Channel Reservation Protocol for Over-Subscribed Channels and Destinations

George Michelogiannakis
Stanford University / Lawrence Berkeley National Laboratory
mihelog@stanford.edu

Nan Jiang
Stanford University
njiang37@stanford.edu

Daniel Becker
Stanford University
dub@stanford.edu

William J. Dally
Stanford University / NVIDIA Research
dally@stanford.edu

ABSTRACT

Channels in system-wide networks tend to be over-subscribed due to the cost of bandwidth and increasing traffic demands. To make matters worse, workloads can overstress specific destinations, creating hotspots. Lossless networks offer attractive advantages compared to lossy networks but suffer from tree saturation. This led to the development of explicit congestion notification (ECN). However, ECN is very sensitive to its configuration parameters and acts only after congestion forms. We propose channel reservation protocol (CRP) to enable sources to reserve bandwidth in *multiple* resources in advance of packet transmission and with a *single* request, but without idling resources like circuit switching. CRP prevents congestion from ever occurring and thus reacts instantly to traffic changes, whereas ECN requires 300,000 cycles to stabilize in our experiments. Furthermore, ECN may not prevent congestion formed by short-lived flows generated by a large combination of source–destination pairs.

General Terms

Congestion control, congestion notification, large-scale networks, tree saturation, reservation protocol

1. INTRODUCTION

Due to the ever-increasing storage and computation demands, today's high performance computing (HPC) and datacenter networks are often composed of tens of thousands of computation units [44, 31, 17]. This has led major companies to operate more than one datacenter or HPC network clusters, scattered around the world as in cloud networks [23], or co-located in large datacenters [7]. Each cluster may require very high bandwidth communication to other clusters [32], stressing the costly inter-cluster links.

In such large scales, the primary constraint for application performance can easily become network bandwidth because of the cost of high-radix routers as well as optical and electrical channels [3]. This problem is made worse by application behavior which often results in ill-behaved traffic patterns that over-subscribe parts of the network or certain destinations. For example, MapReduce [12] performs considerable data shuffling before advancing to the reduce phase after the map phase. In addition, web search utilizes many leaf nodes to perform lookups on part of the search index. After performing their lookup, all leaf nodes then all report their results to the same front end node [33].

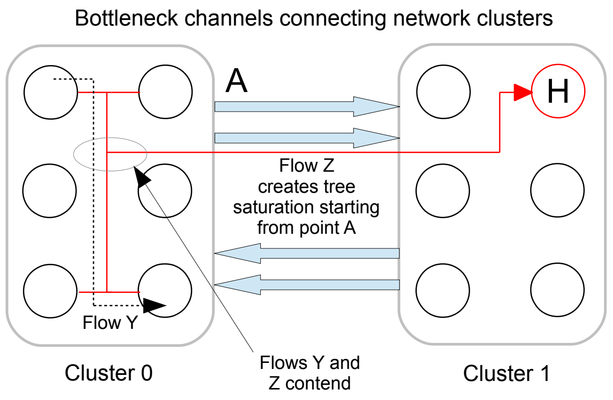

Bottleneck channels connecting network clusters
A
H
Flow Z creates tree saturation starting from point A
Flow Y
Cluster 0
Flows Y and Z contend
Cluster 1

Figure 1: Points A and H can be over-subscribed and cause tree saturation. Flow Y is affected by Z's tree saturation.

In such networks, flow control is a major topic of research. Lossless flow control benefits performance, quality of service (QoS), and predictability over lossy (dropping) flow control networks, such as with Ethernet and TCP [24]. Lossy networks also typically have no router-to-router feedback mechanisms, which limits their adaptive routing capability [20, 24, 4]. Despite these benefits, lossless flow control suffers from congestion which can form multi-hop paths of blocked packets affecting benign flows, known as tree saturation [30, 40]. In contrast, lossy networks do not experience tree saturation because packets are dropped and admission is controlled by mechanisms such as datacenter TCP [5].

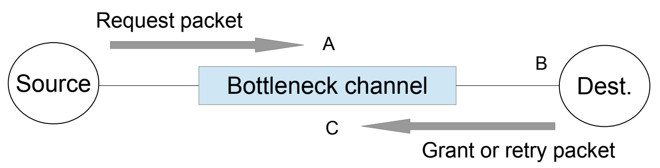
Figure 1 illustrates an example network with two clusters. Each cluster offers full bisection bandwidth internally such that adaptive routing or flow scheduling algorithms can evenly distribute traffic load [20, 4]. However, due to cost and technology constraints, inter-cluster channels are often under-provisioned [32]. In this figure, flow Z consists of several sources in cluster 0 transmitting at full rate to hotspot destination H. Flow Z will create tree saturation at point A if the inter-cluster channels do not have the bandwidth to support flow Z's traffic, and at hotspot destination H if H cannot eject traffic at the same rate as it arrives. Because of tree saturation starting from point A, flow Z affects a benign flow Y which remains internal to cluster 0. Flow Z also affects flows internal to cluster 1 due to tree saturation originating from destination H. This is true even if the benign flows do not use sources and destinations participating in flow Z, because tree saturation can be widespread. Even though other flows contending for bandwidth with flow Z at points A or H inevitably have their performance reduced because the network cannot satisfy the demand, congestion control should leave independent benign flows unaffected by ill-behaved flows.

Research has proposed numerous congestion control and prevention mechanisms [8, 46]. The most popular of these mechanisms is explicit congestion notification (ECN) [25]. ECN detects congestion at the root of the congestion tree and signals to the contributing sources to throttle down via a control packet or by piggybacking onto acknowledgment packets. While ECN can be effective [37, 21], it is also considerably sensitive to its configuration parameters which depend on numerous and sometimes unpredictable factors, including the traffic pattern [37]. Past work has also highlighted ECN's slow adaptation to changes in the traffic pattern [15, 28]. This is based on ECN's *reactive* nature, since ECN acts only after congestion forms, which is also true for TCP.

To mitigate ECN's limitations, researchers have proposed speculative reservation protocol (SRP) [28]. SRP reserves bandwidth in destinations in advance of the payload of long flows. This prevents destinations from becoming hotspots, even momentarily. To mitigate the round-trip latency overhead of reservations, flows transmit packets speculatively with low priority and without waiting for a grant. Even though SRP prevents tree saturation due to hotspot destinations, it does not prevent congestion due to over-subscribed network channels. Therefore, in Figure 1, SRP would be ineffective against tree saturation at point A.

In this paper, we propose channel reservation protocol (CRP). CRP extends SRP to eliminate tree saturation in lossless networks by enabling flows to reserve *multiple* resources with a *single* request. CRP makes lossless networks more attractive in the datacenter or HPC environment because tree saturation is a significant drawback of lossless flow control [24, 17]. CRP can be applied in numerous settings, such as networks with more intra-rack bandwidth than inter-rack bandwidth [2]. However, we focus on multiple network clusters where each cluster offers full bisection bandwidth internally, but inter-cluster channels are over-subscribed. This setting represents datacenters in different geographical locations, or network clusters of the same datacenter [32, 7, 23]. In this setting, resources that require a reservation are destinations (to prevent hotspots) and inter-cluster channels. CRP differs with circuit switching [46] because it avoids idling resources due to the round-trip delay between circuit set up and data traversal [29].

Due to its proactive nature, CRP reacts instantaneously to congestion whereas ECN requires 300,000 cycles to restore throughput and latency after a change in traffic in our experiments. Furthermore, ECN may not address congestion due



A: Records in the request's reservation vector what cells the channel has no availability for.
B: Selects a cell that is available in its own and the request's reservation vector and issues a grant. Otherwise it issues a retry.
C: If the response is a grant, marks the allocated bandwidth as unavailable to other flows.

Figure 2: An overview of CRP.

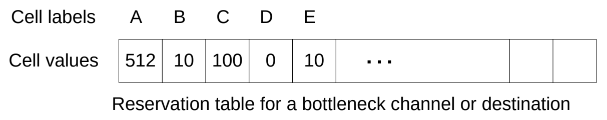


Figure 3: Each cell corresponds to a specific future time slot and records the available bandwidth in cycles.

to many short flows that share a bottleneck channel, such as occurs with small flows of uniform random (UR) traffic. In that case, each individual flow terminates before ECN takes effect, but the aggravated affect of multiple flows still causes congestion. In such cases, with CRP sources sustain 5× higher accepted traffic in our experiments. Finally, our results show that CRP is reasonably insensitive to network configuration and traffic pattern, whereas ECN's configuration depends considerably on those parameters. However, CRP can cause a significant bandwidth overhead for control packets under some traffic patterns, compared to ECN.

2. CHANNEL RESERVATION PROTOCOL

With CRP, each over-subscribed channel and destination maintains a reservation table. Each cell in the table represents a future time slot and records the available bandwidth in clock cycle (flit) granularity. Sources send requests which record the availability of participating resources. Destinations then calculate the earliest common availability that satisfies the request size, and respond with a grant. Grants then finalize the reservations in participating resources while in transit back to the source, as shown in Figure 2.

2.1 Reservation Tables

We associate a reservation table with each resource. Each cell of a table corresponds to one time slot and the value in that cell represents the number of cycles of bandwidth available in the corresponding time slot. A sample configuration of 32 table cells (V_{cells}), each accounting for 512 cycles (C_{max}), is illustrated in Figure 3. In this example, resources can be reserved up to 16384 cycles into the future. In the figure, cell *A* corresponds to the time slot comprising cycles 0-511, cell *B* represents time slot of cycles 512-1023, and so on. As shown, time slot (cell) A has 512 available cycles while time slot B has only 10.

A resource can accommodate a request of size x during time slot i if $t[i] > 0$ and $t[i] + t[i+1] \geq x$. That is, the corresponding table cell must have at least one free clock cycle and the sum of that table cell and the next cell must be at least the request size. This allows requests to straddle two adjacent cells, reducing the adversary effects of fragmentation.

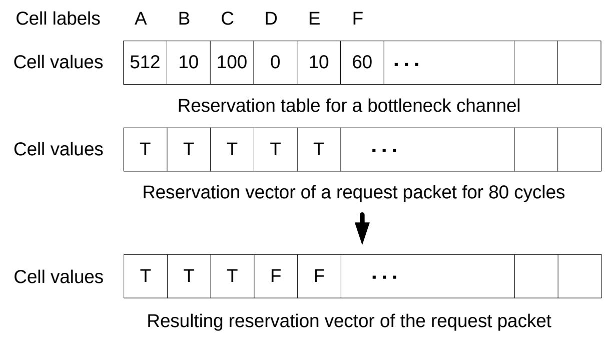

Cell labels A B C D E F
Cell values 512 10 100 0 10 60 ...
Reservation table for a bottleneck channel
Cell values T T T T T ...
Reservation vector of a request packet for 80 cycles
Cell values T T T F F ...
Resulting reservation vector of the request packet

Figure 4: Elements in the request's bit vector get deasserted in any channel that cannot accommodate the requested size.

Reservation tables are logically shifted by one cell when time advances by C_{max} cycles. In the above example, when system time becomes 512, cell A is deleted and the table is shifted left by one. A new cell with value C_{max} is inserted at the right. To maintain synchronization, all resources must be synchronized to a global time base with an accuracy of at least $\pm C_{max}$ cycles using a technique such as [9].

Reservation tables are maintained for each critical resource. Each destination's network interface card (NIC) contains a table for that destination's *egress* channel. Each router maintains a table for each output port that drives a potential bottleneck channel, e.g., an inter-cluster channel.

2.2 Reservation Handling in Channels

Reservation requests carry a bit vector of length V_{cells} where each bit signals the availability of a time slot. Request vectors are initialized to all *true*. Each time a request vector passes a resource, that resource resets any bits of the vector that correspond to time slots for which the resource does not have the requested bandwidth. In effect, each resource *ANDs* its availability into the request vector as it passes. When the request vector reaches the destination, it indicates the time slots for which all required critical resources had sufficient bandwidth to handle the request *at the time the request passed the resource.*

Reservation vectors are shifted left by one bit at the same time as reservation tables. When reservation vectors are shifted, a *true* bit is inserted at the right end because resources are initially available during a new time slot. Reservation vector shifting is performed by routers.

Consider Figure 4. In this example, cells A and C remain *true* because the table can accommodate the request of 80 cycles in those time slots. Cell D sets its bit to *false* because it has no bandwidth remaining, and cell E sets its bit to *false* because the sum of cells E and F is insufficient for the request. Finally, cell B remains *true* because 80 cycles can be accommodated by the sum of cells B and C.

2.3 Reservation Handling in Destinations

Destinations compare the reservation vectors of arriving requests against their own tables and compute the earliest time slot that all participating resources on the request's path, including the destination, can accommodate. Similar to vector handling in channels, a time slot is considered available at the destination if it has an available cycle and the sum of its and the next cell's available bandwidth satisfies the incoming request. After *ANDing* their availability

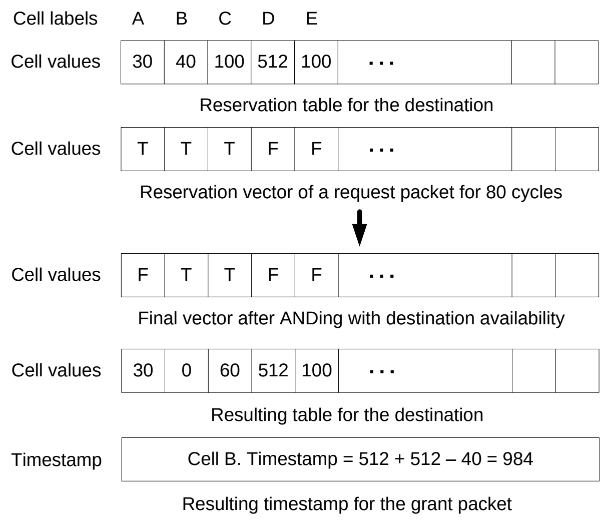

Cell labels A B C D E
Cell values 30 40 100 512 100 ...
Reservation table for the destination
Cell values T T T F F ...
Reservation vector of a request packet for 80 cycles
Cell values F T T F F ...
Final vector after ANDing with destination availability
Cell values 30 0 60 512 100 ...
Resulting table for the destination
Timestamp
Cell B. Timestamp = 512 + 512 – 40 = 984
Resulting timestamp for the grant packet

Figure 5: Destinations compare their availability against request vectors and produce mutually-acceptable timestamps.

into the request vector, destinations generate a grant response that carries a timestamp corresponding to the left-most *true* bit of the final vector. This is the earliest time that all required resources are available.

This operation is illustrated in Figure 5. In this example, cell A cannot accommodate the request at the destination because the sum of cells A and B in the destination's reservation table is less than the requested 80 cycles. However, cell B can accommodate the request because cells B and C combined have 140 free cycles. Cell C is also able to accommodate the request. We choose cell B as the earliest possible that can handle the request and decrement the table entries for cells B and C to reserve the 80 cycles (40 from each cell). The actual reserved period straddles the two slots comprising the last 40 cycles of slot B and the first 40 cycles of slot C. The timestamp is computed to be 984 which corresponds to the first cycle of the last 40 cycles of slot B.

2.4 Grant and Retry Operation

As grant packets travel back to the source, they decrement the appropriate cells of each reservation table along their route to mark the allocated bandwidth as unavailable to other flows (point C of Figure 2). If at that time the cell corresponding to the timestamp does not have enough cycles to satisfy the request, any remaining slots are decremented from the next cell. In the example of Figure 5, the destination produced a grant timestamp for cycle 984 for a reservation size of 80 cycles. When that grant reaches the channel that has the reservation table shown in Figure 4, cell B corresponds to cycle 984 and thus will be decremented by 10 and set to 0, whereas cell C will be decremented by the remaining 70 cycles and thus be set to 30.

If the two table cells no longer have sufficient cycles to satisfy the request, the grant is converted to a retry response. This can occur if the bandwidth was reserved by another flow during the time it took for the reservation request to reach its destination, generate a grant, and return to the participating channel (from point A until point C in Figure 2). A retry is also issued by a destination if the reservation request's final vector has no *true* bits or a common availability does not exist. A retry instructs the source to retransmit its request after a short delay (R_{cycles}). The only exception to this is when a retry is issued because a destination's reservation table has no availability for the specified reservation size in any time slot and regardless of the reservation request's vector. In that case, the retry instructs the source to retransmit its request after $V_{cells} \times C_{max}$ cycles, minus the round-trip delay.

When a grant is converted to a retry at a channel, reservations in the destination and channels the response has already traversed are not cancelled. That bandwidth is efficiently taken advantage of by speculative packets (explained below) and control packets, which is confirmed by our experiments. Having heavily oversubscribed resources closer to traffic sources increases the probability that bandwidth will be unnecessarily allocated in this manner in downstream resources.

To prevent retries due to multiple flows attempting to reserve the same time slot, reservation requests have the option of eagerly reserving the earliest time slot when they traverse a participating channel (point A of Figure 2). Destinations will preferentially grant this pre-reserved time slot if available. Reservation grants release any ungranted eager reservations for the grant's flow. This requires tag matching since eager reservations are associated with flow identifiers. Eagerly reserving more than one time slots (R_{res}) for a single request increases the probability that destinations will grant a pre-reserved slot, but also increases the probability of idling bandwidth even though there are eligible requests.

Grants and retries are forced to return using the reverse of the request path so they can manipulate the correct channel reservation tables. The reverse path is identified either by having request packets record their path, or by simple calculations in the case of deterministic routing.

2.5 Protocol Considerations

The size of a reservation (x) is constrained to be in the range $x \in [N_{min}, N_{max}]$. Also, because reservation requests consider availabilities in only up to two adjacent cells, N_{max} must be no greater than $2C_{max}$. The minimum reservation size (N_{min}) is chosen large enough to amortize the latency and bandwidth overheads of request and grant packets. Flows smaller than N_{min} do not use CRP. The maximum reservation size (N_{max}) is chosen small enough so that maximum-sized reservations do not create fairness or starvation issues. Flows larger than N_{max} are divided into segments of N_{max} in length. Each segment of a large flow participates in CRP independently.

Even though small flows do not participate in CRP, they consume bandwidth and thus must be taken into account. Otherwise, tree saturation can form due to the combined load of participating and non-participating flows. To mitigate this, each resource decrements the number of cycles in its earliest non-zero reservation cell by the size of each non-participating data packet that traverses the resource. This approach removes bandwidth used by non-participating packets from the reservation pool—albeit with a slight delay. CRP also accounts for the bandwidth consumed by control packets by reserving ϵ additional bandwidth for each request. For example, if we assume a 5% control bandwidth overhead for each flow, we set ϵ to 5% such that CRP treats reservation requests as if they were 5% larger.

Systems with predominantly flows smaller than N_{min} will render CRP ineffective. In that case, CRP can be applied to a *collection* of flows to a common destination or to a group of destinations. For example, CRP can be applied to all flows from the same source destined to the same network cluster because those flows will content for the same over-subscribed inter-cluster channels.

Parameters V_{cells} and C_{max} determine the size of our reservation window and the granularity of allocation. Reservation tables must be large enough such that issuing a retry due to a full reservation schedule at a destination is rare, unless the corresponding resource is over-subscribed in which case its reservation table will become and remain full regardless of its size. Vector sizing should also reduce the problem of fragmentation, where a resource is unable to produce a grant because the available bandwidth is scattered along multiple non-adjacent time slots. The choice of C_{max} also needs to consider N_{min} and N_{max} so that each table cell is able to service at least one request in full and so that leaving counts smaller than N_{min} is unlikely. For example, with an anticipated request size of 256 cycles (flits), a viable choice for C_{max} is $256 \times (1 + \epsilon)$ which enables each cell to service one full request with overhead.

In addition, parameter C_{max} must be small enough to bound the timing uncertainty of each reservation. The fact that a grant may use cycles from two adjacent table cells adds an uncertainty of C_{max} between the timestamp and the actual reserved cycle. Providing an upper bound for this uncertainty is the reason we restrict reservations to using cycles between adjacent slots only. Furthermore, a destination may generate a grant for the first cycle of the chosen time slot even though a participating channel may have used its last remaining cycles in that time slot. Reservation bit vectors do not record this information to avoid the associated overhead. This adds further uncertainty of C_{max}, for a total uncertainty of $2C_{max}$.

An excessively large C_{max} reduces performance if the timing uncertainty of each reservation becomes comparable to the available buffering in each router, in which case tree saturation may form from granted flows. Finally, the number of bits reservation requests carry ($V_{cells} \times C_{max}$) should allow reservation requests to remain single-flit packets.

CRP uses speculative packets to mitigate the request–grant round-trip latency under low and medium loads, as proposed by SRP [28]. Each flow initiates packet transmission speculatively without a grant. Speculative packets have a limited time to wait (TTW) and are dropped at routers if their cumulative buffering time in the network becomes larger. The effect of TTW in the speculative packet drop probability has been studied in [28]. Dropped packets cause negative acknowledgment (NACK) packets. NACKs cause the source to stop the speculative transmission, transmit a reservation request, and retransmit the dropped packets as normal packets upon receiving a grant. A speculative packet that successfully reaches its destination creates an acknowledgment (ACK) packet. In CRP, speculative packets provide the added benefit that bandwidth left unreserved due to limitations such as fragmentation is productively used by speculative packets.

CRP uses four virtual channels (VCs) [11]. The first VC is for reservation requests. The second VC is for response control packets (grants, retries, ACKs and NACKs). The third VC is for speculative packets, and the fourth VC for packets from granted flows or small flows not participating in CRP. The VCs for control packets have the highest priority, while the VC for speculative packets has the lowest priority. Control VCs have a reduced implementation cost compared to data VCs due to their typically low utilization factor and small packet size. Note that packet types can share VCs. However, we use four VCs to optimize performance. In networks that require multiple VCs per traffic class for performance or to prevent deadlocks, the control and speculative VCs can be shared by all data VCs.

2.6 Starvation and Traffic Classes

To avoid starvation where one flow constantly receives retry responses, we can randomize the time that sources wait for until they resubmit their requests after a reply, as well as increase the number of slots flows eagerly reserve according to age. For example, flows can eagerly reserve one more slot in every resource for each retry response beyond a predefined number. Eventually, a request will eagerly reserve the maximum amount of slots in each resource. While this dramatically increases the probability for a grant, it does not guarantee that slots will be available in every resource. To mitigate this, flows can be given the option of moving other reservations to another free slot, or preempting other reservations. In addition, fairness and QoS can be implemented by dividing reservation vectors among traffic classes, to guarantee bandwidth per traffic class.

3. EVALUATION METHODOLOGY

We modified Booksim [27] to compare a baseline network with VC [11] cut-through flow control without congestion control, a network with Infiniband-style ECN [37, 39, 25], and a network with CRP. We compare against ECN because it is a primary congestion control technique for lossless networks and has been applied to lossy networks [5, 38]. We do not show results for SRP because SRP does not prevent in-network congestion and thus is comparable to the baseline network in our experiments. We also do not compare against lossy networks because fairly comparing lossy and lossless flow control is outside the scope of this paper.

In the baseline network, a single VC carries all traffic. ECN uses an additional control VC for congestion notification packets. CRP uses four VCs as described in Section 2.5. In all networks, the VC for normal (non-speculative) packets uses virtual output queueing (VOQ) to eliminate head-of-line blocking (HOL). In large-radix routers, VOQ can be implemented cost-effectively with roughly the same throughput [34]. All other VCs use FIFO input buffers. Each input buffer has a capacity of 4KB statically assigned to each VC. The router crossbar has a 2× speedup over channels. Outputs have buffers of 512B per VC. Switches use separable input-first allocation with priority arbiters.

We model clusters of networks with oversubscribed inter-channel channels [32]. Each network cluster consists of a 144-node Fat Tree [36] topology with two levels. The first level consists of 12 routers. Each router has 12 down and 12 up channels. The second level consists of another 12 routers with 12 down channels each. Routing follows nearest common ancestor with a random choice of channels when traversing up the tree. We use two network clusters interconnected by four channels in each direction. When traversing to a different cluster, packets cross a dateline and thus use a different set of VCs. Only destinations and inter-cluster channels participate in CRP. Inter-cluster channels are connected to leaf routers. Routers with an inter-cluster channel have 13 down and 13 up channels, instead of 12. All channels in the network have a capacity of 10Gb/s and a latency of 32ns, except for channels between clusters which have a latency of 64ns. Routers operate at 1GHz; thus, the zero-load latency through each router is 26ns [1].

Packet size is set to 256B, divided into 32 8B flits. Control packets consist of one 8B flit. Sources generate traffic in units of messages (flows). By default, message size is set to 8 packets (2KB). To mitigate HOL blocking between flows to different destinations, sources create an independent flow queue for each destination, similar to Infiniband [25]. Flow queues arbitrate for injection in a round-robin fashion. CRP does not depend on a specific number of flow queues.

To clearly illustrate the presence or absence of tree saturation and because CRP can hide the request–grant latency with speculative packets under low to medium loads (resulting in low injection latencies), we focus on in-network latency. In addition, due to the over-subscription ratio, even low injection rates exceed available network bandwidth. Therefore, injection queues constantly grow and therefore total packet latency (including injection) constantly increases. For injection rates lower than that, CRP has similar average injection latency with a properly-configured ECN under stable traffic. Our results show offered traffic per source (injection rate) versus minimum or average accepted data traffic among all sources that participate in the traffic pattern.

We simulate realistic traffic observed in HPC systems and datacenters [13, 33] with our combined traffic pattern which uses background hotspot traffic where half of the sources in each cluster transmit at full rate to a hotspot in each network cluster (each source transmits to all hotspots). In this traffic pattern we collect statistics for benign foreground intra-cluster UR traffic where every source (including the ones participating in the hotspot pattern) transmits at a variable rate only to destinations in the same cluster. The hotspot traffic represents the small number of nodes that communicate with most of the machines in the system (and across clusters), because they are running the task scheduler, aggregators or monitoring systems [13]. The background hotspot traffic can also represent the collection of intermediate web search results using parts of the search index, performed by numerous leaf nodes and sent to few front end web server nodes, or reduction operations with constructs such as MPI [33]. The benign foreground intra-cluster UR pattern represents multi-purpose traffic that is confined within the same or consecutive clusters or racks [13]. It is this traffic that should remain unaffected by the ill-behaved hotspot traffic.

The default ECN and CRP configuration parameters are shown in Table 1. These parameters were chosen after experiments in order to optimize throughput for the combined traffic pattern for each network and according to the suggestions in [37]. V_{cells} was set to 32 in order to keep reservation requests single-flit packets. Furthermore, C_{max} is large enough to reduce fragmentation by being able to accommodate a single 256B message in each cell, including ϵ.

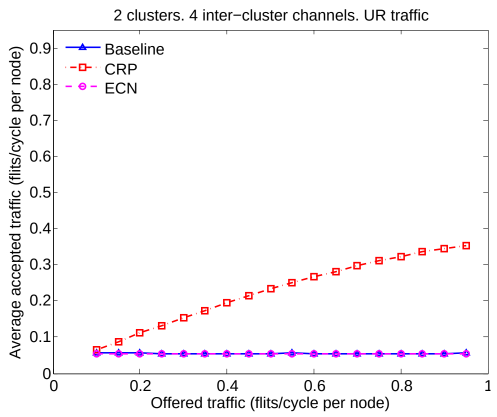
4. EVALUATION

4.1 Single Traffic Class

For our initial evaluations, we use traffic patterns simpler than the combined traffic pattern explained in Section 3. Specifically, we conduct experiments using either global UR

Table 1: Default configuration parameters for ECN and CRP networks.

Protocol	Parameter	Description	Value
CRP	ϵ	Reservation overhead adjustment	5%
	TTW	Speculative packet time to wait	200 cycles
	N_{max}	Maximum reservation size	16 packets
	N_{min}	Minimum reservation size	4 packets
	V_{cells}	Reservation table and vector cells	32
	C_{max}	Reservation counter maximum value	269 cycles
	R_{res}	How many request sizes reservation requests eagerly reserve	1
	R_{cycles}	Cycles to wait after a retry	30
ECN	IPD_+	Inter-packet delay increment	9600ns
	IPD_-	Inter-packet delay decrement	100ns
	T_-	Inter-packet delay decrement timer	4μs
	C_{thres}	Congestion threshold	95% input buffer capacity



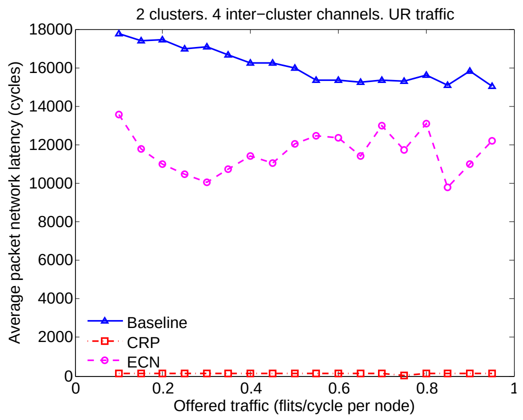


Figure 6: Performance under global uniform random traffic.

traffic or the hotspot pattern which is part of our combined traffic pattern. In all our evaluation results in this paper, SRP is comparable to the baseline network and is thus not shown, because SRP only prevents congestion due to destination hotspots, not inter-cluster channels [28]. In this section, simulation cycles were reduced to keep simulation time constant as injection rate increases, which causes a slight artificial drop in reported average latency and retry probability for higher injection rates.

With hotspot traffic, all three networks have an equal average accepted rate of 0.01 flits per cycle per node. This is simply the available bandwidth of the bottleneck links. CRP has very little variation in sender throughput with a standard deviation 2.5× smaller than that of ECN and 14× smaller than that of baseline. CRP increases fairness in cases where there are at least two resources to reserve, such as our experiments. This is because even though a node may be close to one resource, it will not be close to the second resource, compared to other nodes. That second resource will keep the node from dominating the first resource, since reservations are for the same time slot in both resources.

Figure 6 presents evaluation results for global UR traffic where each source selects among destinations across clusters with equal probability. This pattern represents traffic from load-balanced applications which have to be mapped to a system that does not provide sufficient bandwidth between all source–destination pairs. As shown, ECN and baseline have very high network latency, indicative of tree saturation. At 95% flit injection rate per node, CRP sources have a 5× larger average accepted rate than ECN. Even with CRP, flows that need to traverse an inter-cluster channel are restricted to 0.05 flits per node per cycle because of the 18:1 over-subscription ratio of inter-cluster channels; this reduces the reported accepted rate of global UR traffic even though intra-cluster flows are unaffected by inter-cluster flows. Put differently, because each source also transmits to destinations in the same cluster and CRP eliminates tree saturation, the average and minimum accepted rates per source are higher than 0.05 for CRP. In contrast, ECN and baseline do not exceed 0.05 due to widespread tree saturation. CRP has a standard deviation of 6.5% of its average accepted rate. This is because flows traversing inter-cluster channels need to reserve two resources and thus have a higher probability to receive a retry response. ECN has a larger standard deviation of 12% of its average accepted rate.

In this testcase, ECN is ineffective because when ECN signals congestion, it throttles down the flow that receives the congestion notification. However, by the time the congestion notification arrives, the offending flow has already completed because it is short-lived. The congestion information remains in the source, but since a flow is defined as a source–destination pair, this information does not affect packets to other destinations (even if its stressing the same inter-cluster channels) from the same source. ECN does maintain state at every source for each destination, but in this experiment, by the time a source selects the same destination, the ECN timers have reset and therefore the source is allowed to transmit at full rate again. In our experiments, sources received only 3 congestion notifications per destination in 150,000 cycles. Therefore, ECN does not prevent congestion due to a combination of short-lived flows to different destinations.

While the above behavior for ECN is based on the par-

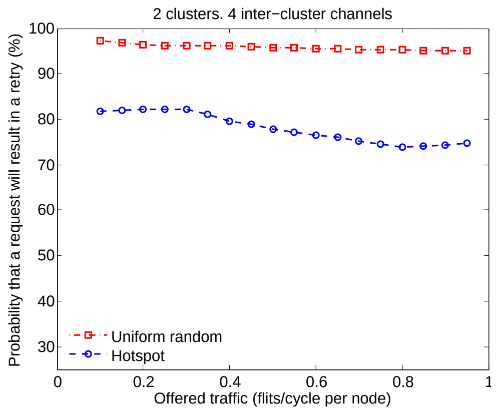


Figure 7: The average probability that any reservation request will result in a retry response.

ticular implementation and its use of timers, the same flaw can exist in improved ECN schemes as well as TCP due to the lack of feedback from the network and the minimum window size. In addition, schemes which initiate a new connection (flow) with default parameters irrespective of past statistics, will also show this weakness with starting values of more than 0.05 flits per cycle per node in our experiments. Even a minimum window size of a single packet is enough to overwhelm the inter-cluster channels in these experiments, because at any one time 72 sources will be sending a single packet through the inter-cluster channels. Randomized backoff times will also not resolve this issue unless they are exceptionally large, because by the time sources generate a new flow to the same destination, a significant amount of time has passed. Even if a flow halts transmission, sources have other flows for the same cluster waiting for injection and by the time the round-robin selector revisits the halted flow, a significant amount of time has passed and the randomized backoff has expired. The key weakness of such schemes is that state is maintained per destination, and not per network cluster or group of destinations sharing an over-subscribed resource. Therefore, multiple short-lived flows do not share congestion information. CRP solves this problem without grouping destinations, because doing so introduces non-trivial dependencies between groups, because groups may share any number of over-subscribed resources.

CRP's throughput under global UR traffic is limited by the high number of retry responses. Specifically, reservation requests (not necessarily initial requests) have a 95% probability of resulting in a retry response under a 95% flit injection rate per node. This means that on average a reservation request will have to be sent 18 times before generating a grant response. The probability across injection rates is shown in Figure 7. Retry probabilities do not vary significantly with the injection rate because inter-cluster channels are over-subscribed for injection rates more than 0.1 flits per node per cycle. These probabilities are averaged across inter-cluster and intra-cluster flows.

Retry responses occupy bandwidth. Under a 95% flit injection rate per node, the bandwidth overhead of control packets is 14% for CRP. Since an inter-cluster flow can send to any destination in the other cluster, there is a significant probability that two flows will get granted the same time slot from different destinations; one of those grants will be converted to a reply in the inter-cluster channel the two flows share. The control packet overhead for ECN is 1%.

Under hotspot and a 95% flit injection rate per node, CRP has a 75% probability that a request will result in a retry, a 4% bandwidth overhead. However, this doesn't penalize CRP's performance because retries under hotspot are due to full destination reservation tables instead of failing to find a common availability. This is unavoidable due to the over-subscription ratio, regardless of the reservation table size.

The high control bandwidth overhead for CRP illustrates the adversary effects of making allocation decisions without global knowledge. This way, flows which need to reserve multiple resources are more likely to receive a retry response, due to flows reserving the same time slot. Increasing R_{res} may reduce the number of retries. In contrast, flows to the same cluster will never receive a retry response unless their destination's reservation table is full.

4.2 Hotspot Traffic Affecting Benign Traffic

Figure 8 shows the performance of the foreground intra-cluster UR traffic in our combined pattern. This experiment shows how benign intra-cluster traffic is affected by adversarial traffic overstressing resources. ECN and CRP can accept all offered foreground traffic, on average. CRP offers a 3.5% lower minimum throughput due to the extra bandwidth overhead for its control messages compared to ECN. ECN achieves high performance because inter-cluster channels are stressed by steady-state long flows from a few sources to only a single hotspot destination per cluster, in contrast with Figure 6. The baseline and SRP networks perform similarly to the baseline in Figure 6 and are thus not shown.

The retry probability for CRP is only 1.3% at a 95% flit injection rate per node because inter-cluster requests are destined to the same hotspot destination. Therefore, the only case that a retry is produced, except for reservation tables having no vacancy, is if a flow eagerly reserves a time slot in an inter-cluster channel that another flow gets granted from the destination. That grant then gets converted to a retry when it traverses the inter-cluster channel back to the requesting node. The retry probability increases with the number of hotspot nodes in each cluster, because then different destinations can grant flows for the same time slot.

Moreover, ECN has a 15% higher packet network latency by average for injection rates larger than 10%. Also, ECN's maximum latency is 4391 cycles at a 95% flit injection rate per node, while CRP's is 4913 cycles. The minimum latency for both networks is 39 cycles. CRP has a higher maximum latency due to rare occurrences where short-lived contention forms due to the timing uncertainty of reservations, discussed in Section 2.5. This, however, does not degrade throughput because the buffers absorb the contention. In contrast, ECN relies on congestion momentarily forming to generate congestion notifications to throttle down sources. Therefore, congestion is much more frequent with ECN because it is part of ECN's normal operation, which leads to its 71% higher average network latency.

Another factor for ECN's higher average latency is that in our traffic pattern the two contention points are inter-cluster channels and hotspot destinations, which are oversubscribed by different ratios; hotspot destinations are oversubscribed by 18:1, while inter-cluster channels by 4.5:1. This creates a challenge for ECN since the optimal configuration also depends on the over-subscription ratio and the num-

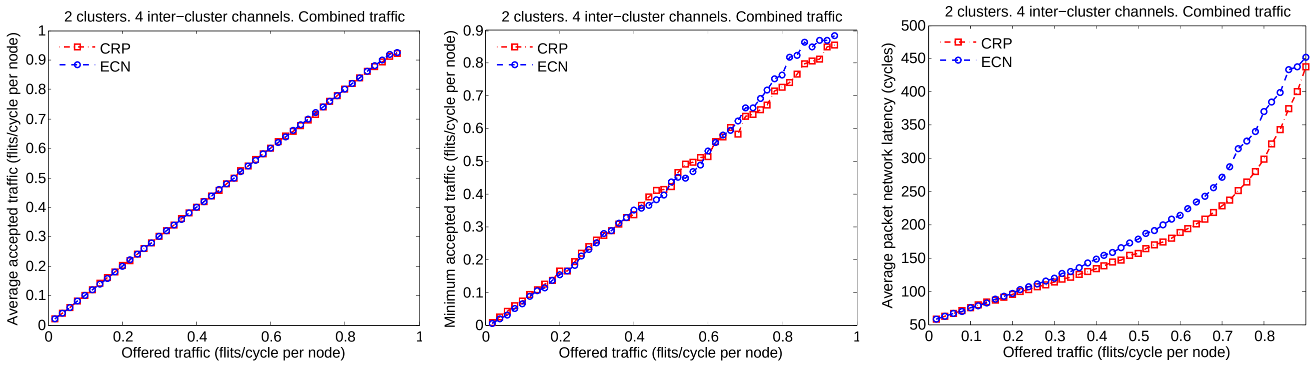


Figure 8: Foreground intra-cluster UR traffic performance with our combined traffic pattern.

ber of senders to a congestion point. In our simulations, we chose to configure ECN to perform optimally for the inter-cluster channels, which makes ECN too aggressive towards the hotspot destinations because it is not throttling down sources for long enough after each congestion notification. This contention does not penalize throughput in our experiments because the buffers are large enough to absorb the contention, but it does penalize latency. If we had configured ECN to perform optimally for hotspot destinations, throughput would had been penalized instead because ECN would had been too conservative towards inter-cluster channels. This illustrates a key limitation of ECN.

4.3 Transient Traffic

While ECN can be effective at handling congestion under steady-state traffic, realistic traffic in system-wide networks is often dynamic and unpredictable. Therefore, speedy adaptation to changes in the traffic pattern is critical. To illustrate the inherent *reactive* property of ECN, we conduct experiments using a step-function traffic pattern. This pattern first warms up the network for 50,000 cycles with 40% of intra-cluster UR traffic at each source and then switches to our combined traffic pattern. Throughput and latency for the intra-cluster UR foreground traffic from shortly before the onset of the hotspot until ECN's performance stabilizes are illustrated in Figure 9. The baseline network is not shown because it never recovers.

ECN's throughput and latency are adversarially affected for 300,000 cycles. ECN is unaware of the hotspot from the moment the hotspot is activated until sources are throttled down because of congestion notifications. This causes a number of packets to be injected that cannot be ejected at the pre-hotspot rate, due to tree saturation from the hotspot traffic. Those packets are destined to further increase congestion in the network. After 300,000 cycles, those packets are drained and ECN's performance is restored to pre-hotspot levels. Near the end of the recovery period, the throughput curve spikes to compensate for the initial post-transition throughput deficiency. ECN's maximum average network latency during the transition is 37,000 cycles; that number becomes 120 cycles after ECN stabilizes. ECN's reaction time also depends on the over-subscription ratio of the hotspot because a node can only generate one congestion notification per packet. There is no way to encode the severity of the congestion. While ECN can be configured to detect congestion faster, doing so will negatively affect its performance in other traffic patterns, such as benign traffic where lower congestion detection thresholds will produce more false positives. This poor adaptation behavior to transient traffic extends to other reactive schemes as well, such as TCP.

In contrast, CRP is unaffected by the step-function adversarial traffic. This highlights the key property of CRP of preventing congestion from ever occurring, and therefore avoids even momentary interference with benign flows. While congestion may form in speculative VCs, that does not reduce performance because speculative packets never deprive granted packets of bandwidth. Furthermore, although temporary congestion may form in control VCs, that congestion quickly dissipates since the traffic pattern necessary to form the congestion cannot be sustained [28].

4.4 Sensitivity to Message Size

Figure 10 illustrates results for our combined traffic pattern where each message's size is randomly chosen between 1 and 14 packets. This stresses CRP's handling of small flows. Compared to the results of Figure 8, ECN has a 2% lower average throughput, 4.5% lower minimum throughput, and comparable network latency under a 94% injection rate. However, CRP is more negatively affected with 3% lower average throughput, 9% lower minimum throughput and approximately twice the network latency under a 94% injection rate. This is because CRP's handling of flows too small to participate in CRP is based on the expectation that current traffic behavior is a good indicator for the future, as discussed in Section 2.5, which is less true with randomly-chosen message sizes. Furthermore, CRP does not throttle small flows and therefore does not prevent congestion formed solely by small flows, which occurs under high loads in our experiment. If numerous small flows are expected, reducing N_{min} may be beneficial. However, this will increase CRP's overhead relative to the data payload due to control messages. Alternatively, many small flows to the same destination, or perhaps simply traversing a common set of inter-cluster channels, can be grouped into one CRP request larger than N_{min}. ECN is unaffected by message size since flows are throttled regardless of message size.

Even though CRP's performance degrades in this case chosen specifically as adversarial to CRP, in experiments with better-behaved traffic where small flows have a fixed size and do not cause congestion by themselves, CRP's performance is hardly affected because short packets are recorded in the reservation tables as discussed in Section 2.5. In contrast, SRP completely disregards short packets [28].

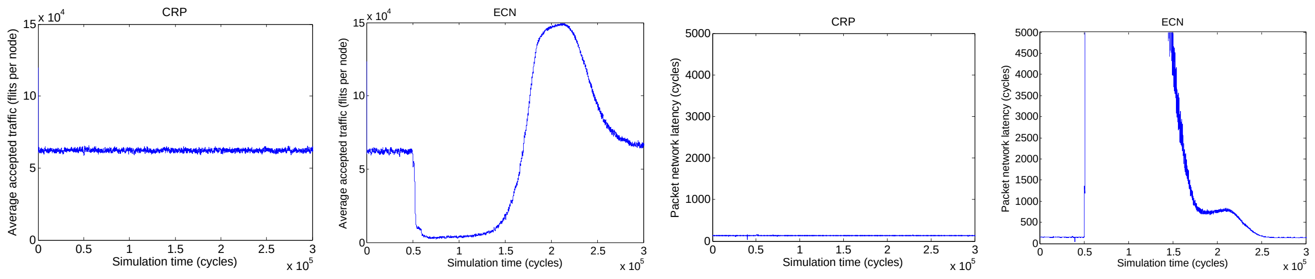


Figure 9: ECN is adversarially affected and recovers 300,000 cycles after the onset of a hotspot. CRP is unaffected.

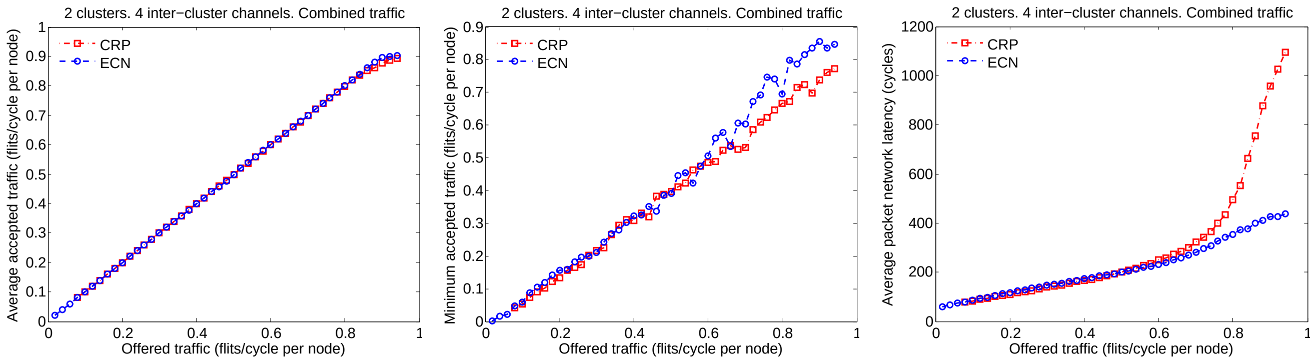


Figure 10: Foreground intra-cluster UR traffic performance with our combined pattern and randomly-chosen message sizes of 1 to 14 packets.

4.5 Sensitivity to the Network Configuration

To test CRP's sensitivity to the network configuration, we experiment with two network clusters interconnected by one or two channels instead of the default four. CRP's performance is unaffected compared to Figure 8. ECN suffers a marginal 2% reduction in minimum throughput for two inter-cluster channels, and 1% for one inter-cluster channel.

We also conduct experiments for three and four clusters interconnected in a bidirectional ring fashion, using our combined traffic pattern. Flows choose the shortest route to their destination cluster. Therefore, with three clusters flows still traverse one inter-cluster channel whereas with four clusters flows traverse one or two inter-cluster channels. As Figure 11 shows, ECN needs to be re-configured for each of the two cases, especially for the four network clusters where it offers 38% lower throughput, whereas CRP's performance is unaffected and is comparable to that of Figure 8.

Finally, we modify the hotspot background traffic to have all sources transmit to the hotspots, instead of only half the sources per cluster. With ECN, the foreground traffic achieves only 62% average accepted rate and 53% minimum under 95% flit injection rate per node. This shows that ECN cannot achieve optimal throughput in this different traffic pattern without new configuration parameters. In contrast, CRP's performance is comparable to Figure 8.

4.6 Sensitivity to Configuration Parameters

With our default configuration and the combined traffic pattern, using a larger C_{max} of 534 decreases throughput by 5% due to the increased inaccuracy in generated timestamps, as discussed in Section 2.5. A drop in throughput is also realized by setting C_{max} to a lower value than 269, because a single request may no longer fit into a single cell, which exacerbates the fragmentation problem. Furthermore, CRP's performance is comparable for R_{cycles} set to 0, 30 and 60 cycles for our combined traffic pattern. The same is true for R_{res} set to 0, 1 and 2, except that excessively large values for this parameter cause underutilization of bandwidth. Finally, the expected message size affects C_{max}, N_{max} and N_{min}, as discussed in Section 2.5. However, if message sizes cannot be predicted, CRP can segment or merge messages to the same destination such that the amount of packets per reservation remains reasonable. Except for the expected message size, CRP's configuration parameters are not affected by the traffic pattern and network configuration.

5. DISCUSSION

Our experiments identify two limitations of CRP. Firstly, as the number of flows requesting a resource increases as well as the number of resources flows require, so does the control overhead from retry responses. To mitigate this, destinations can choose an available time slot at random instead of the earliest one. While this has the potential to increase reservation time even without congestion, speculative packets mitigate latency under low loads. The number of retries can also be reduced by adjusting R_{res} and R_{cycles}.

The second limitation of CRP is the fact that different flows may need to reserve different numbers of resources. Flows that do not traverse inter-cluster channels will not receive a retry response unless their destination's reservation table is full. However, flows that do traverse inter-cluster channels may receive a retry response because a common availability was not found. This can cause unfair bandwidth allocation based on the number of participating resources flows traverse. To mitigate this, we can vary the number of time slots a request can eagerly reserve (R_{res}) depending on

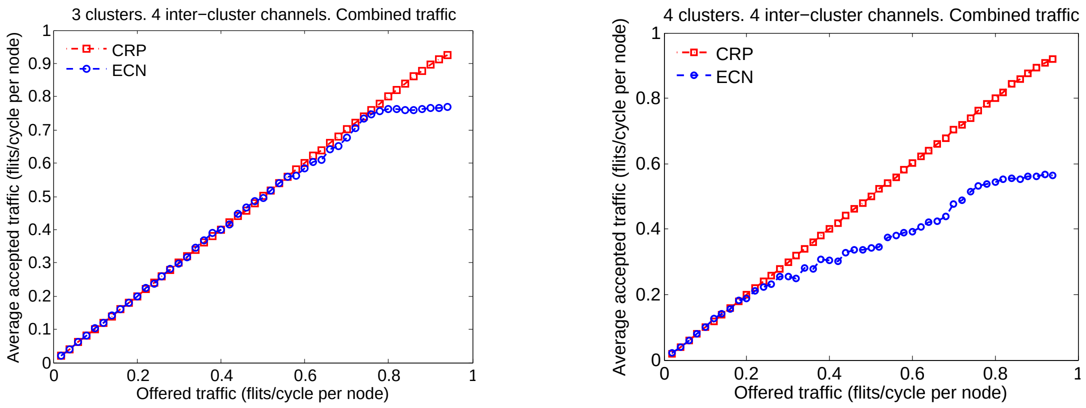


Figure 11: ECN needs to be re-optimized for three or four network clusters. CRP's performance is unaffected.

the number of resources it needs to allocate. This problem is caused by the lack of global knowledge when making allocation decisions locally. It can also appear in ECN since flows may traverse a different number of congestion points. To prevent this issue and the bandwidth overhead for retries from affecting CRP's scalability, fairness policies may be necessary, as discussed in Section 2.6.

Implementing CRP requires the simple modifications to routers and NICs that are required for SRP [28] for speculative and control packets, with the addition of retry responses and reservation tables. Similarly to SRP, NIC injection buffers should be large enough to allow packets to remain stored for a duration at least equal to the round-trip delay without stalling the source even in the presence of bursts. The round-trip delay is the minimum time packets have to be stored for until an ACK can arrive. In addition to SRP, CRP requires reservation tables at NICs, as well as tables at any router output transmitting to a channel participating in CRP. These tables and the corresponding handling logic are a minor cost increase for the large-radix and complex system-wide network routers. For example, a reservation table with C_{max} set to 269 and V_{cells} set to 32 requires 288 register cells, less than a 0.1% increase in the number of register cells of a YARC router [41]. This cost will increase for larger-scale networks with more over-subscribed resources that need larger C_{max} or V_{cells}, but still remains a small fraction of router cost.

Furthermore, CRP incurs a negligible latency overhead due to the speculative packets. Moreover, the bandwidth overhead of control packets other than retries is a function of the message size, N_{max} and N_{min}, but remains low because control packets are much smaller than data packets. Speculative packets do consume bandwidth but have the lowest priority and are dropped under congestion such that they never deprive other packets of bandwidth.

Even though CRP creates reservations, it is simply a statistical scheme to prevent resource over-subscription. Therefore, in case of improper synchronization, a clock drift, or transient error that causes a flow's granted timestamp to be altered, the overall bandwidth demand on resources will still not exceed their capabilities. In addition, a retry response erroneously converted to a grant due to a transient error will only cause one flow more than what resources can handle to be transmitted; this is easily absorbed by buffers. Finally, routers can detect malicious requests from sources in terms of size or number of requests, and impose quotas per source.

An alternative method for tackling the problem of multiple network clusters interconnected with under-provisioned channels is to have packets use SRP [28] to each participating resource in sequence. While this method simplifies the reservation logic and eliminates retry responses, each resource requires excessive buffering because it needs to hold all packets waiting to be granted by the next resource.

In addition, CRP can be used in networks without over-subscribed inter-cluster channels. In this case, CRP behaves like SRP [28]. CRP can also be applied to dropping flow control in order to reduce the probability of dropping packets. Finally, CRP applies to heterogeneous networks after adjusting the reservation tables in each part of the network to match available bandwidth.

6. RELATED WORK

ECN has been widely implemented in Infiniband-style networks [39, 25], such as the InfiniScale IV router [21]. Several studies have noted its good performance characteristics [37, 21, 45]. Further studies have proposed improvements to ECN by improving when routers detect congestion and how traffic sources react [39, 15, 45]. However, numerous studies have also pointed out ECN's shortcomings. The study of [37] points out that ECN provides good congestion control for long-lived flows, but risks causing network instability if its configuration parameters are not matched to the traffic pattern. This can be a challenging task given ECN's sensitivity to its configuration parameters as well the unpredictability of the traffic pattern. Finally, all ECN methods have been reported to have an adversary effect on latency at the onset of a hotspot at the edge of or inside the network [15, 28].

Many alternatives to ECN have been proposed [8]. Circuit switching also creates end-to-end reservations which include channels [46]. However, circuit switching does not allow other flows to use a reserved resource—even if no data is flowing in it—from the time a circuit is established until teardown. This imposes many idle cycles because of the latency from setting up a circuit until data transmission, which can seriously degrade network throughput [29]. In contrast, CRP creates fine-grain future reservations and thus time slots adjacent to a reservation are available to other flows. CRP is simply a statistical scheme to allocate bandwidth; it does not enforce strict reservation because flits may arrive at different times than their reservations. In that case, bandwidth is used by speculative and control packets.

SRP [28] and alternative end-to-end congestion prevention techniques [10] prevent congestion due to hotspot destinations but do not detect or resolve in-network congestion. Such techniques are ineffective for the testcases we evaluate.

In other congestion control techniques, networks prioritize packets causing congestion to speed up resolution [42]. This is based on the expectation that congestion is transient. EyeQ moves congestion and QoS handling from the network to the endpoints by speeding up the network fabric compared to the edge links [26]. Furthermore, global network knowledge can be used—when feasible—to perform admission control [43]. Other techniques do not prevent congestion but instead try to reduce its adversary effects by placing congested packets in different lanes or VCs [22, 14, 19]. For instance, RECN places congested packets in separate queues to avoid interference with benign traffic [18]. However, RECN requires large and costly CAMs, and is limited in the number of saturation trees it can handle by the size of the queues. Also, with flit-reservation in on-chip networks, packets make reservations at each hop and only for the next router, instead of end-to-end like CRP [35]. Therefore, tree saturation can still form.

In the past, lossy networks have used random early detection (RED) to probabilistically drop packets even without a full buffer, in order to indirectly signal congestion to TCP flows [16]. More recently, TCP in datacenter networks has been modified to reduce packet dropping due to oversubscription [6], such as implementing ECN with TCP [38] or modifying TCP [5]. Even though tree saturation does not form in lossy networks, packet drops and congestion control directly affect throughput. CRP is applicable to dropping networks to reduce the dropping probability and offer advantages such as with transient traffic. Reducing packet drops reduces the bandwidth demand for retransmitting packets.

Adaptive routing and flow scheduling techniques aim to distribute load evenly [20, 4], but do not protect from tree saturation when traffic exceeds available bandwidth.

7. CONCLUSION

This paper proposes CRP, a reservation protocol that *prevents* congestion at both over-subscribed channels and destinations by reserving multiple resources with a single request, and without idling resources from reservation until data traversal like circuit switching. Reservation requests record the available time slots for participating resources en route to their destination. Destinations then generate a grant with a timestamp for which all required resources are available. By reserving resources in advance of packet transmission, congestion is prevented from ever occurring. In contrast, *reactive* methods such as ECN allow tree saturation to form before acting. As we show, CRP reacts instantly to the onset of a hotspot. In contrast, ECN requires 300,000 cycles to stabilize in our experiments. Furthermore, ECN may not react to congestion in network channels formed by short-lived flows generated by a large combination of source–destination pairs. CRP is also reasonably insensitive to the network configuration and traffic pattern, whereas ECN's optimal configuration is sensitive to numerous and often unpredictable factors, making ECN a challenge to configure. However, CRP has a higher control bandwidth overhead than ECN. CRP is an effective technique to make lossless flow control more attractive by eliminating tree saturation.

Acknowledgments

This work was supported in part by the National Science Foundation under Grant CCF-0702341, in part by the National Security Agency under Contract H98230-08-C-0272-P007 and in part by the Robert Bosch, Prof. Michael Farmwald and Prof. Michael J. Flynn Stanford Graduate Fellowships.

8. REFERENCES

[1] Mellanox infiniscale IV switch architecture provides massively scalable 40GB/s server and storage connectivity, 2007. Press release published online.

[2] Dennis Abts, Michael R. Marty, Philip M. Wells, Peter Klausler, and Hong Liu. Energy proportional datacenter networks. *SIGARCH Computer Architecture News*, pages 338–347, 2010.

[3] Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. A scalable, commodity data center network architecture. In *Proceedings of the ACM SIGCOMM 2008 conference on Data communication*, pages 63–74, 2008.

[4] Mohammad Al-Fares, Sivasankar Radhakrishnan, Barath Raghavan, Nelson Huang, and Amin Vahdat. Hedera: dynamic flow scheduling for data center networks. In *NSDI '10: Proceedings of the 7th USENIX conference on Networked systems design and implementation*, 2010.

[5] Mohammad Alizadeh, Albert Greenberg, David A. Maltz, Jitendra Padhye, Parveen Patel, Balaji Prabhakar, Sudipta Sengupta, and Murari Sridharan. Data center TCP (DCTCP). In *Proceedings of the ACM SIGCOMM 2010 conference*, pages 63–74, 2010.

[6] Andreea S. Anghel, Robert Birke, Daniel Crisan, and Mitchell Gusat. Cross-layer flow and congestion control for datacenter networks. In *DC-CaVES ' 11: Proceedings of the 3rd Workshop on Data Center - Converged and Virtual Ethernet Switching*, pages 44–62, 2011.

[7] Berk Atikoglu, Yuehai Xu, Eitan Frachtenberg, Song Jiang, and Mike Paleczny. Workload analysis of a large-scale key-value store. In *SIGMETRICS '12: Proceedings of the 12th ACM SIGMETRICS/PERFORMANCE joint international conference on Measurement and Modeling of Computer Systems*, pages 53–64, 2012.

[8] E. Baydal, P. Lopez, and J. Duato. A family of mechanisms for congestion control in wormhole networks. *IEEE Transactions on Parallel and Distributed Systems*, 16(9):772 – 784, 2005.

[9] S. Chakravarthi, A. Pillai, J.P. Neelamegam, M. Apte, and A. Skjellum. A fine-grain clock synchronization mechanism for myrinet clusters. In *LCN '02: Proceedings of the 27th Annual IEEE Conference on Local Computer Networks*, pages 708 – 715, 2002.

[10] Nikos Chrysos, Lydia Y. Chen, Cyriel Minkenberg, Christoforos Kachris, and Manolis Katevenis. End-to-end congestion management for non-blocking multi-stage switching fabrics. In *Proceedings of the 6th ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, 2010.

[11] William J. Dally. Virtual-channel flow control. *IEEE Transactions on Parallel and Distributed Systems*, 3(2):194–205, 1992.

[12] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[13] Christina Delimitrou, Sriram Sankar, Aman Kansal, and Christos Kozyrakis. ECHO: Recreating Network Traffic Maps for Datacenetrs with Tens of Thousands of Servers. In *IISWC '12: Proceedings of the 2012 IEEE International Symposium on Workload Characterization*, 2012.

[14] 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 *HPCA '05: Proceedings of the 11th*

International Symposium on High-Performance Computer Architecture, pages 108–119, 2005.

[15] Joan-LLuis Ferrer, Elvira Baydal, Antonio Robles, Pedro Lopez, and Jose Duato. Congestion management in mins through marked and validated packets. In *PDP '07: Proceedings of the 15th Euromicro International Conference on Parallel, Distributed and Network-Based Processing*, pages 254–261, 2007.

[16] Sally Floyd and Van Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, 1993.

[17] S. Ben Fred, T. Bonald, A. Proutiere, G. Régnié, and J. W. Roberts. Statistical bandwidth sharing: a study of congestion at flow level. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 111–122, 2001.

[18] P. J. Garcia, J. Flich, J. Duato, I. Johnson, F. J. Quiles, and F. Naven. Dynamic evolution of congestion trees: Analysis and impact on switch architecture. In *HiPEAC'05: Proceedings of the 2005 International Conference on High Performance Embedded Architectures and Compilers*, pages 266–285, 2005.

[19] P.J. Garcia, F.J. Quiles, J. Flich, J. Duato, I. Johnson, and F. Naven. Efficient, scalable congestion management for interconnection networks. *Micro, IEEE*, 26(5):52 –66, 2006.

[20] Patrick Geoffray and Torsten Hoefler. Adaptive routing strategies for modern high performance networks. In *HOTI '08: Proceedings of the 2008 16th IEEE Symposium on High Performance Interconnects*, 2008.

[21] E.G. Gran, M. Eimot, S.-A. Reinemo, T. Skeie, O. Lysne, L.P. Huse, and G. Shainer. First experiences with congestion control in infiniband hardware. In *IPDPS '10: IEEE International Symposium on Parallel Distributed Processing*, pages 1–12, 2010.

[22] Wei Lin Guay, Sven-Arne Reinemo, Olav Lysne, and Tor Skeie. dftree: a fat-tree routing algorithm using dynamic allocation of virtual lanes to alleviate congestion in infiniband networks. In *NDM '11: Proceedings of the first international workshop on Network-aware data management*, pages 1–10, 2011.

[23] Mitchell Gusat, Robert Birke, and Cyriel Minkenberg. Delay-based cloud congestion control. In *GLOBECOM '09: Proceedings of the 28th IEEE conference on Global telecommunications*, pages 5691–5698, 2009.

[24] Mitchell Gusat, Cyriel Minkenberg, and Gergely János Paljak. Flow and congestion control for datacenter networks. (RZ3742), 2009 2009.

[25] Tsuyoshi Hamada and Naohito Nakasato. Infiniband trade association, infiniband architecture specification, volume 1, release 1.0, http://www.infinibandta.com. In *in International Conference on Field Programmable Logic and Applications, 2005*, pages 366–373, 2005.

[26] Vimalkumar Jeyakumar and Changhoon Kim. Eyeq: Practical network performance isolation for the multi-tenant cloud. In *4th USENIX Workshop on Hot Topics in Cloud Computing*, 2012.

[27] Nan Jiang, Daniel U. Becker, George Michelogiannakis, James Balfour, Brian Towles, John Kim, and William J. Dally. A detailed and flexible cycle-accurate network-on-chip simulator. In *ISPASS '13: IEEE International Symposium on Performance Analysis of Systems and Software*, pages 86–96, 2013.

[28] Nan Jiang, Daniel U. Becker, George Michelogiannakis, and William J. Dally. Network congestion avoidance through speculative reservation. In *HPCA '12: Proceeding of the 18th International Symposium on High-Performance*, pages 1–12, 2012.

[29] P. Kermani and L. Kleinrock. A tradeoff study of switching systems in computer communication networks. *IEEE Transactions on Computers*, C-29(12):1052 –1060, 1980.

[30] C. P. Kruskal and M. Snir. The performance of multistage interconnection networks for multiprocessors. *IEEE Transanctions on Computers*, pages 1091–1098, 1983.

[31] Hong Liu, C.F. Lam, and C. Johnson. Scaling optical interconnects in datacenter networks opportunities and challenges for wdm. In *HOTI '10: 2010 IEEE 18th Annual Symposium on High Performance Interconnects*, pages 113 –116, 2010.

[32] Ajay Mahimkar, Angela Chiu, Robert Doverspike, Mark D. Feuer, Peter Magill, Emmanuil Mavrogiorgis, Jorge Pastor, Sheryl L. Woodward, and Jennifer Yates. Bandwidth on demand for inter-data center communication. In *Proceedings of the 10th ACM Workshop on Hot Topics in Networks*, HotNets-X, 2011.

[33] David Meisner, Christopher M. Sadler, Luiz André Barroso, Wolf-Dietrich Weber, and Thomas F. Wenisch. Power management of online data-intensive services. *SIGARCH Computer Architure News*, 39(3):319–330, 2011.

[34] T. Nachiondo, J. Flich, and J. Duato. Buffer management strategies to reduce hol blocking. *IEEE Transactions on Parallel and Distributed Systems*, 21(6):739 –753, 2010.

[35] Li-Shiuan Peh and W. J. Dally. Flit-reservation flow control. In *HPCA '00: Proceeding of the 6th International Symposium on High-Performance Computer Architecture*, pages 73–84, 2000.

[36] F. Petrini and M. Vanneschi. k-ary n-trees: high performance networks for massively parallel architectures. In *IPPS '97: Proceedings of the 11th International Parallel Processing Symposium*, pages 87 –93, 1997.

[37] 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. In *High Performance Interconnects for Distributed Computing*, 2005.

[38] K. Ramakrishnan, S. Floyd, and D. Black. RFC 3168: The addition of explicit congestion notification (ECN) to IP, 2001.

[39] J.R. Santos, Y. Turner, and G. Janakiraman. End-to-end congestion control for infiniband. In *INFOCOM '03. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications*, volume 2, pages 1123 – 1133, 2003.

[40] S. L. Scott and G. S. Sohi. Using feedback to control tree saturation in multistage interconnection networks. In *ISCA '89: Proceedings of the 16th annual international symposium on Computer architecture*, pages 167–176, 1989.

[41] Steve Scott, Dennis Abts, John Kim, and William J. Dally. The blackwidow high-radix clos network. In *ISCA '06: Proceedings of the 33rd annual international symposium on Computer Architecture*, pages 16–28, 2006.

[42] Yong Ho Song and Timothy Mark Pinkston. Distributed resolution of network congestion and potential deadlock using reservation-based scheduling. *IEEE Transactions on Parallel and Distributed Systems*, 16(8):686–701, 2005.

[43] Mithuna Thottethodi, Alvin R. Lebeck, and Shubhendu S. Mukherjee. Self-tuned congestion control for multiprocessor networks. In *HPCA '01: Proceedings of the 7th International Symposium on High-Performance Computer Architecture*, 2001.

[44] A. Vahdat, M. Al-Fares, N. Farrington, R.N. Mysore, G. Porter, and S. Radhakrishnan. Scale-out networking in the data center. *Micro, IEEE*, 30(4), 2010.

[45] Shiyang Yan, Geyong Min, and M. Awan. An enhanced congestion control mechanism in infiniband networks for high performance computing systems. In *AINA '06: 20th International Conference on Advanced Information Networking and Applications*, pages 845 –850, 2006.

[46] Cui-Qing Yang and A.V.S. Reddy. A taxonomy for congestion control algorithms in packet switching networks. *IEEE Network*, 9(4):34 –45, 1995.