Contention-based Congestion Management in Large-Scale Networks

Gwangsun Kim, Changhyun Kim, Jiyun Jeong, Mike Parker[†], John Kim
KAIST
{gskim, nangigs, cjy9037, jjk12}@kaist.ac.kr
[†]Intel Corp.
mike.a.parker@intel.com

***Abstract*—Global adaptive routing exploits non-minimal paths to improve performance on adversarial traffic patterns and load-balance network channels in large-scale networks. However, most prior work on global adaptive routing have assumed *admissible* traffic pattern where no endpoint node is oversubscribed. In the presence of a greedy flow or hotspot traffic, we show how exploiting path diversity with global adaptive routing can spread network congestion and degrade performance. When global adaptive routing is combined with congestion management, the two types of congestion – *network* congestion that occurs within the interconnection network channels and *endpoint* congestion that occurs from oversubscribed endpoint nodes – are not properly differentiated. As a result, previously proposed congestion management mechanisms that are effective in addressing endpoint congestion are not necessarily effective when global adaptive routing is also used in the network. Thus, we propose a novel, low-cost contention-based congestion management (CBCM) to identify endpoint congestion based on the contention within the intermediate routers and at the endpoint nodes. While contention also occurs for *network* congestion, the endpoint nodes or the destination determines whether the congestion is endpoint congestion or network congestion. If it is only network congestion, CBCM ignores the network congestion and adaptive routing is allowed to minimize network congestion. However, if endpoint congestion occurs, CBCM throttles the hotspot senders and minimally route the traffic through a separate VC. Our evaluation across different traffic patterns and network sizes demonstrates that our approach is more robust in identifying endpoint congestion in the network while complementing global adaptive routing to avoid network congestion.**

I. INTRODUCTION

Large-scale networks that are found in supercomputer networks and data center networks can have a significant impact on overall system performance and cost [1], [2]. The increasing pin-bandwidth has enabled high-radix routers [3] in large-scale networks where the bandwidth is partitioned into a larger number of ports. The high-radix routers enable high-radix topologies, such as the flattened butterfly [4] and the dragonfly [5] topologies. Recently proposed HPC systems, including the Cray Cascade system [6] and the IBM PERCS [7] leverage a variation of the dragonfly topology. A critical component of the recently proposed high-radix topologies is non-minimal, global adaptive routing [8] that enables high performance on adversarial traffic patterns. Global adaptive routing for high-radix topologies have been improved, including alternative load-balancing for randomize routing [9], indirect global adaptive routing [10], and addressing the impact of the long-channel latency [11].

However, these prior work have not evaluated the impact of inadmissible traffic and the impact of adaptive routing on such traffic patterns. There have been recent works on congestion management [12], [13], [14] that proposed alternative congestion mechanisms, compared with the explicit congestion notification (ECN) mechanism. Other prior work have proposed isolating hotspot traffic [15], [16] but were limited to fat-tree network or other topologies where minimal routing (and deterministic routing) was used.

In this work, we explore the impact of inadmissible traffic on adaptive routing and congestion management. Admissible traffic pattern is defined as when none of the output nodes are oversubscribed, while with inadmissible traffic, an output endpoint node is oversubscribed and cause congestion back into the network [17]. Large-scale networks are often utilized by multiple users and/or applications instead of being used by a single workload. As a result, an adversarial user can in effect create a hotspot traffic pattern with continuous communication (many-to-one) and result in a performance bottleneck for others. In addition, poorly written parallel program or an unbalanced workload where multiple nodes continuously access a single memory module or node can create a hotspot traffic pattern. Prior works on congestion management [12], [13], [15], [16] address such endpoint congestion but do not explore them on high-radix networks with non-minimal, global adaptive routing.

We first differentiate between *network* congestion and *endpoint* congestion. Network congestion refers to network channel congestion where a network channel between routers in the interconnection network is saturated. In comparison, endpoint congestion occurs when a destination (or an endpoint) is oversubscribed and causes saturation back into the network. Prior work [10], [11], [18] on adaptive routing addresses network congestion while prior work [19], [12], [13], [14] on congestion management attempts to minimize endpoint congestion. In this work, we show how global adaptive routing can degrade overall performance if endpoint congestion occurs in the network. In addition, previously proposed congestion management mechanisms are not necessarily effective when endpoint congestion is combined with adaptive routing.

Thus, we propose a novel *contention*-based congestion management (CBCM) to detect contention within each router and

978-1-5090-3508-3/16/$31.00 © 2016 IEEE

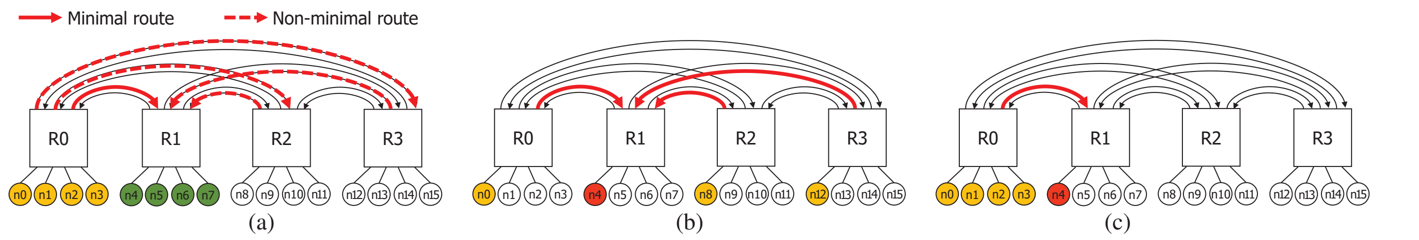


Fig. 1. Examples of different types of congestion in interconnection networks: (a) network congestion, (b) endpoint congestion, and (c) both network and endpoint congestion.

use the information to identify possible endpoint congestion. However, the endpoint (or destination) determines whether it was network congestion or endpoint congestion that occurred. If network congestion occurred, the congestion information is ignored at the endpoints to enable adaptive routing to load-balance the network channels. If endpoint congestion is identified, control packets are sent back to the sources contributing to the endpoint congestion to throttle the sources and minimize endpoint congestion. We describe the contention detection mechanism as well as the interface at both the source and the destination (i.e., endpoint) to support CBCM.

In particular, the contributions of this work include the following.

- We analyze how global adaptive routing with non-minimal routing spreads congestion for inadmissible traffic and can degrade performance, compared with minimal, deterministic routing.
- We show how congestion management needs to complement global adaptive routing and ensure that traffic causing endpoint congestion are minimally routed deterministically and does not spread congestion. As a result, we show how prior work on congestion management is not necessarily effective if global adaptive routing is used.
- We propose contention-based congestion management (CBCM) that leverages the contention degree within the router to identify endpoint congestion and throttle the sources accordingly – while complementing global adaptive routing algorithm.
- We evaluate CBCM across different traffic patterns and topologies to show how it minimizes performance degradation that can occur with global adaptive routing in the presence of endpoint congestion.

II. Congestion Analysis

In this section, we first describe different types of congestion that can occur in interconnection networks and then, describe how global adaptive routing can spread congestion when endpoint congestion occurs.

A. *Congestion in Interconnection Network*

Different types of congestions that can occur in interconnection networks are shown in Figure 1, using a 16-node 1D flattened butterfly [4] topology as an example. In this work we differentiate between *endpoint* congestion and *network* congestion. Network congestion is defined as congestion that occurs in the network channels of the interconnection network while endpoint congestion is defined as an endpoint (or node) being oversubscribed. An example of network congestion is shown in Figure 1(a) where no endpoint nodes are oversubscribed but a network channel connecting two routers (i.e., R0-R1) is congested because of the traffic pattern. In this particular example, using minimal routing results in the network congestion but non-minimal, adaptive routing exploits the path diversity and load-balances the network channels. The non-minimal routes from R0 to R1 in Figure 1(a) are shown with the dotted lines.

An example of endpoint congestion is shown in Figure 1(b) for a hotspot traffic pattern where some nodes send all of their traffic to a single endpoint node (i.e., *n*4 connected to R1). Since the endpoint is oversubscribed, endpoint congestion impacts other traffic that is not part of the hotspot traffic. Prior work [19], [12], [13], [14], [20], [21], [15] on congestion management have addressed such endpoint congestion and different approaches to overcome such congestion have been proposed. However, to the best of our knowledge, prior work have not considered the impact of adaptive routing on endpoint congestion. Figure 1(c) is an example where both endpoint and network congestions occur in the network. The traffic pattern is similar to Figure 1(b) but the hotspot traffic comes from nodes connected to the same router (i.e., R0). For this traffic pattern, adaptive routing does not provide any benefit as the endpoint becomes the bottleneck but only spreads the congestion throughout the network and degrades the performance of other traffic.

Table I summarizes the different types of congestion that can occur within the interconnection network. For Case (II), adaptive routing should be exploited while for Case (III), congestion management needs to be done. However, one key observation that we address is that hotspot nodes that cause endpoint congestion should not be allowed to adaptively route their traffic and *spread* the congestion. Tree saturation [22] is a well-known behavior from hotspot traffic or endpoint congestion. However, with adaptive routing, the impact of tree saturation can be more significant if sources contributing to the tree saturation adaptively route traffic and spread the traffic. In the following section, we describe the performance impact of the congestion spreading.

B. *Congestion Spreading*

To evaluate the impact of congestion spreading, we evaluate with a *combined* traffic pattern that combines two different traffic patterns. In this particular example, we assume nodes

TABLE I
DIFFERENT COMBINATIONS OF CONGESTION THAT CAN OCCUR IN INTERCONNECTION NETWORKS AND HOW ROUTING ALGORITHM AND CONGESTION MANAGEMENT SHOULD RESPOND. (O: PRESENCE, X: ABSENCE)

Case	Network congestion	Endpoint congestion	Adaptive routing	Congestion management
I	X	X	Not needed	Not needed
II	O	X	Route adaptively around network congestion	Not needed
III	X	O	Not needed	Throttle and minimally route inadmissible traffic that causes endpoint congestion
IV	O	O	Adaptive routing for admissible traffic that causes network congestion	Throttle and minimally route inadmissible traffic that causes endpoint congestion

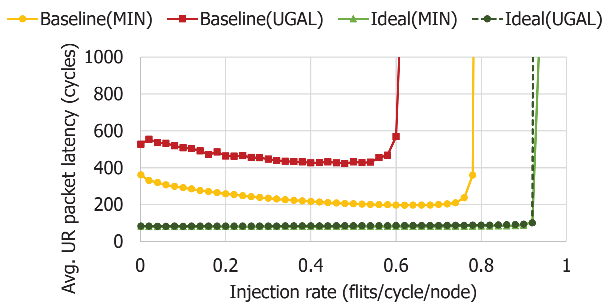


Fig. 2. Latency-throughput for a combined traffic pattern.

0,5,8,12 send hotspot traffic to node 4 and all of the remaining nodes send uniform random (UR) traffic to each other on a 16-node 1D flattened butterfly topology, similar to Figure 1(b).[1] We measure the latency-throughput for the uniform random traffic within the combined traffic and the results are shown in Figure 2 for both minimal (MIN) and adaptive (UGAL [8]) routing algorithms. The offered load of the UR traffic is varied while the hotspot traffic continuously injects traffic towards the hotspot destination. We also compare against an *ideal* routing algorithm where separate VCs are used for each traffic pattern (i.e., all of the hotspot traffic use a single VC and the UR traffic uses separate VCs) and the hotspot traffic is routed minimally. As shown in Figure 2, both Ideal(MIN) and Ideal(UGAL) provide high throughput as the impact of the traffic pattern is isolated through separate VCs and as expected, Ideal(UGAL) does not provide any benefit over Ideal(MIN) for UR traffic.

In the baseline without the traffic isolation through separate VCs, the UR traffic results in increased latency and reduced throughput with MIN routing due to the endpoint congestion from the hotspot traffic. This congestion causes tree-saturation [22], [1] as well as head-of-line blocking [23] and degrades performance. Different prior work [19], [12], [13], [14], [20], [21], [15] have tried to minimize the impact of endpoint congestion; however, prior work on congestion management have assumed deterministic routing (e.g., minimal routing). If global adaptive routing is used (UGAL [8]), both throughput and latency degrades further compared to MIN routing as saturation throughput is reduced by 23% while zero-load latency increases by 46%. Adaptive routing results in some fraction of the hotspot traffic being routed non-minimally and thus, spreads congestion across more channels and further degrades performance.

[1]We used the evaluation methodology described in Section V-A, and use single-flit packets to isolate the impact of flow control.

A similar trend is shown for a combined traffic pattern where hotspot traffic is combined with an adversarial (worst-case) traffic pattern for the 1D FBFLY (i.e., nodes connected to R_i sends traffic to nodes in R_{i+1}). Although not shown because of space constraints, Ideal(UGAL) provides a significant improvement in throughput compared with Ideal(MIN) for this traffic pattern since non-minimal paths are exploited. However, Baseline(UGAL) still results in performance degradation compared with Baseline(MIN) because of the congestion spreading. The key observation is that ***if adaptive routing is used when endpoint congestion occurs in the network, performance will degrade since congestion spreading occurs across other network channels***.

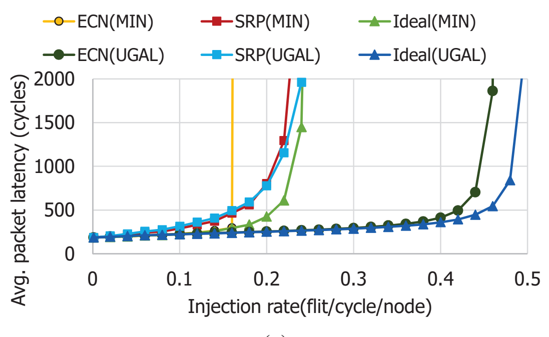
III. LIMITATIONS OF PRIOR CONGESTION MANAGEMENT

In this section, we describe the limitations of previously proposed congestion management schemes, including explicit congestion notification (ECN) [19], [24], and recently proposed speculative reservation protocol (SRP) [12]. In particular, if global adaptive routing is used when endpoint congestion occurs, we show how existing congestion management approaches are not necessarily effective because the non-minimal routing ends up spreading the congestion.

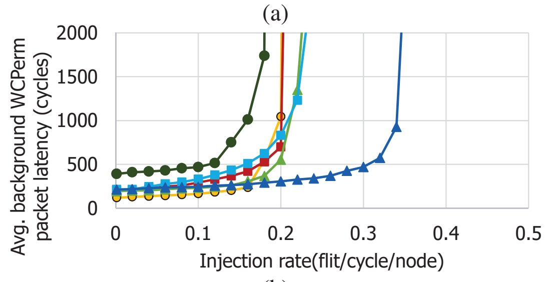
A. Explicit Congestion Notification (ECN)

Explicit congestion notification (ECN) [24] uses a threshold to determine congestion and marks packets that traverse through a congested buffer – and notifies the sender of congestion. If the depth of the buffer does not exceed the threshold, ECN does not detect congestion. Some of the limitations of ECN were previously identified, which include slow transient response [12]. However, if ECN is used in a network topology with global adaptive routing that includes non-minimal routing, there are other limitations that include the following.

- While ECN addresses endpoint congestion by throttling sources when congestion is detected, ECN waits until the threshold is exceeded. As a result, the endpoint congestion causing traffic (i.e., hotspot traffic) can spread its traffic (or congestion) with global adaptive routing while waiting for the threshold to be reached.
- ECN detects *network* congestion and throttles sources that contribute to the network congestion. However, at a high



(a)



(b)

Fig. 3. Limitations of SRP and ECN with (a) a worst-case permutation and (b) combined traffic (worst-case permutation traffic combined with a 4-to-1 hotspot traffic pattern) on a 16-node 1D flattened butterfly network.

offered traffic load, ECN can unnecessarily throttle sources when some of the traffic can be routed non-minimally.

- Since buffer occupancy is used to detect congestion but the buffer is shared by different flows, non-minimally routed packets can also be routed through the congested buffer and unnecessarily throttle sources that do not directly contribute to the channel congestion.

Performance results for different adversarial traffic patterns are shown in Figure 3. For an adversarial permutation traffic pattern (i.e., node n_i sends traffic to node $n_{i+4 \bmod N}$, where N is the network size), as shown in Figure 3(a), ECN(UGAL) performs relatively well and nearly matches Ideal(UGAL) – i.e., congestion management does not interfere with adaptive routing (within 8% in terms of network throughput). However, when combined with hotspot traffic (Figure 3(b)), ECN(UGAL) degrades performance by approximately 2× compared with Ideal(UGAL) and both latency and throughput are worse compared with ECN(MIN). The limitation of adaptive routing with this particular traffic pattern is shown in Figure 4 with the impact of adaptive routing and endpoint congestion highlighted for R3. As some of the hotspot traffic are routed non-minimally through R3, the channel between R3-R0 becomes congested from this non-minimally routed traffic. As a result, nodes $n13$, $n14$, $n15$ which need to send traffic to R0 (i.e., $n1$, $n2$, $n3$) also send traffic through this congested buffer – thus, ECN incorrectly notifies $n13$, $n14$, $n15$ that their traffic should be throttled and results in throughput degradation.

B. *Speculative Reservation Protocol (SRP)*

Speculative Reservation Protocol (SRP) is a reservation-based endpoint congestion management mechanism [12]. Reservation-based approaches can increase network latency

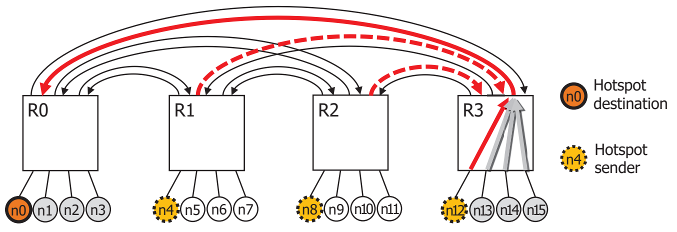
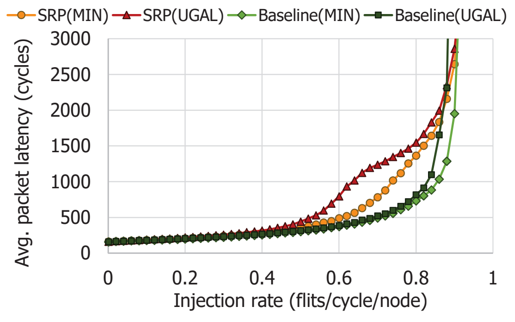


Fig. 4. An adversarial permutation traffic pattern combined with hotspot traffic in a 16-node flattened butterfly network to highlight the limitation of ECN.

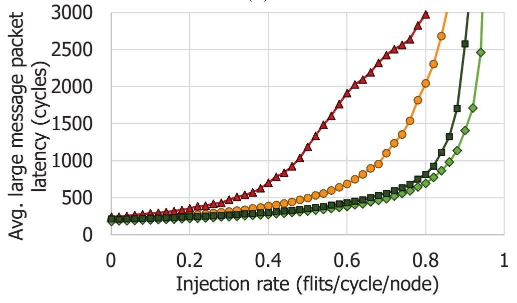
because of the reservation overhead but SRP speculatively transmits packets without reservation to reduce the overhead. Speculative packets have lower priority than non-speculative packets and can be dropped after a timeout. While SRP minimizes endpoint congestion, SRP restricts the number of speculative packets that can be injected into the network (e.g., up to 1 message per destination from each source). This approach can limit performance for permutation traffic where each source sends traffic to a single destination and become more problematic when round-trip latency is high. As shown in Figure 3, with UGAL adaptive routing algorithm, the throughput of SRP is significantly degraded compared to the ideal routing because of the long round-trip latency of reservation request and grant packets. SRP can easily be modified to increase the performance by injecting multiple messages to the same destination in the network (with additional hardware complexity to support multiple messages in-flight). However, this can fundamentally increase the congestion for other traffic patterns.

Another limitation of SRP is with benign, uniform random traffic when no endpoint congestion exists. As shown in Figure 5(a), the latency for both MIN and UGAL routing with SRP at intermediate loads can be up to 2.45× higher than without SRP. SRP allows transmitting multiple in-flight reservations towards different destinations to overlap the reservation overhead and improve overall performance. However, since two reservations can be done independently, the reservation grants received at the source can be overlapped in time. Since the source can only transmit one message at any given time, the message with latter arriving grant will need to be stalled at the source and wait until the other message has been transmitted – and thus, increasing the latency. When the traffic approaches saturation, many of the messages will experience such delay at the source but once the delayed packets start being injected, the throughput of the network will not be impacted. The intermediate latency for SRP(UGAL) is much higher as the non-minimally routed speculative packets encounter higher per-hop latency due to its low priority compared with non-speculative packets.

SRP also implements time-out to ensure that if speculative packets cause too much network congestion, the speculative packets are dropped. However, since SRP cannot distinguish endpoint congestion from network congestion, speculative packets can be dropped when network congestion exists and result in performance loss. To illustrate this limitation, we

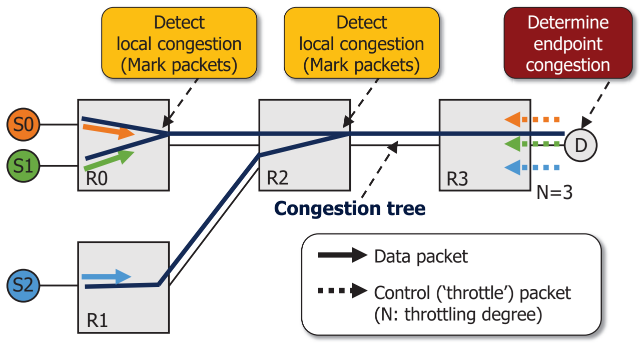


(a)



(b)

Fig. 5. Limitations of SRP with a (a) uniform random traffic and (b) combined UR-UR traffic pattern in a 16-node 1D flattened butterfly network.



Detect local congestion (Mark packets)
Detect local congestion (Mark packets)
Determine endpoint congestion
S0
S1
S2
R0
R1
R2
R3
D
N=3
Congestion tree
Data packet
Control ('throttle') packet (N: throttling degree)

Fig. 6. Overview of the proposed congestion management mechanism.

evaluate another combined traffic where there is no endpoint congestion. We use combined traffic with two UR (uniform random) traffic (UR1-UR2) – half of the nodes at each router send UR traffic with each other using large (4-packet) messages (UR1) while the other half of the nodes also send UR traffic within itself but use small (single-packet) messages (UR2).[2] We assume UR2 continuously communicates at a high (90%) offered traffic load but since the messages are non-speculative, they have higher priority. In comparison, UR1 traffic sends packet speculatively (in parallel with reservation requests). However, network congestion and the higher priority non-speculative packets will result in the speculative packets from UR1 having higher latency in the network and then, eventually dropped. For the results shown in Figure 5(b), at the injection rate of 0.6 for SRP(UGAL), approximately 15% of speculative packets were dropped and the average packet latency of UR1 traffic is significantly increased by 4.4×. While SRP(MIN) suffers from throughput degradation and increased latency, UGAL aggravates the problem as some packets can be routed non-minimally and speculative packet latency can further increase due to additional hops.

IV. Contention-Based Congestion Management

A. Overview

The main goal of the proposed contention-based congestion management (CBCM) is congestion management that addresses endpoint congestion while complementing global adaptive routing that adaptively routes traffic around network congestion. Thus, CBCM throttles sources that contribute to endpoint congestion to prevent congestion tree from being formed and ensure the same sources do not spread the congestion through adaptive routing. CBCM uses a separate virtual channel to isolate the impact of endpoint congestion. Once endpoint congestion is identified along with the sources that contribute to the endpoint congestion, the traffic from the sources are throttled and minimally routed through this extra virtual channel.

A high-level overview of CBCM is described in Figure 6 for a simple traffic where three sources (S0, S1, S2) send traffic to a single destination node (D) through a multi-stage network. Each router locally detects congestion at each output port based on the *contention degree*. Every packet that goes through a congested output port will be marked in the packet header (Section IV-B1). At each destination, it is determined whether endpoint congestion exists based on the marked packets. If endpoint congestion is detected, 'throttle' control packets are sent back to the sources that contribute to the endpoint congestion (Section IV-B2). When the source receives a 'throttle' control packet, it throttles the flow towards the hotspot destination by $1/D_t$, where D_t is the throttling degree specified in the control packet (Section IV-B3). The throttled packets will be isolated from normal packets through a separate VC and will be restricted to use only minimal routing. The source unthrottles itself based on packet generation rate towards the destination (Section IV-B4) when it no longer contributes to endpoint congestion. When unthrottled, the source will notify the hotspot destination by sending an 'unthrottle' control packet.

Examples of how CBCM works under different combinations of congestion are described below, based on different scenarios shown earlier in Table I and Figure 1.

Case I : There is no network or endpoint congestion. Packets are routed minimally and no source node is throttled.

Case II : When network congestion occurs due to adversarial traffic pattern, contention occurs for an output port in R0 as shown in Figure 1(a). As a result, packets will be marked as congested and be received at its destination at R1. Even though the endpoint receives marked packets, the endpoint determines that endpoint congestion did not occur since unmarked packets

[2]We chose UR2 with small messages as SRP does not use reservation mechanism for them because of the high reservation overhead and always injects them non-speculatively with high priority. Recent work [14] has addressed the limitation of small messages in reservation for endpoint congestion using SRP. However, the problem that we describe is related to network congestion and [14] does not reduce such network congestion.

will arrive from other non-minimal paths without congestion, which signifies that congestion was limited to a limited region (i.e., network congestion occurred). Therefore, no congestion management is done and adaptive routing routes some packets non-minimally to minimize network congestion.

Case III: Within R1 of Figure 1(b), contention for router output to $n4$ will be detected. Thus, all packets injected at R1 to $n4$ will be marked as congested and the destination node will determine that three sources are contributing to endpoint congestion – resulting in control packets being sent to the three source nodes ($n0$, $n8$, $n12$).

Case IV: Similar to case II, $n4$ in Figure 1(c) will received marked packets. However, unlike Case II, *all* packets towards $n4$ will be marked at R1 since the contention degree towards $n4$ is sustained at a high level (Section IV-B1). Thus, $n4$ will send control packets to $n0$ - $n3$ and have them throttled. Once a control packet is received, $n0$ - $n3$ also stop using adaptive routing and only minimal routing is used until endpoint congestion disappears.

In the following subsection, we describe how contention is determined within the routers and how endpoint differentiates between network and endpoint congestions.

B. Mechanism

1) Congestion Detection at Switches: In CBCM, we leverage *contention* within the router to estimate possible congestion since contention often leads to congestion. Other metrics (such as queuing latency or queue depth) can be used to estimate congestion; however, those metrics only start to appear *after* congestion has started. The contention degree is observed at each router instead of observing it only at endpoints since for some traffic patterns, the contention can only be observed at intermediate routers while endpoint congestion (oversubscription) exists. An example of this intermediate router creating endpoint congestion is shown in Figure 6.

Within each router, contention degree (D_c) for each output port is determined each cycle. If $D_c > 1$, contention is experienced at the output port while if $D_c \leq 1$, then there is only one (or no) request for an output port and no contention. The contention can be determined through switch allocation (i.e., how many packets request a particular switch output) if there is a single VC per port. However, this does not provide accurate contention information since each input often has multiple virtual channels (VCs) [25] and each VC can contribute to congestion. Thus, we propose a *weighted* contention degree (D_{c_w}) to determine the contention for an output port in the VC allocation stage. If there are v input VCs that have requests from a single input port, each request contributes a contention of $1/v$ to each output – i.e., normalize the weight of each request by the number of input VCs at the input port that have requests (not the total number of input VCs). If there is only one VC at the input port with a request for an output, the weight of the VC is simply 1. At each output, the weighted contention degree is calculated by summing

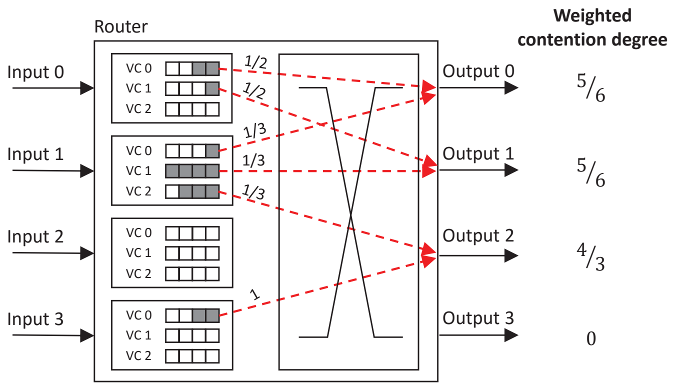


Fig. 7. Contention degree calculation example with weight normalization. A shaded VC buffer indicates that flits are present and the front flit is requesting an output VC.

the weighted inputs. An example of this weighted contention degree approach is shown in Figure 7.

However, calculating weighed contention degree can be complex since it requires non-integer operations. As a result, we approximate the weighted contention degree with randomization or *randomized contention degree* (D_{c_r}). Instead of calculating weights, one request from each input is randomly selected and only the randomly selected input is used to calculate the contention degree. Our evaluations show that this randomized approach has minimal impact on overall performance as we use a moving average of contention degree (described below) for detecting congested output port. Intuitively, randomly selecting only one request per input is sufficient if a moving average of the contention degree over time is used. For example, with the weighted approach, each VC at an input port will contribute a weight of $1/v$ to D_{c_w} every cycle, where v is the number of VCs with requests for an output. However, with a randomized approach, a randomly selected VC will contribute a weight of 1 but on average, each VC will be selected once every v cycles – thus, the effective weight contributed to the moving average is $1/v$. By using the moving average, both of these approaches have approximately similar results. This approach is different from using switch contention information since switch contention does not include contention information for other VCs that did not receive VC allocation and do not participate in the switch allocation.

If there exists endpoint congestion, contention degree will be sustained at a level that is greater than one.[3] The goal is to identify steady-state congestion and throttle the sources accordingly; thus, temporary (or instantaneous) congestion in the network needs to be ignored. To properly identify steady-state congestion, moving average (MA) of the contention degree (D_{c_r}) is calculated for each output port. The number of samples for moving average is specified by a parameter *NumSamples*. As allocation requests are granted and new requests arrive, there can be oscillation in the contention degree.

[3]The inverse is not necessarily true. High contention degree can be caused by either endpoint congestion, network congestion, or both.

However, if there are more than one steady flow towards the same output, the lower bound of the contention degree will be sustained at a level greater than 1.0. In order to obtain the amplitude of the contention degree oscillation, maximum and minimum of the contention degree are also sampled every interval specified by a parameter $BoundSampleInterval$. Moving averages of the maximum and minimum values are calculated with the samples as well. Then, the metric for local congestion detection is calculated by:

$$metric = MA(D_{c_r}) - (MA(\max D_{c_r}) - MA(\min D_{c_r}))/2$$

The second term of the equation estimates half of the amplitude of the movement of the contention degree, and by subtracting it from the term $MA(D_{c_r})$, the metric estimates the lower bound of the movement of the contention degree, and this indicates the minimum number of sustained flows that go through this output port. This term is used to approximate the variance (or the fluctuation) of the contention. If the metric is greater than one, it signifies that there is sustained contention, which will form congestion tree. The router marks the header of all packets that go through this output port such that the destination node can determine whether endpoint congestion exists. We assume $NumSamples = 100$ and $BoundSampleInterval = 10$ in this work. The moving average calculation can be done at low cost by having a FIFO queue where contention degree is pushed each cycle, and maintaining a sum of the contention degree which is incremented by current contention degree and decremented by what is popped from the queue every cycle. Since the congestion detection is done over a large number of samples specified by the aforementioned parameters, there is no impact on the critical path of each individual packets.

2) Throttling Decision at Hotspot: We define the throttling degree (D_t) as the number of sources contributing to the endpoint congestion which represents how much throttling should be done at each source. At each destination node, the throttling degree is determined by counting the number of sources that send traffic to this particular destination node. Each node maintains a list of source node IDs that contributes to a possible endpoint congestion for this destination node. Initially, this list is empty. When a marked packet is received, the packet's source ID is added to the list and the node starts to observe all received packets during an epoch that begins at this cycle. Whenever a marked packet is received, it will be added to the list, but when an un-marked packet is received, the list is emptied since this can signify that there exists an uncongested path towards this destination, indicating network congestion, not endpoint congestion. The procedure starts over when a marked packet is received next.

When all packets received during an epoch are marked, this node is identified as a hotspot. Then, 'throttle' control packets will be generated and sent to the sources in the list with throttling degree D_t (= the number of source IDs in the list) included in the packet. Once an endpoint is identified as a hotspot, all sources that sent packets to this endpoint, regardless of the packet marking, will be added to the list. After the sources are throttled and the endpoint congestion is removed, packets toward the hotspot will not experience high contention and they can arrive the destination unmarked even if many-to-one hotspot traffic continues to exist.

Once the sources are notified, the destination node continues to monitor endpoint congestion as D_t can change – e.g., D_t can increase if additional nodes contributes to the endpoint congestion. The bandwidth overhead parameter (ε) determines the maximum bandwidth that can be used by the control packets. For example, if the parameter has the value 0.05, and throttle control packets are sent to D_t nodes, no control packet can be sent during the next $D_t/0.05$ cycles. Afterwards, if the number of sources in the list is different from D_t, the endpoint will send another set of 'throttle' packets with the updated D_t.

3) Injection Throttling at Sources: When a source receives a 'throttle' control packet from node i, it throttles injection towards the node i by $1/D_{t,i}$ with the throttling degree $D_{t,i}$ specified by the throttle control packet. Throttling is done independently for each destination, and packets can be injected towards other destinations without any restriction. The packets towards the throttled destination are restricted to use a dedicated, low-priority VC to isolate the identified hotspot traffic from other traffic. This VC is shared by all nodes in the system. While using a separate VC is sufficient to isolate traffic, using only a separate VC without any throttling can result in throughput unfairness between the hotspot senders, as we show later in Section VI-C. Thus, CBCM leverages both throttling and VC isolation in this work.

The mechanism for throttling or rate controlling at source is described in Algorithm 1. A source maintains a token for each identified hotspot destination node. Once a source receives a throttle control packet from destination i, the token for node i (t_i) is reset to 0. A token for the destination is incremented every $D_{t,i}$th cycle and a packet can be injected only if the number of available tokens is greater than or equal to the packet size in flits. After a packet is injected, the number of tokens is decremented by the packet size. The throttling state (and the tokens) is managed independently within each node and continues until the source determines to unthrottle or a control packet is received from the destination that signifies a change in the throttling degree ($D_{t,i}$).

4) Unthrottling Decision: After a source is throttled, the source measures packet generation rate towards the identified hotspot destination during an epoch.[4] The packet generation rate refers to the rate new packets are generated at the source and queued – it is not necessarily the rate at which packets are injected into the network since the injection rate is determined by the state of the network. Since packets cannot be generated when the source queue or the network interface queue is full, we assume the source still contributes to the congestion and unthrottling is not done if the queue is full. When the packet generation rate during the last epoch is less than $1/D_{t,i}$ for the

[4]This epoch is different from the epoch used at the destination to determine endpoint congestion.

Algorithm 1 Pseudo-code for injection throttling at sources.

```
$t_i$: the number of tokens for destination $i$
$D_{t,i}$: throttling degree for destination $i$
At each cycle, execute the following statements:

if throttle control packet received for dest $i$ then
    $t_i \leftarrow 0$
end if

if $\exists$ packet to inject then
    if NOT being throttled towards dest $i$ then
        inject packet
    else if $t_i \geq$ packet_size then
        inject packet
        $t_i \leftarrow t_i -$ packet_size
    end if
end if

for each dest $i$ that this source is throttled towards
    do $t_i \leftarrow t_i + 1/D_{t,i}$
```

hotspot node i, it is assumed that either the traffic towards the hotspot was reduced or this node was incorrectly detected as a hotspot sender due to temporary congestion in the network. Thus, the source unthrottles itself, and will inject packets to the normal VCs towards its destination. The unthrottling is done independently for each destination node as well. When a source unthrottles itself, it sends an 'unthrottle' control packet to the hotspot. When the hotspot receives an unthrottle packet, it removes the source from the hotspot sender list and generates and sends updated 'throttle' control packets to the rest of the sources that are currently in the list with the updated throttling degree D_t, to allow other sources to use the available endpoint bandwidth.

After a source unthrottles, it is possible that a large number of flit tokens can be accumulated as the source waits for the epoch to finish while injecting little or no traffic. Thus, the flit tokens for the hotspot sender towards the destination is reset when it receives a subsequent throttle control packet later (line 7 in Algorithm 1), in order to properly throttle the newly identified flow towards the hotspot.

V. Evaluation

A. Methodology

We used Booksim [26], a cycle-accurate interconnection network simulator, to evaluate the proposed congestion management mechanism, similar to prior work on high-radix networks [4], [5], [10], [12]. Due to simulation feasibility, for synthetic traffic evaluation, we assumed a 512-node two-dimensional flattened butterfly network with 64 radix-22 routers with 8 nodes concentrated at each router. We assumed a $2\times$ internal speedup of the switch and Virtual-Output-Queueing (VOQ) [17] with four VCs for each output port at each input buffer such that the switch does not become a bottleneck in overall network performance. For SRP, two additional VCs are used for speculative packets and two more VCs are used for control packets while ECN requires one control packet VC. For CBCM, we assume a separate high-priority control VC for a single-flit 'throttle' and 'unthrottle' control packets and a low-priority VC dedicated for identified hotspot traffic. In order to model long global channels, we assume 100-cycle channel latency for channels between routers and VC buffer depth of 256 to cover the high credit round-trip delay and 4-packet output buffer. For synthetic traffic, the message size is 4 packets, where each packet consists of 32 flits. At each network source, our simulator models one queue for each network destination similar to Infiniband queue-pairs and if there are messages for different destinations, they are arbitrated within the source in a round-robin manner.

For SRP-specific parameters, we assumed 2,000-cycle Time-To-Wait (TTW), reservation overhead adjustment of 0.05, and minimum reservation message size of 4 packets. For ECN-specific parameters, we assumed buffer threshold of 90% input buffer capacity, inter-packet delay increment of 400 cycles, maximum inter-packet delay of 1,500 cycles, inter-packet delay decrement of 50 cycles, and inter-packet delay decrement timer of 1,000 cycles.

For real workload evaluation, we integrated our modified interconnection network simulator with SST/Macro [27] and used the trace of the following MPI workloads: HILO (Neutron Transport Evaluation and Test Suite) [28], BigFFT (large 3D FFT with 2D domain decomposition pattern) [29], FB (Fill Boundary operation from production PDE solver) [30], MG (geometric multigrid v-cycle from production elliptic solver) [30], CNS (Compressible Navier Stokes from BoxLib library) [30], and NB (Nekbone: Poison equation solver using conjugate gradient iteration with no preconditioner) from [31]. We scaled down the network size for the trace evaluation to a 256-node one-dimensional flattened butterfly with 16 radix-16 routers and 16 nodes concentrated at each router. In order to evaluate the impact of endpoint congestion on real workloads, 32 nodes distributed across 16 routers (i.e., two for each router) are selected as hotspot senders that inject traffic to a single hotspot while other nodes inject trace packets. We also evaluated the impact of uniform random traffic on the workload performance by having half of the network nodes inject UR traffic with each other. The network was warmed up for 500,000 cycles with synthetic hotspot or UR traffic only, and then workload trace packets were injected.

B. Synthetic Workload Result

Figure 8(a) compares the latency-throughput curves for uniform random traffic for the ideal routing, SRP, and CBCM with minimal and UGAL routing algorithms. The ideal routing and CBCM resulted in nearly identical performance. For benign traffic pattern, even if temporary congestion occurs at switches, CBCM rarely detects endpoint congestion. At the injection rate of 0.5 flits/cycle/node, incorrect detection occurred less than 0.001% of runtime, and even when the network was saturated, it occurred for only 2.2% of runtime

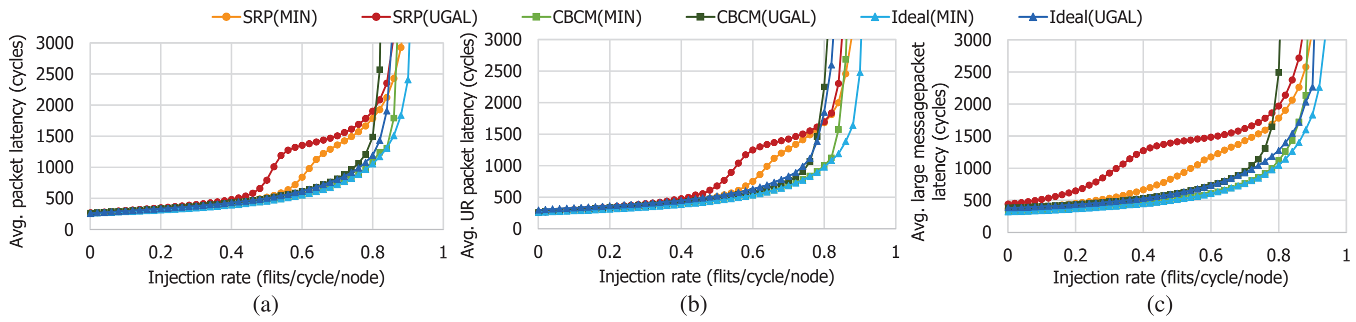


Fig. 8. Latency-throughput curves of (a) uniform random traffic, (b) uniform random traffic combined with 40-to-1 hotspot traffic, and (c) UR-UR combined traffic pattern.

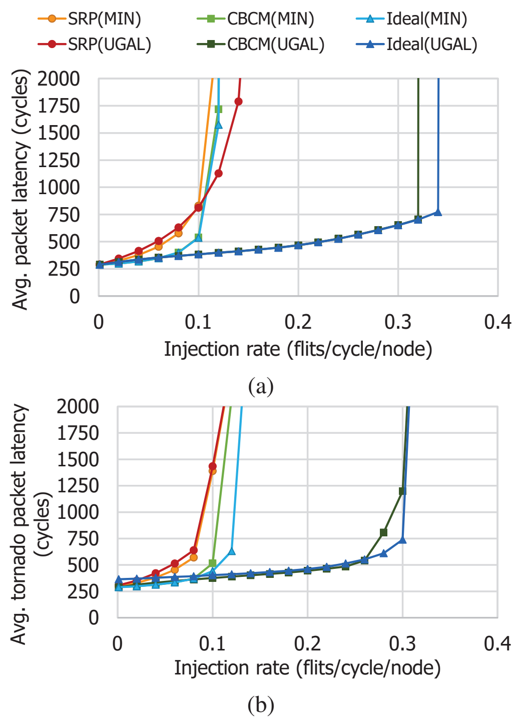


Fig. 9. Latency-throughput curves of (a) tornado traffic and (b) tornado traffic combined with 63-to-1 hotspot.

for each flow. Furthermore, after the incorrect detection occurs, the source quickly unthrottles itself based on packet generation rate, resulting in high performance, although there is slight degradation in saturation throughput. For the SRP, the latency was increased for both MIN and UGAL for medium and high traffic load as it suffers from the overlapped reservations and high latency of non-minimally routed speculative packets, as described in Section III-B.

The performance comparison with a combined UR-hotspot traffic pattern is shown in Figure 8(b). Out of 512 nodes in the network, 40 hotspot senders and a single hotspot were randomly selected. The same set of hotspot nodes is used across different schemes evaluated. The hotspot senders inject traffic at 100% of injection bandwidth and the UR nodes only send traffic among themselves. Although not shown, the baseline with minimal routing algorithm experiences very high latency due to congestion created by the hotspot traffic even at a low UR traffic load, and the network saturated around 10% load. The baseline with UGAL routing algorithm resulted in a very high latency even with near zero-load and was not shown.

However, with CBCM, the hotspot senders were effectively throttled resulting in UR traffic latency and throughput that are very close to that of the ideal routing mechanism. On the other hand, SRP suffered from the increase in latency at intermediate loads similar to the UR-only traffic pattern, although it effectively prevented congestion by hotspot traffic.

Figure 8(c) shows the performance for a combined UR-UR traffic pattern where half of the nodes send large UR messages with each other while the other half of the nodes send small messages with each other. The large message injection rate was varied while the small message was injected at a fixed high (90%) load. SRP resulted in higher latency for the large messages as speculatively sent packets were deprioritized, resulting in high latency and often packet drop. The increase in latency was aggravated with UGAL compared to minimal routing as some packets were sent non-minimally and the additional hops further increased the latency of the speculative packets. At an intermediate load of 40%, SRP with UGAL resulted in approximately 2.4$\times$ higher latency than CBCM with UGAL as 52% of the packets were sent speculatively and 24% of them were dropped. By contrast, CBCM provided near-ideal performance as network congestion was correctly distinguished from endpoint congestion.

The results for the worst-case tornado traffic pattern are shown in Figure 9(a). As described in Section III-B, SRP saturated very early as it limits outstanding speculative packets to prevent congestion regardless of routing algorithm. On the other hand, CBCM showed high performance as endpoint congestion was not detected for this traffic pattern. Even if packets can be marked due to temporary congestion, since each destination only receives packets from a single source in a permutation traffic pattern, no throttling was done as throttling degree is never greater than one. We also evaluated CBCM with a non-permutation worst-case traffic pattern (where each node in router R_i send traffic to any of the nodes in router R_{i+1}) for CBCM but the performance was nearly identical to that with the worst-case permutation traffic pattern as it can distinguish network congestion from endpoint congestion.

The trends of latency-throughput curves for worst-case tornado traffic combined with 63-to-1 hotspot traffic is shown in Figure 9(b). In this traffic pattern, the first nodes at router R_1

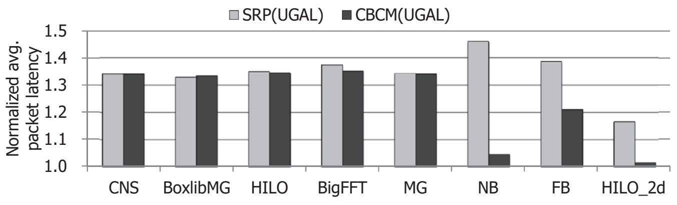


Fig. 10. Real workload packet latency with half of the nodes sending short messages a in UR traffic pattern with each other at 70% load.

to R_{63} (i.e., one node at each router) send traffic to node 0 at router R_0, while other nodes send the worst-case permutation traffic. The latency-throughput curve of the baseline was not shown as its zero-load latency exceeded the range of latency shown in the plot. Similar to the UR-hotspot combined traffic pattern, CBCM effectively managed the congestion and resulted in a nearly ideal performance. The SRP resulted in the same early saturation as the worst-case permutation traffic, while endpoint congestion by hotspot traffic was prevented.

C. Real Workload Result

To analyze the impact of congestion management and adaptive routing on real workloads, we combined the traffic from workload traces with different synthetic workloads such as hotspot traffic and uniform random traffic. Although the results are not shown due to page constraints, when hotspot traffic was combined, the baseline without any congestion management experienced significantly higher trace packet latency – e.g., for `HILO_2d` with UGAL routing, the latency increased by approximately 80× compared to when the hotspot did not exist as adaptive routing spread the congestion. With ECN, for some workloads, the latency increase was relatively small but for some workloads, it resulted in a significantly higher latency with UGAL as it occasionally mislabeled trace packets when they were mixed in the same buffer as the hotspot traffic, as described in Section III-A, and result in up to 3× increase in latency. Both SRP and CBCM were effective in isolating the hotspot traffic and once the hotspot traffic was isolated, there were minimal impacts but both SRP and CBCM increased average packet latency by up to 7%.

Figure 10 shows the trace packet latency with half of the network nodes injecting short messages in UR traffic pattern with each other at 70% load. The results show the packet latency normalized to when the traces were simulated alone without the mixed traffic. This UR combined traffic is an example of where network congestion is created and SRP incorrectly identifies that there exists endpoint congestion. The workloads are organized in Figure 10 in increasing order of average size of the messages generated from the trace – i.e., workloads on the left generate relatively short messages while the workloads on the right generate relatively long messages on average. Results show that for workloads with long messages, SRP results in a significant increase in latency as long messages suffer from low priority than short messages that are sent without reservation at a high priority. By contrast, CBCM does not cause such unfairness and can result in up

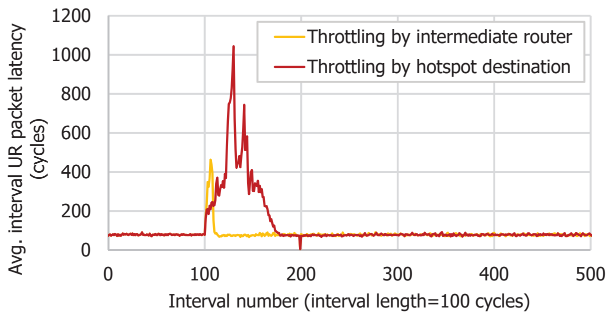


Fig. 11. Transient behavior comparison of intermediate throttling and endpoint throttling for combined traffic (UR + hotspot).

to 40% reduction in latency compared with SRP, while for some workloads such as `CNS` with relatively short messages, the difference between CBCM and SRP was small.

D. Hardware Overhead

The CBCM can be implemented with low overhead. It requires two additional VCs (one for control packet and another for low-priority hotspot traffic). At the endpoint, the network interface needs to have a storage (bit vector of size N) for identifying sources that contribute to endpoint congestion, where N is the number of nodes in the network. The interface for source requires the logic for injection throttling (Algorithm 1), registers for the number of tokens and throttling degrees for throttled flows, and the logic for measuring packet generation rate. At routers, for each output port within the router, a buffer is needed to store the samples of contention degree, as well as the buffer to store the minimum and maximum values of contention degree. There is additional ALU logic needed to calculate the moving average; however, since the width of the ALU is only $log_2(k)$ wide, where k is the radix of a router, the overhead is relatively small. Assuming a radix-64 router and a total of 120 contention degree samples including D_{c_r} (100 samples), $\min D_{c_r}$ (10 samples), and $\max D_{c_r}$ (10 samples), the overhead of CBCM is only ~5.6 KB per router, which represents only ~3% storage overhead compared to the YARC router [32]. The overhead of CBCM can be further reduced at the cost of sacrificing performance by using a smaller number of samples for moving average in congestion detection or using a coarse-grained bit vector at the endpoints.

VI. Discussion

A. Early Congestion Detection

While CBCM determines endpoint and network congestion at the endpoint network interface, by using contention within the routers, congestion can be detected earlier if the sources causing endpoint congestion are closely located (e.g., in the same router) as the root of the congestion tree is closer to the sources. For such congestion, it is beneficial to detect congestion at an intermediate router and notify/throttle the sources early before congestion is further spread throughout the network, especially if network round-trip latency is long. For the network and traffic pattern shown in Figure 1(c), Figure 11 compares the transient behavior of detection at

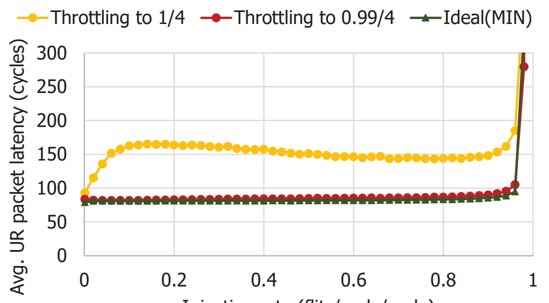


Fig. 12. Latency-throughput curves of background UR traffic as 4-to-1 hotspot senders are throttled to different injection rate.

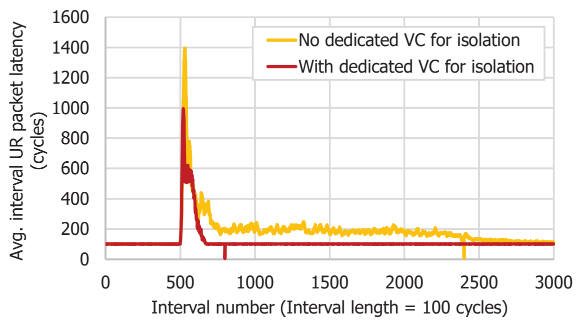


Fig. 13. Impact of dedicated VC for hotspot traffic isolation on transient behavior for UR traffic combined with hotspot traffic.

an intermediate router (i.e., R0) to that of detection at the destination router, assuming UGAL routing algorithm and channel latency of 100 cycles. Background traffic is uniform random with an injection rate of 40%, and the hotspot traffic begins injection at the 100th interval. As the intermediate router can throttle the hotspot sources sooner, the background UR traffic is less affected by the congestion and the peak of the latency is reduced to approximately 0.42×, and it also took approximately 0.22× of time for the latency to stabilize. While there are performance benefits of intermediate router throttling, it comes at the additional cost of detecting congestion at an intermediate router as contention detection metric needs to be computed at each switch for all network destination nodes, not just for each output port. This is needed for the intermediate router to distinguish network congestion from endpoint congestion and prevent throttling due to network congestion. There is also an overhead of having an intermediate router create control packets and inject them back towards the sources as well.

B. *Separate VC vs Shared VC for Hotspot Traffic*

In our proposed CBCM, we assumed a separate VC to isolate the traffic causing endpoint congestion to minimize the performance impact of the congestion. However, CBCM can also be implemented without a separate VC and we discuss the design consideration as well as the trade-off in this section. Ideally, if there are N hotspot senders, each of them should be throttled to the rate of $1/N$ and the hotspot will be able to accept the traffic at the full ejection channel bandwidth. However, due to various non-ideal behaviors of routers, including imperfect VC and switch allocators, head-of-line blocking, etc., the hotspot destination node may not be able to accept traffic at the full rate when N senders each inject at the rate of $1/N$. Thus, contention among hotspot packets will frequently occur and latency will increase for UR packets that share the same buffer as the hotspot packets. Thus, if the hotspot packets are not isolated from the background packets, the hotspot senders need to inject at a rate slightly lower than $1/N$, such as $0.99/N$. Figure 12 compares the latency-throughput curves of the background UR traffic as 4-to-1 hotspot senders are throttled to $1/4$ or $0.99/4$. Compared to the ideal routing mechanism that separates the two different traffic by providing a separate set of VCs, throttling to $1/N$ resulted in 1.6× to 2.0× higher latency across medium and high background UR traffic loads, whereas throttling to $0.99/N$ resulted in nearly ideal performance. At the same time, the accepted throughput of hotspot traffic differed by less than 1%. Thus, if the hotspot traffic is not isolated through a separate VC from other background traffic that does not cause endpoint congestion, it is desirable to throttle the hotspot senders to a rate slightly lower than $1/N$.

Meanwhile, using a shared VC can also impact the transient response with CBCM even if the impact on steady-state measurement can be minimized with a reduced throttling rate. Figure 13 shows the impact of having a dedicated, low-priority VC for isolating the identified hotspot traffic on a 256-node one-dimensional flattened butterfly network. 16 randomly selected nodes send hotspot traffic at the injection rate of 100% to a hotspot destination, whereas the remaining 239 nodes send background UR traffic among them at the injection rate of 40%. In the beginning, hotspot senders do not inject traffic, but they start injection at the 500th interval. For the configuration without a dedicated VC, once hotspot senders are identified, they are throttled to $0.99/N$, where N is the throttling degree as discussed earlier.

Providing the dedicated VC resulted in a lower peak for UR traffic latency and the latency reached a steady state approximately 13.2× faster compared to when a dedicated VC was not provided. At the same time, prioritizing UR packets over hotspot packets better prevents hotspot traffic from further spreading, and the number of in-flight hotspot packets is reduced faster. On the contrary, if the hotspot traffic shares the same VCs with UR traffic, the UR packets can lose the VC and switch allocation when they contend with the hotspot packets. In addition, sharing VCs for both traffic is not desirable since a hotspot packet at the front of a VC buffer can block UR traffic queued behind, regardless of the whether hotspot packets are deprioritized. Without the dedicated VC, the UR traffic experienced continuous jitter and slightly higher latency even after it is stabilized due to the head-of-line blocking. However, there is a trade-off between the transient behavior and the throughput of hotspot traffic. When the dedicated low-priority VC was used for the traffic, hotspot accepted throughput was reduced to 93% whereas sharing the same VCs resulted in 100% throughput.

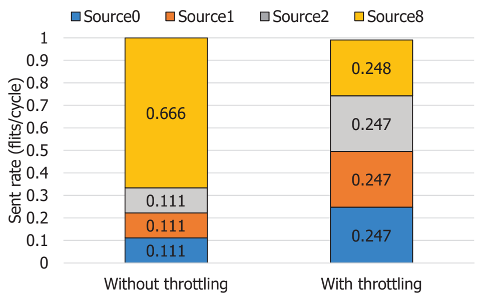


Fig. 14. Fairness among hotspot senders with and without throttling them assuming the hotspot packets are isolated with a dedicated VC.

C. Hotspot Throttling on Fairness

The CBCM introduces a separate, low-priority VC to isolates hotspot traffic as well as throttling the hotspot senders. With the separate VC, whether the hotspot senders are throttled or not does not have a significant impact on non-hotspot traffic. However, not throttling them can result in unfairness among the hotspot senders as a source closer to the hotspot can consume more bandwidth than other sources located farther away. Figure 14 shows the distribution of hotspot bandwidth among four hotspot senders with and without the source throttling, assuming a dedicated hotspot traffic VC. Three of them (node 0-2) were located in a router one hop away and one of them (node 8) was located in the same router as the hotspot (node 9) in a 16-node flattened butterfly network. Without source throttling, the node closer to the hotspot consumed 2× more bandwidth than the aggregate bandwidth consumed by the nodes one hop away. By contrast, when the hotspot senders were throttled, throughput fairness was achieved among them while the total bandwidth consumed was nearly unaffected.

VII. Related Work

Recently, there were several studies to manage the network congestion [19], [12], [13]. ECN [19] is a widely implemented mechanism in Infiniband networks [24], and different improvements have been proposed [33], [21]. Ferrer et al. [34] presented a comparison study of different ECN proposals and performed a transient hotspot study to demonstrate the response time of each methods. In that research, they showed ECN mechanism still has some negative impact on latency and throughput at the onset of the hotspot. SPR [12] was discussed in detail earlier in Section III-B. There was another research considering oversubscribed network using multiple resource reservation [13]. However, this research just considers the pre-decided hotspot channel, not all the channels that can be possibly saturated. There were other reservation protocols that affect packets that are inside the network but do not control the injection rate of the node that can be a source of hotspot [35], [36]. A recent work on reservation protocol for short messages [14] addresses possible endpoint congestion by short messages but does not address the limitation of SRP due to the speculative packets not distinguishing between endpoint and network congestions. Similar problems of many-to-one traffic have been recently identified as TCP Incast traffic [37] and different solutions have been proposed for such networks. However, we focus on high-performance interconnection network with lossless flow control where packets cannot be dropped and thus, prior work on incast traffic are not directly applicable.

Cray BlackWidow [32] network leverages both adaptive routing and deterministic routing; deterministic routing is used for some traffic in the network that requires ordering and for other traffic, adaptive routing is used within a fat-tree topology. In comparison, our work focused on how adaptive routing exacerbates endpoint congestion by spreading inadmissible traffic. There have been previous works on different adaptive routing algorithms, including improved load-balancing of non-minimal traffic [10] and incorporating the impact of the long-channel latency [11]. However, these prior works did not evaluate the impact of inadmissible traffic. Recent work exploited contention for adaptive routing in high-radix topologies [38]. However, their work was on properly identifying *network* congestion and did not explore the impact of adaptive routing on endpoint congestion. In addition, the detection of congestion is based on a threshold counter and the mechanism is not necessarily appropriate to detect endpoint congestion. It remains to be seen if this adaptive routing approach can be combined with CBCM proposed in this work.

Congestion for on-chip networks and its impact on routing has also been investigated heavily. Region congestion awareness [39] was proposed to collect congestion information from different regions of the on-chip network and provide better load-balancing. Destination-Based Adaptive Routing (DBAR) [40] proposes a low-cost congestion propagation network to provide a more accurate congestion estimation, especially within workload consolidation systems. In addition, different metrics for congestion detection have been proposed, including free virtual channel count [41], free buffer count [42], and output queue length [18]. However, none of the prior work considers the impact of congestion spreading that can occur from global adaptive routing when endpoint congestion occurs.

VIII. Conclusion

In this work, we identified the limitations of global adaptive routing if endpoint congestion occurs as global adaptive routing can spread the congestion in the network. We analyzed how prior work on congestion management, while effective in reducing endpoint congestion, can be problematic if combined with global adaptive routing that includes non-minimal routing. Thus, we propose a novel *contention-based congestion management* (CBCM) that properly differentiates between endpoint congestion and network congestion – to leverage the benefits of adaptive routing to load-balance network congestion while reducing the impact of endpoint congestion by detecting *contention* within the network. We describe how the contention is detected within the network routers but the endpoint router ultimately determines whether endpoint congestion occurred or not and sends separate control packets to notify sources contributing to the endpoint congestion. Our evaluation shows that CBCM is more effective than prior work, including speculative reservation protocol (SRP) and explicit congestion notification (ECN), when used on high-radix topologies with global adaptive routing.

Acknowledgement

We thank the anonymous reviewers for their comments. This research was supported in part by the MSIP under the Mid-career Researcher Program (NRF-2013R1A2A2 A01069132), Next-Generation Information Computing Development Program (2015M3C4A7065647) through NRF, ITRC support program (IITP-2016-H8501-16-1005) supervised by the IITP, and in part by gift from Intel.

References

[1] W. J. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. San Francisco, CA: Morgan Kaufmann, 2004.

[2] D. Abts and J. Kim, "High performance datacenter networks: Architectures, algorithms, and opportunities," *Synthesis Lectures on Computer Architecture*, vol. 6, no. 1, pp. 1–115, 2011.

[3] J. Kim, W. J. Dally, B. Towles, and A. K. Gupta, "Microarchitecture of a high-radix router," in *Proceedings of ISCA'05*, pp. 420–431.

[4] J. Kim, W. J. Dally, and D. Abts, "Flattened butterfly: A cost-efficient topology for high-radix networks," in *Proceedings of ISCA'07*, pp. 126–137.

[5] J. Kim, W. Dally, S. Scott, and D. Abts, "Cost-efficient dragonfly topology for large-scale systems," *Micro, IEEE*, vol. 29, no. 1, pp. 33–40, Jan 2009.

[6] G. Faanes, A. Bataineh, D. Roweth, T. Court, E. Froese, B. Alverson, T. Johnson, J. Kopnick, M. Higgins, and J. Reinhard, "Cray cascade: A scalable hpc system based on a dragonfly network," in *Proceedings of SC'12*, pp. 1–9.

[7] B. Arimilli, R. Arimilli, V. Chung, S. Clark, W. Denzel, B. Drerup, T. Hoefler, J. Joyner, J. Lewis, J. Li, N. Ni, and R. Rajamony, "The percs high-performance interconnect," in *Proceedings of HOTI'10*, pp. 75–82.

[8] A. Singh, "Load-balanced routing in interconnection networks," Ph.D. dissertation, Stanford University, 2005.

[9] M. Garca, E. Vallejo, R. Beivide, M. Odriozola, and M. Valero, "Efficient routing mechanisms for dragonfly networks," in *Proceedings of ICPP'13*, 2013, pp. 582–592.

[10] N. Jiang, J. Kim, and W. J. Dally, "Indirect adaptive routing on large scale interconnection networks," in *Proceedings of ISCA'09*, pp. 220–231.

[11] J. Won, G. Kim, J. Kim, T. Jiang, M. Parker, and S. Scott, "Overcoming far-end congestion in large-scale networks," in *Proceedings of HPCA'15*, pp. 415–427.

[12] N. Jiang, D. Becker, G. Michelogiannakis, and W. Dally, "Network congestion avoidance through speculative reservation," in *Proceedings of HPCA'12*, pp. 1–12.

[13] G. Michelogiannakis, N. Jiang, D. Becker, and W. Dally, "Channel reservation protocol for over-subscribed channels and destinations," in *Proceedings of SC'13*, pp. 1–12.

[14] N. Jiang, L. Dennison, and W. J. Dally, "Network endpoint congestion control for fine-grained communication," in *Proceedings of SC'15*, pp. 35:1–35:12.

[15] J. Duato, I. Johnson, J. Flich, F. Naven, P. Garcia, and T. Nachiondo, "A new scalable and cost-effective congestion management strategy for lossless multistage interconnection networks," in *Proceedings of HPCA'05*, pp. 108–119.

[16] J. Escudero-Sahuquillo, P. Garcia, F. Quiles, J. Flich, and J. Duato, "An effective and feasible congestion management technique for high-performance mins with tag-based distributed routing," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 24, no. 10, pp. 1918–1929, Oct 2013.

[17] N. McKeown, A. Mekkittikul, V. Anantharam, and J. Walrand, "Achieving 100% throughput in an input-queued switch," *Communications, IEEE Transactions on*, vol. 47, no. 8, pp. 1260–1267, Aug 1999.

[18] A. Singh, W. Dally, A. Gupta, and B. Towles, "Goal: a load-balanced adaptive routing algorithm for torus networks," in *Proceedings of ISCA'03*, pp. 194–205.

[19] J. Santos, Y. Turner, and G. Janakiraman, "End-to-end congestion control for infiniband," in *Proceedings of INFOCOM'03*, pp. 1123–1133 vol.2.

[20] J. Ferrer, E. Baydal, A. Robles, P. Lopez, and J. Duato, "A scalable and early congestion management mechanism for mins," in *Proceedings of PDP'10*, pp. 43–50.

[21] J.-L. Ferrer, E. Baydal, A. Robles, P. Lopez, and J. Duato, "Congestion management in mins through marked and validated packets," in *Proceedings of PDP'07*, pp. 254–261.

[22] G. F. Pfister and V. A. Norton, "Hot spot contention and combining in multistage interconnection networks," *IEEE Transactions on Computers*, vol. 34, no. 10, pp. 943–948, 1985.

[23] M. Karol, M. Hluchyj, and S. Morgan, "Input versus output queueing on a space-division packet switch," *Communications, IEEE Transactions on*, vol. 35, no. 12, pp. 1347–1356, Dec 1987.

[24] "Infiniband architecture specification," volume 1, release 1.2.1, Infiniband trade association. [Online]. Available: http://www.infinibandta.org.

[25] W. J. Dally, "Virtual-channel flow control," in *Proceedings of ISCA'90*, pp. 60–68.

[26] N. Jiang, D. Becker, G. Michelogiannakis, J. Balfour, B. Towles, D. Shaw, J. Kim, and W. Dally, "A detailed and flexible cycle-accurate network-on-chip simulator," in *Proceedings of ISPASS'13*, pp. 86–96.

[27] H. Adalsteinsson, S. Cranford, D. A. Evensky, J. P. Kenny, J. Mayo, A. Pinar, and C. L. Janssen, "A simulator for large-scale parallel computer architectures," *Int. J. Distrib. Syst. Technol.*, vol. 1, no. 2, pp. 57–73, Apr. 2010.

[28] A. L. McPherson, H. Dong, M. Ravishankar, P. Sathre, M. B. Sullivan, W. Taitano, J. A. Willert, T. C. Germann, D. A. Knoll, B. R. Lally, P. S. McCormick, , and S. D. Pakin, "Quasi diffusion accelerated monte carlo," Tech. Rep., 2011, Los Alamos National Laboratory.

[29] D. F. Richards, J. N. Glosli, B. Chan, M. R. Dorr, E. W. Draeger, J.-L. Fattebert, W. D. Krauss, T. Spelce, F. H. Streitz, M. P. Surh, and J. A. Gunnels, "Beyond homogeneous decomposition: Scaling long-range forces on massively parallel systems," in *Proceedings of SC'09*, Article 60.

[30] J. Bell, A. Almgren, V. Beckner, M. Day, M. Lijewski, A. Nonaka, and W. Zhang, "Boxlib user guide," 2013, Center for Computational Sciences and Engineering, Lawrence Berkeley National Laboratory.

[31] "Characterization of the doe mini-apps," National Energy Research Scientific Computing Center. [Online]. Available: http://portal.nersc.gov/project/CAL/doe-miniapps.htm.

[32] S. Scott, D. Abts, J. Kim, and W. J. Dally, "The blackwidow high-radix clos network," in *Proceedings of ISCA'06*, pp. 16–28.

[33] G. Pfister, M. Gusat, W. Denzel, D. Craddock, N. Ni, W. Rooney, T. Engbersen, R. Luijten, R. Krishnamurthy, and J. Duato, "Solving hot spot contention using infiniband architecture congestion control," *Proceedings of HP-IPC'05*.

[34] J.-L. Ferrer, E. Baydal, A. Robles, P. López, and J. Duato, "On the influence of the packet marking and injection control schemes in congestion management for mins," in *Proceedings of Euro-Par'08*, pp. 930–939.

[35] L.-S. Peh and W. Dally, "Flit-reservation flow control," in *Proceedings of HPCA'00*, pp. 73–84.

[36] Y. H. Song and T. Pinkston, "Distributed resolution of network congestion and potential deadlock using reservation-based scheduling," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 16, no. 8, pp. 686–701, Aug 2005.

[37] Y. Chen, R. Griffith, J. Liu, R. H. Katz, and A. D. Joseph, "Understanding tcp incast throughput collapse in datacenter networks," in *Proceedings of WREN'09*, pp. 73–82.

[38] P. Fuentes, E. Vallejo, M. Garcia, R. Beivide, G. Rodriguez, C. Minkenberg, and M. Valero, "Contention-based nonminimal adaptive routing in high-radix networks," in *Parallel and Distributed Processing Symposium (IPDPS), 2015 IEEE International*, May 2015, pp. 103–112.

[39] P. Gratz, B. Grot, and S. Keckler, "Regional congestion awareness for load balance in networks-on-chip," in *Proceedings of HPCA'08*, pp. 203–214.

[40] S. Ma, N. Enright Jerger, and Z. Wang, "Dbar: An efficient routing algorithm to support multiple concurrent applications in networks-on-chip," in *Proceedings of ISCA'11*, pp. 413–424.

[41] W. Dally and H. Aoki, "Deadlock-free adaptive routing in multicomputer networks using virtual channels," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 4, no. 4, pp. 466–475, Apr 1993.

[42] J. Kim, D. Park, T. Theocharides, N. Vijaykrishnan, and C. Das, "A low latency router supporting adaptivity for on-chip interconnects," in *Proceedings of DAC'05*, pp. 559–564.