization of these added features, we have included three schemes for dictating which rail and consequently which plane any given packet will be injected on.

We include a method to increase overall path diversity of the Fit Fly network by providing additional rules for how terminal nodes are connected to routers on each plane.

The new network model was validated against the previously validated Slim Fly network model by injecting varying levels of traffic into the Fit Fly network. We observe that the accepted load of the network scales with the total number of rails and planes.

We performed multiple experiments comparing four similarly sized networks with one another with two HPC application traces with varying levels of background interference traffic. In these experiments the strong behavior of Slim Fly and Fit Fly models was noted with a distinct lead by the Fit Fly model.

Arguably the comparison of the Fit Fly model may not have been completely fair because the total bandwidth of Fit Fly was double that of Slim Fly. To account for this situation, we performed experiments with Slim Fly and Fit Fly models where the total bandwidth of each network was made comparable. Even with reduced Fit Fly bandwidth or increased Slim Fly bandwidth, the Fit Fly model had greater resilience to high levels of interference traffic. This resilience is observed in application performance and average packet latency, as well as the average number of hops traversed by packets in the network, indicating that the network did not have to divert packets on longer paths to avoid congestion hot spots.

The additional power of Fit Fly does come at extra cost, however. Nevertheless, we have shown that the increased network resilience may be worth the cost of the additional router planes and that increased power can be achieved through utilizing less expensive lower bandwidth links.

We have shown that the CODES+ROSS framework can be a valuable tool to test new ideas such as routing algorithms, load-balancing schemes, and other prototypical features. Insight can be gained by creating new or enhancing existing network models in the CODES model lineup.

In future work, we would like to experiment with other ways to utilize additional rails and planes, such as a quality of service implementation at the planar selection level. Another interesting

venture would be to implement these same multi-rail-multi-plane enhancements in other network models.

Testing new concepts and gaining real-world-scale predictions of how they may perform are crucial to continuing innovation in the field of interconnection networks. As congestion avoidance mechanisms grow in complexity, accurately predicting network behavior via mathematical analysis will become more challenging. High-fidelity parallel simulation frameworks such as CODES give researchers the tools necessary to model, test, and simulate concepts at full scale in an efficient manner.

ACKNOWLEDGMENTS

This work was supported by the U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research, under contract number DE-AC02-06CH11357.

REFERENCES

[1] [n. d.]. CODES: Enabling co-design of multilayer exascale storage architectures.
[2] Kishwar Ahmed, Jason Liu, Abdel-Hameed Badawy, and Stephan Eidenbenz. 2017. A brief history of HPC simulation and future challenges. In *Proceedings of the 2017 Winter Simulation Conference*. IEEE Press, 27.
[3] Maciej Besta and Torsten Hoefler. 2014. Slim fly: A cost effective low-diameter network topology. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE Press, 348–359.
[4] Christopher D Carothers, David Bauer, and Shawn Pearce. 2002. ROSS: A high-performance, low-memory, modular Time Warp system. *J. Parallel and Distrib. Comput.* 62, 11 (2002), 1648–1669.
[5] Christopher D Carothers, Kalyan S Perumalla, and Richard M Fujimoto. 1999. Efficient optimistic parallel simulations using reverse computation. *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 9, 3 (1999), 224–253.
[6] Nilesh Choudhury, Yogesh Mehta, Terry L Wilmarth, Eric J Bohm, and Laxmikant V Kalé. 2005. Scaling an optimistic parallel simulation of large-scale interconnection networks. In *Proceedings of the 37th conference on Winter simulation*. Winter Simulation Conference, 591–600.
[7] US DOE. 2016. Characterization of the DOE Mini-apps. http://portal.nersc.gov/project/CAL/designforward.htm [Online; posted 14-July-2016].
[8] Greg Faanes, Abdulla Bataineh, Duncan Roweth, Edwin Froese, Bob Alverson, Tim Johnson, Joe Kopnick, Mike Higgins, James Reinhard, et al. 2012. Cray Cascade: a scalable HPC system based on a Dragonfly network. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*. IEEE Computer Society Press, 103.
[9] Mario Flajslik, Eric Borch, and Mike A Parker. 2018. Megafly: A Topology for Exascale Systems. In *International Conference on High Performance Computing*. Springer, 289–310.
[10] Center for Computational Sciences and Engineering Lawrence Berkeley National Laboratory. [n. d.]. BoxLib Codes. https://boxlib-codes.github.io/ [Online; retrieved 14-March-2019].
[11] Marina Garcia, Enrique Vallejo, Ramon Beivide, Miguel Odriozola, Cristobal Camarero, Mateo Valero, Jesús Labarta, Cyriel Minkenberg, et al. 2012. On-the-fly adaptive routing in high-radix hierarchical networks. In *2012 41st International Conference on Parallel Processing*. IEEE, 279–288.
[12] Cray Inc. 2018. Meet Slingshot: An Innovative Interconnect for the Next Generation of Supercomputers. [Online; posted 30-October-2018].
[13] Nikhil Jain, Abhinav Bhatele, Louis H Howell, David Böhme, Ian Karlin, Edgar A León, Misbah Mubarak, Noah Wolfe, Todd Gamblin, and Matthew L Leininger. 2017. Predicting the performance impact of different fat-tree configurations. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. ACM, 50.
[14] Nikhil Jain, Abhinav Bhatele, Sam White, Todd Gamblin, and Laxmikant V Kale. 2016. Evaluating HPC networks via simulation of parallel workloads. In *SC'16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 154–165.
[15] David R Jefferson. 1985. Virtual time. *ACM Transactions on Programming Languages and Systems (TOPLAS)* 7, 3 (1985), 404–425.
[16] Nan Jiang, Daniel U Becker, George Michelogiannakis, James Balfour, Brian Towles, David E Shaw, John Kim, and William J Dally. 2013. A detailed and flexible cycle-accurate network-on-chip simulator. In *2013 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. IEEE, 86–96.
[17] Georgios Kathareios, Cyriel Minkenberg, Bogdan Prisacari, German Rodriguez, and Torsten Hoefler. 2015. Cost-effective diameter-two topologies: Analysis and evaluation. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. ACM, 36.
[18] John Kim, Wiliam J Dally, Steve Scott, and Dennis Abts. 2008. Technology-driven, highly-scalable dragonfly topology. In *2008 International Symposium on Computer Architecture*. IEEE, 77–88.
[19] Brendan D McKay, Mirka Miller, and Jozef Širáň. 1998. A note on large graphs of diameter two and given maximum degree. *Journal of Combinatorial Theory, Series B* 74, 1 (1998), 110–118.
[20] Mirka Miller and Jozef Sirán. 2005. Moore graphs and beyond: A survey of the degree/diameter problem. *The electronic journal of combinatorics* 1000 (2005), DS14–Dec.
[21] Misbah Mubarak, Christopher D Carothers, Robert Ross, and Philip Carns. 2012. Modeling a million-node dragonfly network using massively parallel discrete-event simulation. In *2012 SC Companion: High Performance Computing, Networking Storage and Analysis*. IEEE, 366–376.
[22] Misbah Mubarak, Christopher D Carothers, Robert B Ross, and Philip Carns. 2014. A case study in using massively parallel simulation for extreme-scale torus network codesign. In *Proceedings of the 2nd ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*. ACM, 27–38.
[23] Misbah Mubarak, Christopher D Carothers, Robert B Ross, and Philip Carns. 2014. Using massively parallel simulation for MPI collective communication modeling in extreme-scale networks. In *Proceedings of the Winter Simulation Conference 2014*. IEEE, 3107–3118.
[24] Misbah Mubarak, Christopher D Carothers, Robert B Ross, and Philip Carns. 2017. Enabling parallel simulation of large-scale HPC network systems. *IEEE Transactions on Parallel and Distributed Systems* 28, 1 (2017), 87–100.
[25] Misbah Mubarak, Neil McGlohon, Malek Musleh, Eric Borch, Robert B Ross, Ram Huggahalli, Sudheer Chunduri, Scott Parker, Christopher D Carothers, and Kalyan Kumaran. 2019. Evaluating Quality of Service Traffic Classes on the Megafly Network. In *International Conference on High Performance Computing*. [To Appear].
[26] Fco Javier Ridruejo Perez and José Miguel-Alonso. 2005. INSEE: An interconnection network simulation and evaluation environment. In *European Conference on Parallel Processing*. Springer, 1014–1023.
[27] Caitlin J Ross, Christopher D Carothers, Misbah Mubarak, Robert B Ross, Jianping Kelvin Li, and Kwan-Liu Ma. 2018. Leveraging shared memory in the ross time warp simulator for complex network simulations. In *2018 Winter Simulation Conference (WSC)*. IEEE, 3837–3848.
[28] Alexander Shpiner, Zachy Haramaty, Saar Eliad, Vladimir Zdornov, Barak Gafni, and Eitan Zahavi. 2017. Dragonfly+: Low cost topology for scaling datacenters. In *2017 IEEE 3rd International Workshop on High-Performance Interconnection Networks in the Exascale and Big-Data Era (HiPINEB)*. IEEE, 1–8.
[29] Arjun Singh. 2005. *Load-balanced routing in interconnection networks*. Ph.D. Dissertation. Stanford University.
[30] Shane Snyder, Philip Carns, Jonathan Jenkins, Kevin Harms, Robert Ross, Misbah Mubarak, and Christopher Carothers. 2014. A case for epidemic fault detection and group membership in HPC storage systems. In *International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems*. Springer, 237–248.
[31] Sstsimulator. 2018. sstsimulator/sst-dumpi. https://github.com/sstsimulator/sst-dumpi
[32] Mellanox Technologies. 2019. The Official Store. https://store.mellanox.com/ [Online; retrieved 14-March-2019].
[33] Noah Wolfe, Christopher D Carothers, Misbah Mubarak, Robert Ross, and Philip Carns. 2016. Modeling a million-node Slim Fly network using parallel discrete-event simulation. In *Proceedings of the 2016 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*. ACM, 189–199.
[34] Noah Wolfe, Misbah Mubarak, Christopher D Carothers, Robert B Ross, and Philip H Carns. 2018. Modeling large-scale slim fly networks using parallel discrete-event simulation. *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 28, 4 (2018), 29.
[35] Jongmin Won, Gwangsun Kim, John Kim, Ted Jiang, Mike Parker, and Steve Scott. 2015. Overcoming far-end congestion in large-scale networks. In *2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 415–427.