Defining the Boundaries for Endpoint Congestion Management in Networks for High-Performance Computing

Daniel Postigo
daniel.postigo@unican.es
Universidad de Cantabria
Santander, Cantabria, España

David Herreros
Universidad de Cantabria
Santander, Cantabria, España

Eloy Barón
Universidad de Cantabria
Santander, Cantabria, España

Cristóbal Camarero
camareroc@unican.es
Universidad de Cantabria
Santander, Cantabria, España

Pablo Fuentes
fuentesp@unican.es
Universidad de Cantabria
Santander, Cantabria, España

ABSTRACT

A hotspot traffic pattern of communications can be a common phenomenon in HPC topologies that causes significant and lasting network performance degradation. This performance deterioration remains persistent over time, intensifying its impact even after the cessation of the detrimental traffic injection into the network. To understand its causes and effects, we analyze the network behavior under different hotspot traffic scenarios and compare the performance on various topologies. We examine both the *performance drop* due to traffic flows with endpoint contention, and the *recovery process* of the network after this phenomenon has occurred, if swift action is taken to mitigate it.

Our results show that some topologies are more resilient to hotspot traffic than others, both to reduce the performance drop and/or to accelerate the recovery process. In particular, Flattened Butterfly is more resilient to congestion and consistently demonstrates a rapid recovery. The results of the analysis reinforce the need for mechanisms with effective and expeditious action to reduce the magnitude and duration of the performance drop. Furthermore, they highlight behavioral differences between topologies that can affect the effectiveness of mechanisms using congestion-based metrics.

CCS CONCEPTS

• **Networks** → **Data center networks**; **Network performance analysis**; **Network simulations**; **End nodes**.

KEYWORDS

High-Performance Interconnection Networks, Network Congestion, Hotspot Pattern, Endpoint Congestion

ACM Reference Format:
Daniel Postigo, David Herreros, Eloy Barón, Cristóbal Camarero, and Pablo Fuentes. 2024. Defining the Boundaries for Endpoint Congestion Management in Networks for High-Performance Computing. In *The Seventh International Workshop on Systems and Network Telemetry and Analytics (SNTA '24), June 3–7, 2024, Pisa, Italy.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3660320.3660333

This work is licensed under a Creative Commons Attribution International 4.0 License.
SNTA '24, June 3–7, 2024, Pisa, Italy
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0648-6/24/06
https://doi.org/10.1145/3660320.3660333

1 INTRODUCTION

This paper analyzes the impact of endpoint congestion in several network topologies through a hotspot traffic pattern. Different network topologies have been defined with the aim of promoting high scalability, enabling large system sizes, such as those employed in *High-Performance Computing* (HPC) systems listed in the TOP500[1]. Such topologies include Megafly [9, 24], Dragonfly [7, 16], and Flattened Butterfly [12, 15]. Other classical networks that have widespread use include Fat-tree [17, 22] and Tori [3, 6].

Fat-tree is a well-known indirect hierarchical topology that provides a high bisection bandwidth but incurs in significant cost to scale to large sizes, when compared to other proposals. Megafly [9], also known as Dragonfly+ [5, 20] is an indirect hierarchical topology with all-to-all connectivity at the highest level and a Fat-tree topology inside the lowest hierarchical level. Flattened Butterfly, also known as HyperX [2] or Hamming Graphs, is a cost-efficient direct topology for high-radix networks. Dragonfly is a hierarchical direct topology that follows a two-level distribution, with all-to-all connectivity within each level. This work focuses solely on lossless networks, where packets are not dropped in the event of congestion or contention. Lossy networks are sometimes employed in HPC, but there is a significant emphasis on minimizing packet loss.

Lossless networks are prone to performance drops if links become saturated, since packets cannot be dropped. When congestion arises, it typically gives rise to *Head-of-Line Blocking* (HoLB) [14]. HoLB is prone to occur at the switch where the root of a *congestion tree* is situated. In this scenario, a packet at the head of an input queue becomes stuck while attempting to access the root, thereby obstructing the packets queued behind it, even if they are seeking other output ports within the same switch. This situation is referred to as low-order HoLB [14]. Furthermore, there is the possibility of experiencing high-order HoLB [13] when congestion spreads to switches other than the one where it originated. In such cases, the *congestion tree* branches may block non-congested packets, also affecting flows not contributing to the congestion, severely degrading the network performance. [18]

The focus of this work is analyzing the impact of network congestion when the endpoint is stressed due to concurrent incoming

traffic flows. One example of such behavior is hotspot traffic, where an endpoint receives multiple concurrent streams from different sources, limiting the maximum rate at which each stream can be consumed. When those streams are not capped to ensure the aggregated traffic load does not exceed the endpoint consumption capabilities, hotspot traffic causes a high concentration of traffic in a given network area (a *hotspot*). This can lead to an overall performance degradation of the network, affecting streams from other applications that do not exhibit this behavior.

Congestion trees are often used to model the propagation of congestion in a network [11]. These trees can be used to identify the source of congestion and to develop congestion control mechanisms to alleviate the problem. The *congestion tree* grows as buffers fill up through the switches when the switches run out of credits.

The paper examines the impact of hotspot traffic on network performance in both stationary and transient scenarios. In the stationary scenario, we evaluate the natural performance degradation of the network when no congestion management mechanism is employed. In the transient scenario, we investigate the network performance degradation and subsequent recovery, when possible, in the event of a burst of hotspot traffic.

Section 2, delves into the phenomenon of endpoint congestion and its repercussions on network performance. Section 3 details the methodology for the analysis and experiments. Section 4 presents the findings observed across both steady-state and transient scenarios. Lastly, Section 5 presents the conclusions of the study.

2 ENDPOINT CONGESTION PHENOMENON AND IMPACT ON THE NETWORK

In this study, we assess the variation in the impact of endpoint congestion on the overall network throughput for different topologies.

2.1 Hotspot Traffic Pattern

Hotspot traffic patterns are characterized by a high concentration of traffic in a specific area of the network and can be commonly observed in HPC networks [19]. These patterns can lead to congestion and other performance issues, particularly at end nodes (i.e., endpoints). This pattern is considered particularly adverse because the performance losses it generates can be significant if preventive measures are not implemented.

This network scenario is prone to manifest during *many-to-one* communication operations, where multiple servers send traffic to a common destination server, leading to incast congestion. Furthermore, the utilization of shared network resources, including routers, buffers, virtual channels (VCs), and links, by different traffic flows can contribute to network congestion. This is primarily due to routes overlapping, whether minimal or not, between these flows.

2.2 Congestion Control Mechanisms

Congestion avoidance or mitigation can be achieved through congestion control mechanisms. Existing congestion control mechanisms in interconnects can be divided into two general approaches. One is to throttle traffic injection at the sources that contribute to congestion, and the other is to isolate the congested traffic in specially designated resources[8]. Other elements that can affect the impact of congestion on the network include flow control, which regulates the flow of packets between nodes, and adaptative routing algorithms that take congestion into account when selecting a path for forwarding packets.

In this paper we focus on injection throttling, which is the only suitable policy to tackle endpoint congestion in lossless networks. This mechanism mitigates congestion backspreading into the network and avoids exacerbating the problem, since it tries to ensure that the network is not overwhelmed with traffic. For networks employing non-minimal routing, however, confining traffic to a region of consecutive nodes does not significantly alleviate degradation. This is because the shortest paths of flows with congested endpoints quickly become saturated, spreading overflow onto other routes that will also experience a similar performance decline.

The theoretical limit applied to prevent packets from overflowing into other routes depends on the contention factor at the endpoint (the number of servers whose flows share the same destination).

2.3 Topologies Employed

The analysis of the phenomenon is covered for direct and indirect networks. Direct networks are those in which the network switches are directly connected to at least one compute node. For the analysis in this work, direct networks include the Dragonfly [16] and Flattened Butterfly [15]. The study also assesses the behavior in two indirect networks, the Megafly [9] and the Fat-tree [17], where some network switches are solely connected to other switches and not directly to endpoints.

The Flattened Butterfly [15], also known as HyperX [2], that has been used is a fully symmetric direct topology in which all switches have both compute nodes and other switches connected. It is a cost-efficient architecture employed in high-radix networks, particularly in data centers and supercomputers. The design incorporates concentrated high-radix routers and a globally load-balancing routing algorithm to reduce the overall number of links or cables [15].

The Dragonfly is a hierarchical direct topology, where all its switches have computing nodes and other switches connected. This work employs the canonical Dragonfly, in which both intra-group and inter-group connections are fully meshed, forming a complete graph. The Megafly, also known as Dragonfly+ [5, 20], is a hierarchical indirect topology characterized by high path diversity.

The Fat-tree network topology is a specialized architecture employed in data centers due to its scalability and efficiency. It represents a variant of the tree topology, featuring progressively higher bandwidth as one moves towards the root of the tree. This design effectively minimizes bottlenecks, enabling high-performance parallel and distributed computing. Notably, it operates as a non-blocking network with a wide range of minimal paths, promoting route separation and offering mitigation capabilities against network congestion phenomena. The Fat-tree topology proves particularly well-suited for networks with varying amounts of data transmitted between devices.

2.4 Impact of the Routing Mechanism

In most of the examined topologies, the implementation of non-minimal adaptive routing proves essential to address adverse traffic patterns. Non-minimal adaptive routing involves the selection

Table 1: Description of the networks analyzed in this article.

Network	Dimensions or height	Switch radix	Diameter (# hops)	# switches	# servers (endpoints)
Canonical Dragonfly	1	8	3	264	1056
Flattened Butterfly	3	21	3	216	1296
Fat-tree	2	16	4	320	1024
Megafly	1	12	3	444	1332

between one or more minimal paths and one or more longer non-minimal paths. Choosing a non-minimal path can enhance the bandwidth between endpoints, reducing the chance of hotspots along the non-minimal path. However, it is crucial to note that this approach may lead to performance degradation in cases where congestion stems not from the network itself (*in-network congestion*) but from the destination (*endpoint congestion*), because it allows the congested traffic flows to spread into regions of the network unaffected by the congested streams.

It should be noted that this type of routing does not apply to Fat-tree networks. In such networks the diversity of minimal paths favors the implementation of a policy for minimal routes when communicating between servers, without any loss in performance. Since it is a non-blocking network with high path diversity, there is no need to resort to non-minimal routes unless a specific case of adverse traffic requires it.

However, in Fat-tree networks, *up/down routing* is often employed. This method involves ascending through the levels of the tree hierarchy until a common switch with the path to the destination is found. At that point, a descent begins to approach the destination. The routing behavior is significantly impacted by the policy that determines which of the available paths to take, given the existence of multiple minimal paths.

Under endpoint congestion, the optimal strategy is to minimize the overlap of routes between flows with endpoint congestion and the rest of communications across the network.. Utilizing all possible routes enhances network performance for other scenarios as it exploits the diversity of available paths.

Non-minimal adaptive source routing is employed in the other topologies analyzed in this paper. Adaptive routing combines the benefits of non-minimal routing with the flexibility to use non-minimal routes based on the current state of the network. This approach involves decision-making at the source switch, leveraging network state information available at that node.

Since the work focuses on lossless networks, a deadlock avoidance mechanism is required to prevent the network from stalling if a cycle between packets can occur. All evaluations have been conducted with a virtual channel-based deadlock avoidance mechanism, where cycles are broken by assigning different legs of the network path to different resources (*buffers*) in the input ports of the routers. This may result in certain topologies using more resources than others, which could affect the observed behavior when traffic flows with endpoint congestion are present.

3 SIMULATION METHODOLOGY

In order to ensure repeatability and achieve a fair yet comprehensive comparison between different network topologies, a simulation-based approach through synthetic traffic models has been employed for the analysis. This section introduces the simulation tool used to conduct the experiments, outlines the characteristics of the employed topologies, details the routing strategies used, and provides insights into the utilized workloads.

3.1 Network Simulator

The experiments are conducted using CAMINOS [4], a network simulator written in the Rust language [23] that models at the microarchitecture level of the router. CAMINOS is an event-driven simulator that operates at the flit level. The results were generated and presented using the same simulator. The simulator was used both to run the experiments and to generate and present the results. For all experiments, each data point comprises the results of three separate simulations.

3.2 Analyzed Topologies

The evaluation was performed on four different topologies, as outlined in Section 2.3, each with around 1,000 compute nodes. This network size was selected to achieve non-trivial results without incurring significant delays in the experimentation phase. Table 1 presents the network information for each topology analyzed, while Table 2 shows the parameters that are common to all simulated topologies, including the router microarchitecture.

3.2.1 Flattened Butterfly. This analysis utilizes a balanced 3-D Flattened Butterfly network with a side length of 6. Each switch in the network features 21 ports, with 6 ports specifically designated for server connections. The network has a diameter of 3, with 216 switches and a total of 1296 endpoints (i.e., servers) in this configuration, and six virtual channels are employed.

3.2.2 Dragonfly. This system is made up of 33 groups of 8 switches in each group. All 264 switches used in this setup have a radix of 8, with 4 ports allocated for connecting servers and 4 for establishing global links, facilitating inter-group communication. The network accommodates a total of 1056 servers. With a diameter of 3, the distribution and connection of global links follows the *PalmTree* arrangement, as specified in [10], and four virtual channels are employed.

3.2.3 Megafly. This configuration features 37 groups, each housing 12 *radix-12* switches. Within each group, half of the switches are designated as *leaf-switches*, while the remaining half serve as *spine-switches*. The *leaf-switches* dedicate half of their links to connect with servers, while the *spine-switches* allocate half of their links as global ports for intergroup connections. The network exhibits a diameter of 3, connecting a total of 1332 endpoints. The distribution and connection of global links follows the *PalmTree* arrangement, as detailed in [10], and two virtual channels are employed.

3.2.4 Fat-tree. This configuration utilizes a Fat-tree network with a height of 2 and 3 levels of switches. It consists of 320 *radix-16* switches. The lower level, known as the leaf, has 128 switches, each connected to 8 servers, resulting in a total of 1024 servers or endpoints. The diameter in this setup is 4, and two virtual channels are employed.

Table 2: Common parameters to the analysis performed.

Parameter	Value(s)	Units / Explaination
Router architecture	Input-Output Queueing (IOQ)	Router with queues (buffers) at the inputs and outputs.
Input VC size	256	Capacity of each virtual channel, in phits, at the router's inputs.
Output VC size	256	Capacity of each virtual channel, in phits, at the router's outputs.
Flit size	8	Credits required in the next router's virtual port to begin the transmisión.
Message size	8	The size in phits of the messages created by the servers.
Maximum packet size	8	Messages of greater length will be broken into several packets.

3.3 Routing

Adaptive source routing was used for all analyzed networks, except for the Fat-tree topology where deterministic minimal routing, specifically *up/down routing*, was utilized. *Up/down routing* is a well-established algorithm known for its simplicity and effectiveness in Fat-tree topologies.

In hierarchical topologies, such as Megafly, Dragonfly, and Flattened Butterfly, a source adaptive routing mechanism has been employed to address the limitations of minimal routing discussed in Section 2. Specifically, we utilize a variant of the *Universal Globally Adaptive Load-balance* (UGAL) routing, as detailed by Kim et al. in [16]. The weight function used in our implementation is based on Arjun Singh's research [21].

This function assesses the candidate queue occupancy (i.e., consumed credits) and the estimated distance (i.e., hop count) to the destination. The function uses the product of both values as the metric for the misrouting decision, promoting efficient routing decisions that ensure a fast delivery.

Switches measure their own credits (decrease in packets sent), rather than relying on the occupancy reported by the neighboring switch, to determine occupancy for the misrouting decision. This occupancy metric is achieved by adding up all of the port's buffers, rather than just the *virtual channel* (VC) requested for the next hop. Furthermore, to prevent switching between minimal and non-minimal routes under low stress loads, a constant threshold is used. This ensures that the comparison between occupancy levels remains meaningful even under low queue occupancy.

Table 3: Maximum theoretical load accepted, under uniform random traffic, for the four networks analyzed. Also shown is the size (in servers) of the hotspot (HS) region.

Network	Size of HS region (servers)	HS region configuration	Max acc. load
Canonical Dragonfly	105	1 group	0.0090
		7 groups	0.0714
Flattened Butterfly	126	1 group	0.0080
		7 groups	0.0588
Fat-tree	105	1 group	0.0090
		7 groups	0.0714
Megafly	133	1 group	0.0075
		7 groups	0.0555

3.4 Traffic Workload

A workload consists of traffic that follows a specific pattern (spatial) and has a likelihood of generating (temporal) a packet on individual servers. In this analysis, all scenarios use synthetic traffic generated through a Bernoulli process with a uniform temporal distribution.

The *Uniform Random* (UR) traffic pattern is generally considered benign in most networks. This type of traffic distributes the load randomly among all servers (i.e., computing nodes) in the network. Each source server selects a destination randomly, following a uniform distribution, which allows the load to balance itself naturally. Therefore, when analyzing the impact of adverse patterns on network performance, this pattern provides a good baseline.

The network has been partitioned into two regions or sets of servers, allowing different flows to coexist in the network. On 90% of the servers, UR traffic pattern is generated, while the remaining 10% will follow a *Hotspot* (HS) traffic pattern, referred to in Section 2.1. The aim of this distribution is to analyze the performance of the nodes that are generating homogeneous traffic, which ideally should not be affected by the performance limits of the traffic flows with endpoint congestion.

The two regions are disjoint, with each server only generating and receiving a single type of traffic pattern. However, there may be overlap in the resources shared in the routes of the packets for both patterns, such as buffers, links, or routers ingresses and egresses. The regions are interleaved instead of confining them in contiguous servers to further stress the UR traffic flows and represent a worst-case scenario for the analysis. This is usual on the Dragonfly/Megafly for applications that do not fit in a group.

The pattern has been implemented in two configurations within the HS region. In the first one, a server is randomly selected in a single HS region to receive all the traffic, resulting in a single congestion tree rooted at this designated server. The second configuration is based on splitting the HS region into 7 subregions, designating a destination in each of them. Therefore, there are 7 congestion trees, each with its root in the designated node of each subregion. In this second configuration, the proportion of congestion at the destination node is 1/7 compared to the first.

The maximum theoretical accepted load of the HS destination(s) is given by the number of servers that send them traffic. These theoretical values have been calculated for both the case of a single HS region and for 7 regions. The number depends on the factor, which, for the indicated network sizes, is provided in Table 3. These values have been used as a reference to select the loads applied in HS traffic when nodes have a reduced injection rate.

4 SIMULATION RESULTS

Section 4.1 performs a steady-state analysis using a traffic workload that includes UR traffic load and a combination of HS + UR. Section 4.2 presents a transient analysis of traffic in two and three phases, respectively, to evaluate network behavior when an adverse traffic pattern, such as a HS, is introduced. The objective of this study is to investigate whether network recovery can be achieved by limiting node injection in the HS region. An analysis of the drop rate when the traffic transitions from UR traffic to the combination of HS and UR has been performed but is omitted due to space constraints.

4.1 Behavior in Stationary State

The network's performance is evaluated under steady-state conditions by sweeping the offered load from 0 to 1 *phit per node and cycle* in increments of 0.1 phits/node/cycle for each analyzed topology. After warming up for several cycles, the accepted load, also known as throughput, is measured on the servers. These results are used as a *baseline*.

4.1.1 Uniform Random. Figure 1 displays the performance of the four topologies under uniform random traffic. Error bars are not shown as the difference observed between seeds is negligible.

Both Flattened Butterfly (*blue square*) and Fat-tree (*black star*) topologies achieve nearly the maximum theoretical performance, with an accepted load of 0.98 and 0.97 phits/node/cycle, respectively. Megafly (*teal triangles*) reaches an accepted load of 0.90 phits/node/cycle, slightly above the 0.82 of the Dragonfly network (*red circle*). This may be because they have a greater diversity of shortest paths, which improves performance when small hotspots occur in the network. UR traffic is not completely homogeneous when considering specific moments in time and regions, which can lead to small contention phenomena.

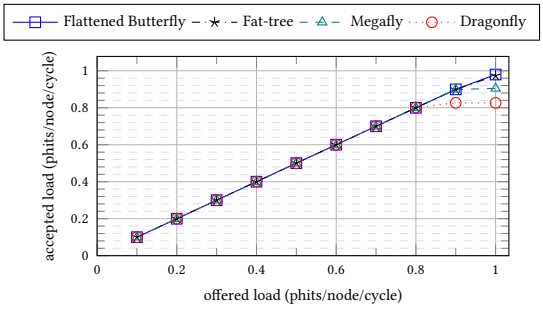

Figure 1: The accepted load (throughput) as a function of the offered load, under UR traffic, for the four topologies analyzed.

4.1.2 Uniform Random + Hotspot. Figure 2 shows the network performance when the network is divided into the two disjoint regions with UR and HS traffic patterns described in Section 3.4. Results discriminate between applying the same traffic load to all nodes in the network, and the use of injection throttling for the nodes belonging to the HS region. The *X-axis* of the figures shows the load injected (i.e., offered) by all nodes in the network when the injection constraint is not applied.

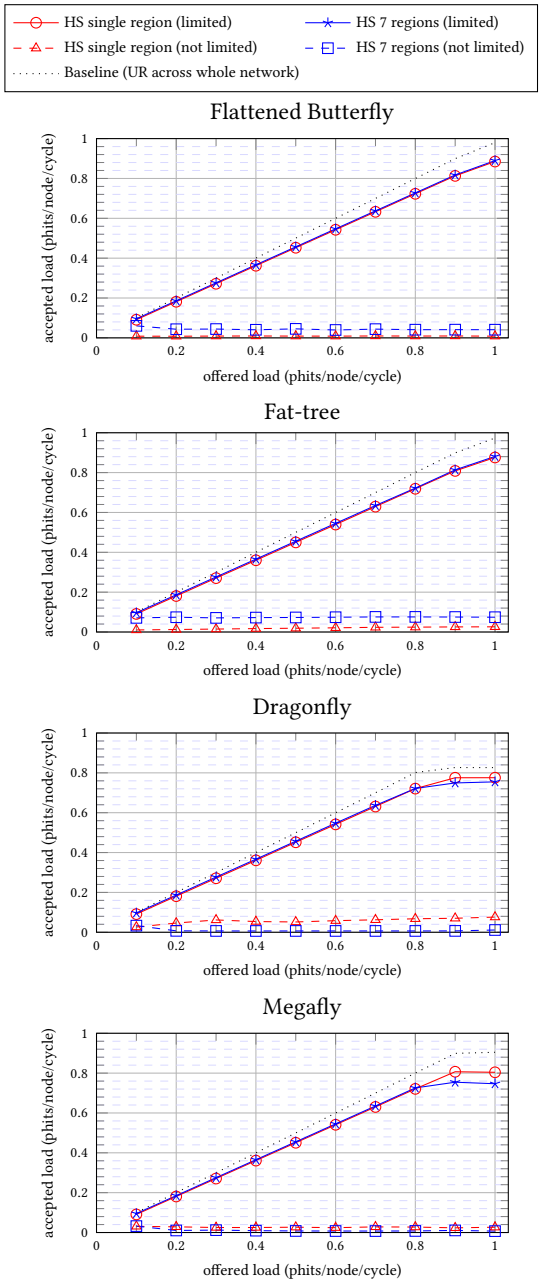

Figure 2: Results of the stationary analysis for the topologies. The throughput is shown as a function of the offered load.

When this constraint is applied, the injected load of nodes in the HS region is limited to values below the theoretical limits presented in Table 3. The results of three different simulations are shown for each value, with error bars excluded when the differences are not significant.

The baseline performance of each topology is represented by a black dashed line without markers. Dashed lines represent values with no injection restriction, while solid lines represent values with

this mechanism active. Red lines indicate that HS traffic corresponds to a large single region, whereas blue lines refer the results for seven HS regions.

Red triangles correspond to a single region injecting traffic at the same rate as the UR traffic streams, while in the red circles the HS stream is reduced below the theoretical maximum load indicated in Table 3. Blue stars correspond to seven HS regions injecting traffic at the same rate as the UR traffic streams, while in the blue squares the HS streams are reduced below the theoretical maximum load indicated in Table 3.

The network performance should ideally be marked by 90% of the nodes that communicate following a UR traffic pattern, with 10% emitting HS traffic. Where the HS traffic origin nodes exceed the theoretical limit, the traffic overflows to non-minimal routes and affects the performance of the entire network when non-minimal adaptative routing is employed. However, this is not observed as the load accepted by the network falls below the theoretical limits shown in Table 3, showing performance degradation beyond the upper bound imposed by the HS traffic flows. Therefore, there is a need for a mechanism that 1) detects that congestion is being generated in the network, and 2) acts precisely on the responsible servers. An ideal mechanism would limit the load injected by the HS region nodes, allowing the rest of the UR nodes to reach acceptable values.

Figure 2 shows how 90% of theoretical maximum is achieved when injection throttling is applied on HS flows, mitigating the impact on other traffic in the network. As demonstrated by the 0.88 phits/node/cycle on the Flattened Butterfly, the 0.87 of the Fat-tree, 0.81 of the Megafly, and 0.77 in the Dragonfly. In the case of the Flattened Butterfly and Fat-tree networks, no significant difference is observed in the performance under stationary traffic between modeling a single large HS region and 7 smaller HS regions. In the case of the Dragonfly and the Megafly at load above 0.8 phits/node/cycle, the throughput is slightly higher when the HS traffic is modeled in a single region compared to 7, although it is true that the differences fall within the error bars.

In theory, a single region has a lower theoretical limit than 7, but there is a higher probability of *route overlap* when multiple HS destinations are used. Therefore, a case that is initially less problematic can lead to a scenario with similar performance limitations, as observed.

4.2 Performance Recovery Rate Analysis

This transient analysis examines the behavior of the network in different phases when a traffic pattern with endpoint congestion enters the network. More concretely, the scenario is structured in 3 phases: an initial phase to *warm-up* the network, a middle phase with the *combination of UR and HS* traffic described in Section 4.1.2 that represents the appearance of adversarial traffic flows in the network, and a third phase where the injection rate of the HS streams is capped below the theoretical limit admitted by the network.

The objective is to assess the network's ability to recover when injection is restricted at HS region nodes after a brief lapse of time. The rationale for using this pattern is to induce the previously mentioned phenomenon of congestion at the final node (*endpoint congestion*) thereby enabling to study its impact on network performance, as well as constitute a reference point for the performance of an optimal injection-throttling mechanism that aims to reduce congestion, restricting the rates of problematic streams while leaving unaffected the remaining communications in the network.

Prior to introducing the traffic pattern, the network undergoes an initial warm-up phase of 20,000 simulation cycles. During this phase, all servers inject UR traffic to ensure queue saturation throughout the network's servers and routers. Once the network is warmed up, the second phase begins, during which 10% of the servers follow a HS traffic pattern. In both phases, each server injects traffic at its full capacity. The second phase intentionally varies the duration to explore the impact of different durations of the HS traffic burst on network recovery. Experiments were conducted with burst durations of 100, 500, and 1,000 simulation cycles.

After completing the second phase, the load injected by HS nodes is restricted to a level below the theoretical limit that the destination server can handle for all HS traffic, as described in Table 3.

After limiting the injection load of these nodes, the simulation covers approximately 140,000 cycles to observe if the network shows any signs of performance recovery. To analyze whether the network could recover in cases where recovery was not achieved for the theoretical limits, tests for lower load caps for HS streams were performed; those tests are not presented here due to space constraints.

Figure 3 shows the outcome when there is a single HS region in the network. Figure 4 displays the results when the HS region is divided into 7 small subregions, each with an equal number of servers. In both scenarios, the selection of the HS common destination within the region is random and dependent on the seed.

Results for the 100-cycle HS bursts are represented with orange dotted lines, the 500-cycle bursts with teal blue continuous lines, and the 1000-cycle bursts with dashed red lines. No markers are added to improve the readability of the figures.

4.2.1 Single hotspot region. Results for a single HS region show a more or less pronounced but clear performance drop when the middle phase with HS traffic burst at maximum load extends for 100 cycles. However, performance rebounds after less than 10,000 cycles for the Flattened Butterfly and Dragonfly topologies. In the case of Megafly and Fat-tree, this recovery takes approximately 20,000 cycles to complete. Afterwards, the network stabilizes to the values observed in the stationary analysis of Section 4.1. It should be noted that while Flattened Butterfly and Dragonfly only suffer small performance losses of 0.2 and 0.3 phits/node/cycle, respectively, Fat-tree and Megafly experience much more pronounced losses in accepted load. In fact, for Fat-tree and Megafly, the losses are so significant that accepted load drops from a traffic load near the maximum consumption rate of the nodes (1 phit/node/cycle) to values close to 0 (Fat-tree) or 0.2 (Megafly). This suggests that the traffic pattern has a greater impact on the performance of these two networks.

With a burst duration of 500 cycles, only Megafly is able to recover to the levels observed in the stationary analysis. Flattened Butterfly remains 0.1 phit/node/cycle below this value. In the case of Fat-tree, the maximum accepted load after the HS burst only reached 0.2 phit/node/cycle, after having peaked at 0.4. For Dragonfly, while

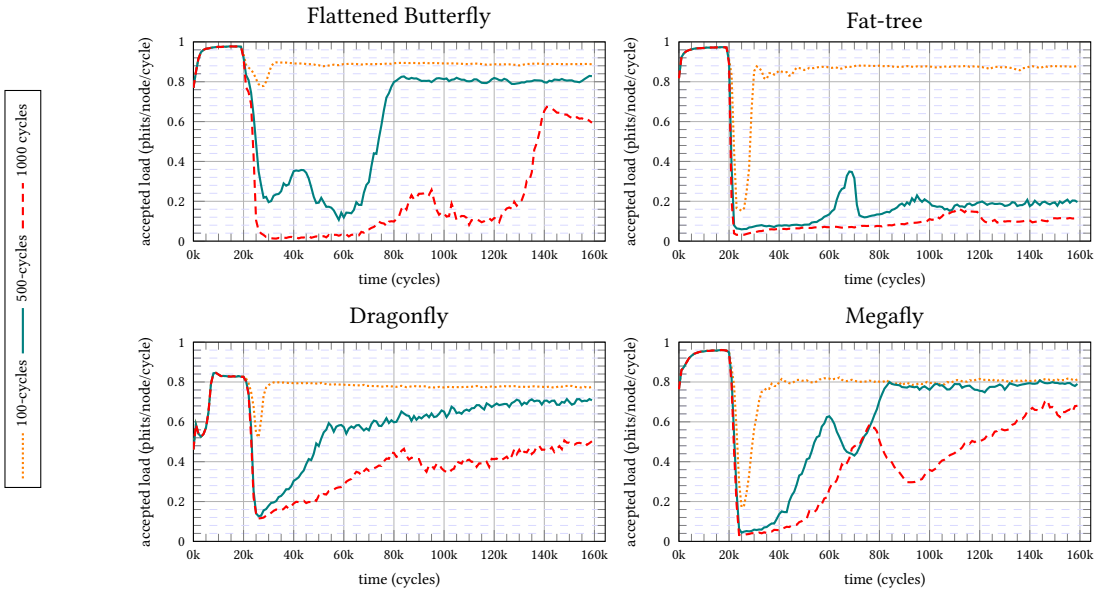

Figure 3: Transient analysis results for single region of HS traffic in the four topologies analyzed. The temporal evolution of the througput during the simulation cycles is plotted. The effect of the HS traffic burst duration is indicated for 3 cycles sets.

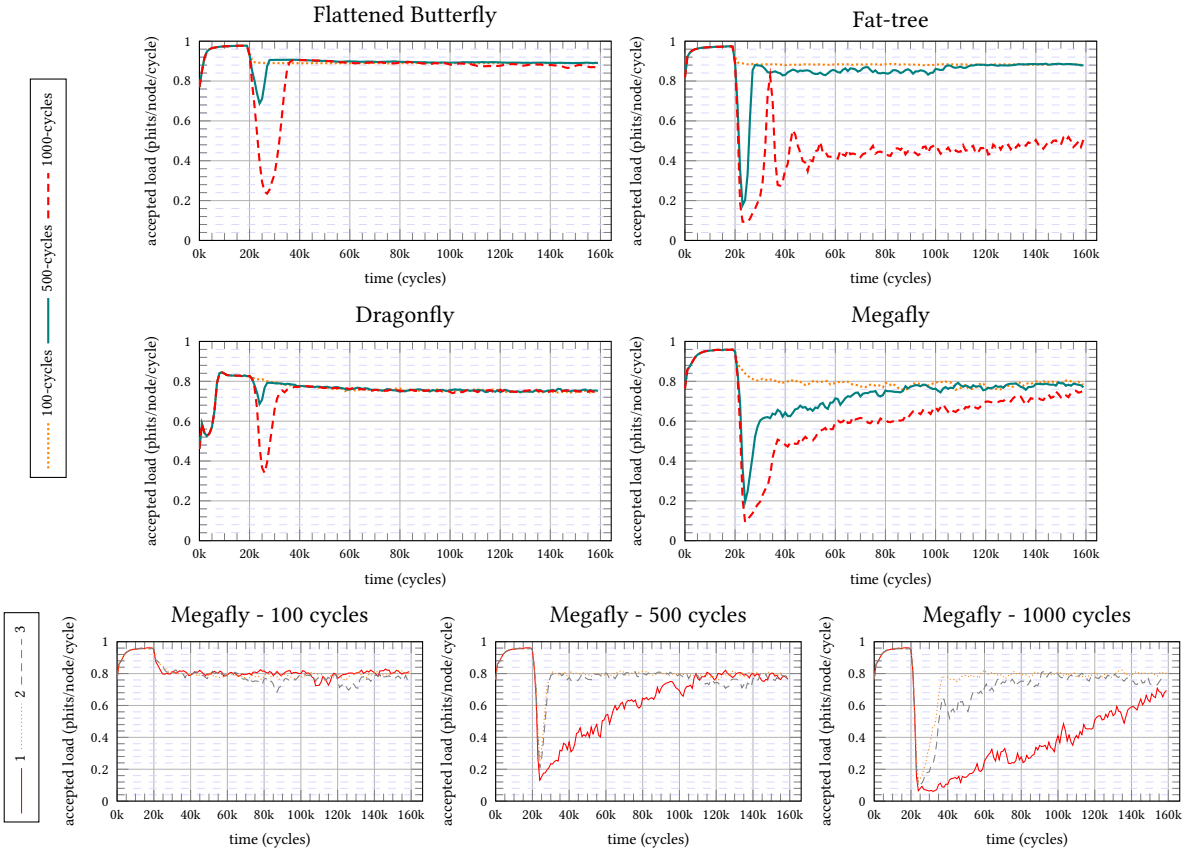

Figure 4: Transient analysis results of the throughput for 7 regions of HS traffic in the 4 topologies analyzed, with different lengths of the traffic burst. For Megafly, results are also split into separate graphs for the different burst length.

a recovery in network performance is observed for some seeds, there can be no assertion that this is the general case.

When analyzing the longest burst of 1,000 cycles of HS traffic, it is evident that none of the networks are able to fully recover from the adverse effects of this type of traffic. However, some networks are able to partially recover their capacity. For instance, Flattened Butterfly stabilizes around 0.6 phit/node/cycle (out of a target of 0.9), Megafly around 0.7 (out of a target of 0.8), and Dragonfly around 0.5 (out of a target of 0.8). Fat-tree, on the other hand, maintains an accepted load below 0.1 phit/node/cycle.

4.2.2 Seven hotspot regions. Results with seven HS regions differ from those described in Section 4.2.1, as shown in Figure 4. The Flattened Butterfly and Dragonfly networks can recover for any duration of HS traffic burst below 20,000 cycles from the start of this adverse traffic. The largest temporary drop in accepted load for Flattened Butterfly is below 0.2 phits/node/cycle with the largest burst (1,000 cycles). For Dragonfly, the accepted load drops from 0.8 phits/node/cycle to 0.4, even during the largest hotspot traffic burst.

In the case of Fat-tree, it can be observed that splitting the HS traffic in 7 regions allows the network to recover when the bursts have a duration of 500 cycles or lower, whereas with a single HS region the network could only recover from a 100-cycle burst. For this network, when the burst is 1,000 cycles, its performance stagnates around 0.4/0.6 phits/node/cycle.

For Megafly, there is a high susceptibility to the specific HS region configuration, which depends on the choice of seed for the experiment. Depending on the experiment, maximum observed load values differ significantly, with some cases showing no performance drop whereas for other scenarios the drop in accepted load is severe and persists for a significant amount of time. Figure 4 displays the evolution of the accepted load with the simulation time for 3 different seeds.

5 CONCLUSIONS

This paper analyzes the network behavior in the presence of endpoint congestion, for a set of network topologies dimensioned to comparable sizes. In this scenario, a congestion management mechanism is needed to reduce the performance loss caused by the intersection of streams with endpoint congestion and other more benign traffic patterns. These mechanisms typically throttle the injection of hotspot streams to avoid performance losses on the benign traffic flows.

This study assesses the appropriate level of action to be applied to the hotspot flows to alleviate the network performance degradation. It also examines the optimal timing for implementing these actions to mitigate the most severe performance drops or the duration of cycles during which significant performance degradation is expected.

Although the contention factor at the endpoint determines a theoretical limit to which each source node can inject HS traffic, practical results have shown that these values also depend on the topology used. For certain topologies, such as Megafly and Dragonfly, the network may require a more restrictive limit on the injected load in practice than the theoretical limit. This is due to large cumulative loads flooding the network when bursts of this type of HS traffic are present. The phenomenon is exacerbated by the non-minimal adaptive routing mechanism, which is unable to distinguish between endpoint congestion and network congestion and tries to alleviate the congestion exploiting non-minimal routes. In other topologies, such as Flattened Butterfly and Fat-tree, the network can handle higher loads than the theoretical limit due to greater path diversity, which reduces the overlap between routes.

These results also serve as a baseline reference for the performance of a congestion management mechanism in these topologies with different speed reactions. Any implementation of such a mechanism must identify the particular network configuration to adapt its behavior in order to maximize performance and properly identify the scenario.

ACKNOWLEDGMENTS

This work has been supported by Grants PID2019-105660RB-C22, TED2021-131176B-I00 and PID2022-136454NB-C21 funded by MICIU/AEI/ 10.13039/501100011033nd by ERDF/EU; by the Spanish Ministry of Science and Innovation Ramón y Cajal RYC2021-033959-I, and the European HiPEAC Network of Excellence. The experiments have been executed on the Altamira HPC cluster, at the Institute of Physics of Cantabria (IFCA-CSIC).

REFERENCES

[1] Top 500. 2023. *Top 500 list.* https://www.top500.org/lists/top500/2023/11/
[2] Jung Ho Ahn, Nathan Binkert, Al Davis, Moray McLaren, and Robert S. Schreiber. 2009. HyperX: Topology, Routing, and Packaging of Efficient Large-Scale Networks. In *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis* (Portland, Oregon) *(SC '09)*. Association for Computing Machinery, New York, NY, USA, Article 41, 11 pages. https://doi.org/10.1145/1654059.1654101
[3] Yuichiro Ajima, Yuzo Takagi, Tomohiro Inoue, Shinya Hiramoto, and Toshiyuki Shimizu. 2011. The Tofu Interconnect. In *2011 IEEE 19th Annual Symposium on High Performance Interconnects*. 87–94. https://doi.org/10.1109/HOTI.2011.21
[4] Cristóbal Camarero. 2024. *RUST Crate caminos-lib.* https://docs.rs/caminos-lib/latest/caminos_lib/
[5] Dong Chen, Philip Heidelberger, Craig Stunkel, Yutaka Sugawara, Cyriel Minkenberg, Bogdan Prisacari, and German Rodriguez. 2016. An Evaluation of Network Architectures for Next Generation Supercomputers. In *2016 7th International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS)*. 11–21. https://doi.org/10.1109/PMBS.2016.007
[6] William James Dally and Brian Patrick Towles. 2004. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann.
[7] Daniele De Sensi, Salvatore Di Girolamo, Kim H. McMahon, Duncan Roweth, and Torsten Hoefler. 2020. An In-Depth Analysis of the Slingshot Interconnect. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE. https://doi.org/10.1109/sc41405.2020.00039
[8] Jesus Escudero-Sahuquillo, Ernst Gunnar Gran, Pedro Javier Garcia, Jose Flich, Tor Skeie, Olav Lysne, Francisco Jose Quiles, and Jose Duato. 2011. Combining Congested-Flow Isolation and Injection Throttling in HPC Interconnection Networks. In *2011 International Conference on Parallel Processing*. 662–672. https://doi.org/10.1109/ICPP.2011.80
[9] Mario Flajslik, Eric Borch, and Mike Parker. 2018. *Megafly: A Topology for Exascale Systems*. 289–310. https://doi.org/10.1007/978-3-319-92040-5_15
[10] Marina García, Enrique Vallejo, Ramón Beivide, Miguel Odriozola, Cristóbal Camarero, Mateo Valero, Germán Rodríguez, Jesús Labarta, and Cyriel Minkenberg. 2012. On-the-Fly Adaptive Routing in High-Radix Hierarchical Networks. In *2012 41st International Conference on Parallel Processing*. 279–288. https://doi.org/10.1109/ICPP.2012.46
[11] P. J. García, J. Flich, J. Duato, I. Johnson, F. J. Quiles, and F. Naven. 2005. Dynamic evolution of congestion trees: Analysis and impact on Switch Architecture. *High Performance Embedded Architectures and Compilers* (2005), 266–285. https://doi.org/10.1007/11587514_18
[12] Jason Howard. 2023. The First Direct Mesh-to-Mesh Photonic Fabric. In *2023 IEEE Hot Chips 35 Symposium (HCS)*. 1–17. https://doi.org/10.1109/HCS59251.2023.10254719
[13] Michael Jurczyk and Thomas Schwederski. 1996. Phenomenon of higher order head-of-line blocking in multistage interconnection networks under nonuniform

traffic patterns. *IEICE Transactions on Information and Systems* 79, 8 (1996), 1124–1129.

[14] M. Karol, M. Hluchyj, and S. Morgan. 1987. Input Versus Output Queueing on a Space-Division Packet Switch. *IEEE Transactions on Communications* 35, 12 (1987), 1347–1356. https://doi.org/10.1109/TCOM.1987.1096719

[15] John Kim, William J. Dally, and Dennis Abts. 2007. Flattened Butterfly: A Cost-Efficient Topology for High-Radix Networks. In *Proceedings of the 34th Annual International Symposium on Computer Architecture (ISCA '07)*. Association for Computing Machinery, 126–137. https://doi.org/10.1145/1250662.1250679

[16] John Kim, Wiliam J. Dally, Steve Scott, and Dennis Abts. 2008. Technology-Driven, Highly-Scalable Dragonfly Topology. In *2008 International Symposium on Computer Architecture*. 77–88. https://doi.org/10.1109/ISCA.2008.19

[17] Charles E. Leiserson. 1985. Fat-trees: Universal networks for hardware-efficient supercomputing. *IEEE Trans. Comput.* C-34, 10 (1985), 892–901. https://doi.org/10.1109/TC.1985.6312192

[18] Jose Rocher-Gonzalez, Jesus Escudero-Sahuquillo, Pedro J. Garcia, and Francisco J. Quiles. 2022. Congestion management in high-performance interconnection networks using adaptive routing notifications. *The Journal of Supercomputing* 79, 7 (2022), 7804–7834. https://doi.org/10.1007/s11227-022-04926-1

[19] Yuval Shpigelman, Gilad Shainer, Richard Graham, Yong Qin, Gerardo Cisneros-Stoianowski, and Craig Stunkel. 2022. *NVIDIA's Quantum InfiniBand Network Congestion Control Technology and Its Impact on Application Performance*. 26–43. https://doi.org/10.1007/978-3-031-07312-0_2

[20] 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)*. 1–8. https://doi.org/10.1109/HiPINEB.2017.11

[21] Arjun Singh. 2005. *Load-balanced routing in interconnection networks*. Stanford University.

[22] C. B. Stunkel, R. L. Graham, G. Shainer, M. Kagan, S. S. Sharkawi, B. Rosenburg, and G. A. Chochia. 2020. The high-speed networks of the Summit and Sierra supercomputers. *IBM Journal of Research and Development* 64, 3/4 (2020), 3:1–3:10. https://doi.org/10.1147/JRD.2020.2967330

[23] The Rust Foundation. 2024. *Rust, The Programming Language*. https://www.rust-lang.org/

[24] Matteo Turisini, Giorgio Amati, and Mirko Cestari. 2023. LEONARDO: A Pan-European Pre-Exascale Supercomputer for HPC and AI Applications. arXiv:2307.16885 [cs.DC]

Received 31 March 2024; accepted 12 April 2024