Adaptive Routing for Convergence Enhanced Ethernet

Cyriel Minkenberg, Alessandra Scicchitano, Mitchell Gusat
IBM Research, Zurich Research Laboratory
Säumerstrasse 4, CH-8803 Rüschlikon, Switzerland
{sil,als,mig}@zurich.ibm.com

Abstract— **A significant drive to consolidate data center networks on a single infrastructure is taking place. 10-Gigabit Ethernet is one of the contenders to fulfill the role of universal data center interconnect. One of the key features missing from conventional Ethernet is congestion management; this void is being filled by the standardization work of the IEEE 802.1Qau working group. However, the schemes under consideration react to congestion only at the sources by reducing the transmission rates of "hot" flows, i.e., those detected as contributing to congestion. This approach ignores a crucial aspect of many data center networks, namely, that there typically are multiple paths between any pair of end nodes. Before reducing transmission rates, it would make sense to look for an alternative, uncongested path first. Here, we propose an adaptive routing scheme that builds—in a fully transparent way—on top of the existing 802.1Qau schemes, by snooping the congestion notification frames to modify the routing behavior of the switching nodes. We demonstrate how this can lead to significant performance improvements by taking full advantage of path diversity.**

I. Introduction

The IEEE 802.3 standard defines Ethernet, one of the most widely implemented local area networks (LAN). Simplicity, scalability, wide availability, and low cost, combined with "good enough" performance have made Ethernet the network of choice for LAN traffic. Its performance parameters such as throughput, drop rate, latency, and jitter are considered acceptable for LAN traffic. While consistent with the keep-it-simple, low-cost philosophy of Ethernet, this also led to the development of specialized networks for applications—storage area networks (SAN), clustering, and high-performance computing (HPC) in particular—with more stringent requirements. Examples of such networks are Fibre Channel, InfiniBand, PCIe Advanced Switching, Myrinet, and QsNet.

As a result, current data center installations typically have a communication infrastructure comprising (at least) three disjoint networks: a LAN, a SAN, and a clustering network. To reduce cost, complexity, power consumption, as well as management and maintenance overhead, it would be highly desirable to create a unified data center network that carries all traffic over a single physical infrastructure.

To ensure that 10-Gigabit Ethernet (10GE) meets data center requirements, several working groups within the IEEE and IETF standards bodies are addressing key issues, including congestion management (IEEE 802.1Qau), traffic differentiation (IEEE 802.1Qbb), enhanced transmission selection (802.1Qaz), and multi-pathing (IETF TRILL). Such an enhanced version of Ethernet is sometimes referred to as Convergence Enhanced Ethernet (CEE).

The most important requirements of data center networks are low latency, losslessness, and high speed. Although losing packets might be tolerable in LANs or WANs, this is no longer true in a data center environment, where packet loss can seriously degrade system performance. To achieve the objective of losslessness and avoid drops due to buffer overflows, all data center networks employ some form of link-level flow control (LL-FC), usually some variation of either credit-based or stop/go-based flow control.

The combination of short buffers, for low latency, and high speed may easily lead to congested switch buffers, which will trigger the LL-FC mechanism, thus propagating the congestion to upstream switches. If congestion persists long enough, a *saturation tree* [1] of congested switches is induced, which can cause a catastrophic collapse of *global* network throughput [1], [2], as a saturation tree affects not only flows directly contributing to the congestion, but also other flows getting caught in the ensuing backlog. Therefore, congestion management (CM) is an essential safeguard against such collapses.

Although the IEEE 802.3 standard provides an LL-FC mechanism called PAUSE (802.3x), which can temporarily pause the link when the buffer is filling up, it does not provide CM at the datalink layer. The IEEE 802.1Qau working group is currently in the process of defining a standard for CM in 10GE networks. All CM protocols proposed in this context, such as ECM or QCN, try to eliminate congestion by reducing the sending rates at the sources. In a nutshell (see also Sec. II-A), these schemes operate by monitoring switch queue-length offsets with respect to a predefined equilibrium threshold (Q_{eq}) and queue length changes, computing a feedback value indicating the level of congestion, sending congestion notification (CN) frames to the sources of "hot" flows when congestion is detected, and reducing per-flow transmission rates at the sources based on the feedback value. This mechanism keeps congestion under control by reducing the aggregate sending rate of all flows that traverse the bottleneck, thus pushing the backlog to the edge of the network.

In this work, we consider exploiting the *multi-path* capability that is often present in data center networks. Multi-path networks offer a *spatial* alternative to the exclusively used *temporal* reaction in the 802.1Qau schemes.

We consider congestion as a change in the conditions of the path, similar to a broken link, for instance, which may be addressed by searching for another path between source and destination and, if one exists, by rerouting some or all of the traffic onto the new path. When there are multiple paths between source and destination, congestion may be solved by assigning different paths to some or all of the flows causing congestion. This would result in a higher overall network throughput than in the existing 802.1Qau approaches, which can only reduce the source rates. If a congested flow can be routed on an alternative, uncongested path, its rate does not need to be reduced. Therefore, combining adaptive routing (AR) with CM can significantly increase the throughput of a congested network. Transmission rates have to be reduced only when no alternative uncongested path exists.

To enable multi-path routing, we configure the switch routing tables to allow multiple routing table entries for every destination MAC address. Our AR scheme is built on top of the existing end-to-end CM schemes being defined in 802.1Qau and takes advantage of the congestion notifications generated to trigger rerouting. A key advantage of the proposed scheme is that no changes are necessary to the Ethernet frame format, existing CM schemes, and Ethernet adapters. The scheme operates by exclusively modifying the routing behavior of the Ethernet switching nodes.

The paper is organized as follows. Sec. II reviews the IEEE 802.1Qau CM schemes and related work in the area of AR. In Sec. III, we first discuss how to setup the switches' routing tables to enable multi-path routing while avoiding routing loops, and then proceed to describe our proposed AR scheme for 10GE data center networks in Sec. IV. We present the evaluation methodology in Sec. V, including network topologies and traffic scenarios. Sec. VI demonstrates, by simulating the proposed scheme on several test topologies, that significant performance gains in terms of throughput and latency can be achieved. We conclude in Sec. VII.

II. Related work

Here we provide a brief overview of relevant work on Ethernet CM and related AR schemes.

A. Ethernet congestion management

This section provides an overview of the efforts of the IEEE 802.1Qau Task Group towards defining a CM scheme for 10GE. Several mechanisms have been considered so far. Their primary goal is to stabilize the switch buffer length around the Q_{eq} threshold so that the buffer is neither over- nor underutilized.

The 802.1Qau group has adopted an end-to-end approach to CM aiming to push congestion out to the edge of the network. The framework is based on controlling the switch queue lengths by generating CN frames that cause the senders to impose and adjust rate limits for flows contributing to congestion. The key components are the following:

1) *Congestion detection and signaling*: Each switch samples the incoming frames with a (randomized) sample interval I_s. When a frame is sampled, the switch determines the output buffer length and computes a feedback value F_b, which is a weighted sum of the current queue offset Q_{off} with respect to the equilibrium threshold Q_{eq} and the level change Q_{delta} since the preceding sample: $F_b = Q_{\text{off}} - W_d * Q_{\text{delta}}$. Depending on the value of F_b, the switch sends a CN to the source of the sampled frame. The congested queue that causes the generation of the message is called *congestion point* (CP).
2) *Source reaction*: Each source has an associated *reaction point* (RP) that instantiates rate limiters for congested flows, and adjusts the sending rates depending on the feedback received. When an RP receives a CN, it decreases the rate limit of the sampled flow if the feedback is negative, and increases it if the feedback is positive. To avoid starvation, accelerate rate recovery when congestion is over, and improve fairness of rate allocation, the RP can autonomously increase rate limits based on timers or byte counting.

Ethernet Congestion Management (ECM; formerly called BCN) [3], [4] signals both positive and negative feedback. Negative feedback is generated only if the buffer level exceeds Q_{eq}. Positive feedback is generated only if the sampled frame is tagged as belonging to a rate-limited flow *and* the tag contains the congestion point ID (CPID) that corresponds to the switch and output queue in question. Here, CPID tagging is needed to be able to filter out false positives; otherwise it might happen that a rate limit is increased by positive feedback from uncongested output queues on the same path, thus compromising stability. Each rate limiter is associated with the CPID of the most recent negative feedback; each rate-limited frame is tagged with the associated CPID.

Another scheme under consideration is Quantized Congestion Notification (QCN), first proposed in [5] and detailed in [6]. The main feature of this algorithm is the total absence of positive feedback. A source autonomously increases the rate of a rate-limited flow after a time interval T during which it has not received any negative feedback. This T is determined by counting the number of bytes transmitted for a given rate-limited flow and comparing it with a threshold that depends on the last feedback value.

Both schemes, as shown in [8], are able to provide good performance in controlling the queue length at the CP. However, the performance benchmarks [9] used in 802.1Qau consider single-path topologies exclusively. Here, we study congestion in multi-path topologies, where congestion may be solved by routing around it rather than by throttling the sources. In such topologies, relying on source-rate adjustments alone may lead to severe underutilization of network resources.

Further details on ECM and QCN can be found in [3]–[7]. As QCN is most likely to become the standardized scheme, we based our AR proposal on QCN.

A discussion on L2 versus higher-level CM, including TCP, is outside the scope of this paper. However, important reasons for L2 CM are (a) that many data center applications do not use TCP, and (b) that several aspects of TCP are not well suited to data centers (e.g., slow start and congestion detection based on duplicate ACKs, i.e., lost packets).

B. *Adaptive routing*

In packet switching networks the goal of routing protocols is to select the path that a message should take to reach its destination. The choice may be made from a set of different paths and based on different decision metrics. Existing routing algorithms can be divided into two categories: *Deterministic* and *adaptive* routing.

Deterministic routing became popular when wormhole switching was invented [10]. Its popularity is due to its minimal hardware requirement. Indeed the use of its simple deadlock-avoidance algorithm results in a design of simple and fast routers. In deterministic routing algorithms, paths between sources and destinations are fixed, and messages with the same source and destination addresses always take the same route [10], [11]. Consequently, a message must wait for each busy channel in the path to become available. The main drawback of these algorithms is that they cannot take advantage of alternative paths to avoid blocked channels.

AR algorithms support multiple paths between each source-destination pair, allowing messages to explore all alternative routes when crossing the network [12], [13]. AR algorithms are either *minimal* or *non-minimal*. Minimal routing algorithms allow only shortest paths to be chosen, whereas non-minimal routing algorithms also allow longer paths. AR algorithms, whether minimal or non-minimal, can be further differentiated by the number of paths allowed. *Partially* adaptive routing algorithms do not allow all messages to use any path, while *fully* adaptive do not have any restriction on the set of paths that can be used.

Partially adaptive routings allow selecting an output channel from a subset of all possible channels. Planar adaptive routing algorithms [14] and turn-model-based algorithms [15] are the most important partially adaptive routing algorithms for the mesh, torus, and hypercube networks.

Many fully adaptive routing algorithms have been proposed so far. In [16] several algorithms based on buffer classes are presented. Deadlocks are avoided by using a buffer of a higher class every time a packet requests a new buffer. In [17], the message flow model was introduced, which proves that a routing algorithm is deadlock-free if no channel can be held forever by a message. Several fully adaptive routing algorithms on tori have been evaluated in [18] of which the one using Negative Hop-based (NHop) deadlock-free routing augmented with a new idea called bonus cards (Nbc) has been shown to have the best performance. In [18], the Nbc routing scheme has been used in the context of Duato's methodology [12], resulting in a routing algorithm named Duato-Nbc with high performance and minimal virtual channel requirements.

An architecture for data center networks based on commodity Ethernet switches was proposed in [19]; however, that approach uses L3 rather than L2 routing and does not support full adaptivity.

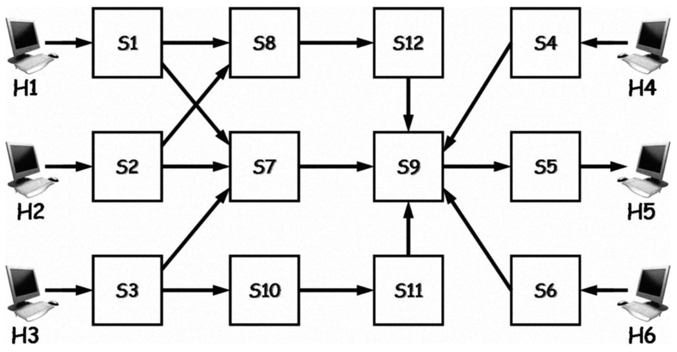


Fig. 1. Loop-free set of links to route to Host 5.

The main contribution of our work is a fully-adaptive non-minimal routing algorithm on top of 802.1Qau that helps reduce congestion by avoiding the CP as long as another uncongested path between source and destination is available, resorting to source rate reductions only if there is no such path. To the best of our knowledge, this is the first work on AR for congestion control in Ethernet networks.

III. Route Configuration

Traditionally, Ethernet networks employ the Spanning Tree Protocol (STP) or one of its variants to construct a routing tree, transforming an arbitrary physical topology (which may contain loops) into a logical one without loops by en- or disabling specific ports in each switch. This is necessary to eliminate routing loops in the topology, which lead to "broadcast storms," i.e., endless replications of broadcast frames. Unfortunately, this also means that potential multi-pathing capabilities are negated.

Note that any network with multiple paths from some source to some destination automatically has a loop in it. Hence, the above problem applies to all multi-path topologies and is therefore of particular interest for CEE, because data centers often employ multi-path network topologies.

Switched Ethernet networks typically do not require explicit programming of the switch routing tables. A switch automatically learns the routing by observing the source MAC addresses of the frames entering. If a frame from MAC m enters on port p, the switch makes an entry in its routing table to route frames destined to MAC m through port p. When a frame arrives for which no entry exists in the routing table, the switch broadcasts the frame to all ports except the one on which the frame arrived. In this way, each switch will discover a route to each end node (assuming that all end nodes generate some traffic that reaches all switches). As this method involves broadcasting, it may also cause broadcast storms on a multi-path topology.

To enable multi-path routing while preventing routing loops, we adopted an approach based on preconfiguring the routing tables of each switch in such a way that, for each destination node d, the directed graph formed by all routes leading to node d is free of loops, see Fig. 1. This guarantees that, regardless of which AR scheme is used, no frame can ever be routed in a loop, without having to keep track of previously visited switches. Route configuration is performed at network initialization.

Note that this also guarantees that the routing is deadlock-free, as there exists a routing subfunction that is free of cyclic channel dependencies [11, Sec. 3.1.3].

Each switch maintains a routing table that maps each destination MAC address to a list of available ports. The ports are listed in order of preference; the first entry is the default routing option. Ports having a shorter distance (hop count) to the destination receive higher preference.

IV. Adaptive Routing in CEE Networks

The basic idea of the proposed AR scheme is to take advantage of the congestion information conveyed by the notifications generated by 802.1Qau-enabled switches. These CNs travel backwards from the congestion point to the sources of the flows contributing to the hotspot. The upstream switches that relay CNs can *snoop* them to find out about downstream congestion. By marking ports as congested with respect to specific destinations, a switch can reorder its preferences of the corresponding output ports contained in the routing table entry for that destination. Clearly, uncongested ports will be preferred over congested ones.

A. Congestion notification snooping

To enable AR, each switch maintains a congestion information table that maps a congestion key (d, p), where d is the destination MAC address and p the local port number, to a small data structure that keeps track of the current congestion status of port p with respect to destination d. This data structure comprises the following four fields:

- A `congested` flag indicating whether congestion has been detected on port p for traffic destined to d.
- A `local` flag indicating (if `congested` is true) whether the congestion occurred locally, i.e., in the output queue attached to port p.
- A feedback counter `fbCount` indicating how many CNs have been snooped for (d, p).
- A feedback severity indication `feedback` providing an estimate of how severe the congestion is.

Whenever a switch receives or generates a congestion notification for a flow destined to d it updates the congestion information corresponding to (d, p), where p is the output port corresponding to the input port on which the CN was received (remote) or the output port that triggered the creation of the CN (local).

If the entry has not been marked as congested (or did not exist yet), the `congestion` flag is set and `local` is set according to whether the CN was generated remotely or locally, `fbCount` is incremented, and the product of `fbCount` and the feedback value carried by the CN is added to `feedback`. As CNs always carry negative feedback values, `feedback` will also be negative and decrease as more CNs are received. The lower the value of `feedback`, the more severe the congestion. We employ such a weighted update to assign more weight to recent CNs to gradually reduce the effect of older entries and false positives. In addition, this allows congestion points that generate small but frequent feedback values to accumulate a significantly negative `feedback` value to be considered congested. This is the case for a queue in equilibrium, i.e., one for which congestion is under control but load demand still exceeds link capacity.

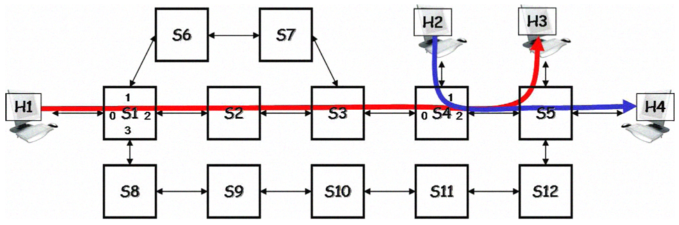


Fig. 2. Routing of congestion notifications.

There is one minor difference in the update procedure if the entry was already marked as congested: `local` is updated only if it already was true, i.e., local congestion can be overridden by remote congestion but not vice versa.

B. Congestion expiration

QCN only signals negative feedback, i.e., the presence or increase of congestion, but not the absence or decrease of congestion. As a result, we need a timer-based approach to expire remote entries in the congestion information table. Local entries can be expired when the corresponding output queue is no longer congested.

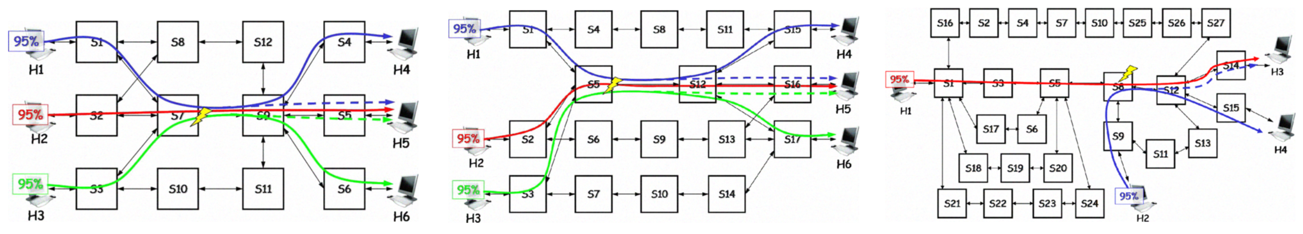
To this end, whenever an entry is updated as being congested, a timer is started. When the timer expires the entry is reset, provided that it is not flagged as local. A local entry is reset when the length of the corresponding output queue drops below $Q_{\text{eq}}/2$.

C. Routing decisions

When a frame arrives, a switch performs a routing lookup for the frame's destination MAC address d. If the default (most preferred) port p_0 is not flagged as congested by the congestion table entry for (d, p_0), the frame is routed to port p_0. If the default port *is* flagged as congested, the next-preferred port is checked, and so on. If all ports are flagged as congested, the frame will be routed to the port with the least severe congestion (i.e., with the feedback value closest to zero).

CN frames receive special treatment. They are not subjected to congestion checks. However, we need to ensure that all ports belonging to alternative paths leading to the congestion point are made aware of the congestion. If all CN frames are always routed on the same path to the reaction point (source), the flow might be rerouted on an alternative path that eventually ends up at the same congestion point. Figure 2 shows an example.

Both **H1** and **H2** are sending at line speed to **H3** and **H4**, respectively, causing severe congestion at port 2 of **S4** when the shortest paths are taken. The shortest reverse path back to **H1** is through switch **S2**. However, if all CNs for **H1** traverse **S2**, **S1** will only mark its port 2 as congested, but never port 1,



(a) Topology 1 (T1) (b) Topology 2 (T2) (c) Topology 3 (T3)

Fig. 3. Evaluated topologies and traffic scenarios. Solid lines correspond to noncontending-flow scenarios, dashed lines (where different) to contending-flow scenario.

so **S1** will route its traffic on the second-shortest path through port 1 to **S6** and **S7**, still ending up at the bottleneck in **S4**. Therefore, **S3** should ensure that it routes CNs also on the reverse path through **S7** and **S6**. Then, **S1** will mark ports 1 and 2 as congested with respect to destination **H3**, and will proceed to route its traffic through the longest path via **S8**–**S12** to **S5**, thus bypassing **S4** and eliminating the congestion.

We address this issue by having each switch *randomly* select one of the available ports when performing a lookup for a CN frame. As long as the congestion persists, the congested switch will keep generating CNs. By routing them randomly, each upstream switch should eventually be traversed.

V. Evaluation Methodology

We present the topologies and traffic scenarios used in our evaluation, explain how the routes are configured, and list the values of relevant simulation parameters.

A. Topologies and scenarios

Figure 3 shows the three different topologies used in our evaluation. The traffic patterns represented by solid arrows in Fig. 3 correspond to *noncontending* flows, as each flow has a different destination. However, because the preferred shortest paths all lead through a single bottleneck link, shortest-path routing will lead to severe congestion in these cases. Note that in each topology a set of routes is available that can eliminate congestion. In general, it is impossible to assign a set of routes that eliminates congestion for any combination of flows *a priori*.

To check whether the scheme marks *all* paths leading to the hotspot as congested, T3 comprises multiple such paths; only the longest path routes around the hotspot.

The traffic patterns represented by dashed arrows in Fig. 3 represent *contending* flows that, despite alternative routes being available, cannot be routed without contention. The purpose of this scenario is to determine whether with AR enabled the underlying CM scheme, i.e., QCN, still works as intended.

We also applied uniform Bernoulli traffic with a load varying from 10 to 100% to each topology. In this scenario, each host generates an equal share of traffic to each of the other hosts. We used this scenario to determine whether the proposed AR scheme can increase the saturation load of a given network topology. In all experiments, we measured the transmission rate per host and the queue length of each switch output queue, all averaged over 1 ms intervals.

Any fully adaptive routing scheme may cause frames to arrive out of order. Therefore, resequencing needs to implemented at the receiving end. We also measured the length of the resequencing buffer at each host.

B. Route configuration

Figure 1 shows an example of how the routes are programmed. It shows a possible route configuration for host **H5** of T1; bold arrows indicate the set of links (output ports) that may be used to route to **H5**. These links form a loop-free directed graph. Not all possible paths are enabled: for instance, **S8** could also route to **S2** to follow the path through **S7**, but this would create a loop. The paths for the other hosts and topologies are configured in a similar way. In general, this requires computing for each destination d a directed acyclic graph that connects all nodes $d' \neq d$ to node d.

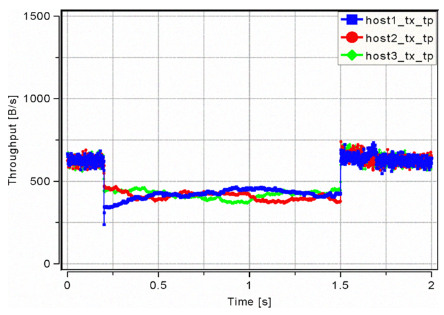
C. Simulation parameters

The CM algorithm used for detection and reaction is QCN as described in Sec. II-A. We implemented QCN version 2.2 as specified in [7]. The simulation parameters were set as shown in Table I.

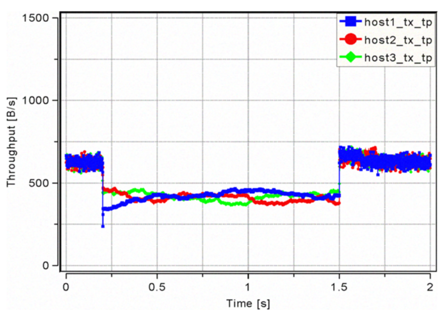
TABLE I
Simulation parameters

Parameter	Value	Unit
line rate	10	Gb/s
frame size	1500	B
buffer size per port	150	KB
Q_{eq}	33000	B
W_d	2	
G_d	1/128	
quantization	6	bit
timer	15	ms
extra fast recovery	enabled	
mean sample interval	150000	B
min. rate	100	Kb/s
fast recovery threshold	5	
byte count limit	150	KB
active increase	5	Mb/s
hyperactive increase	50	Mb/s
min. decrease factor	0.5	
AR timer	250	ms

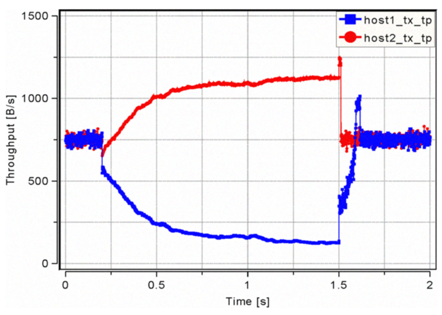
Each simulation run simulated two seconds' worth of real time. During the hotspot experiments, non-congesting random Bernoulli traffic is offered to the network during the initial



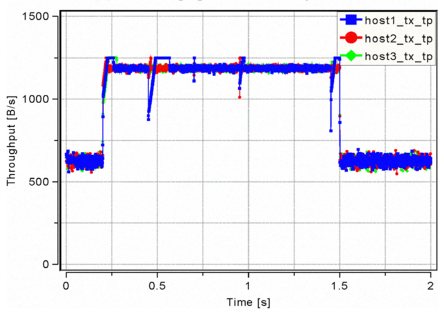
(a) Throughput, CM only, T1



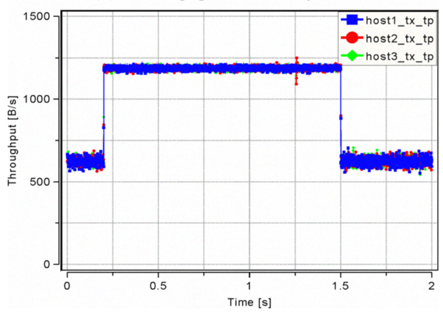
(b) Throughput, CM only, T2



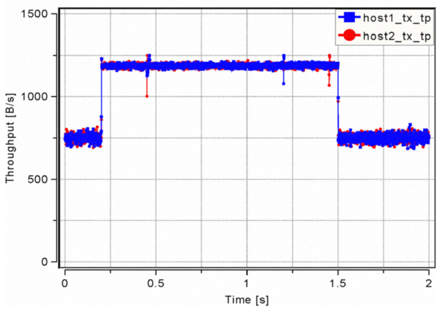
(c) Throughput, CM only, T3



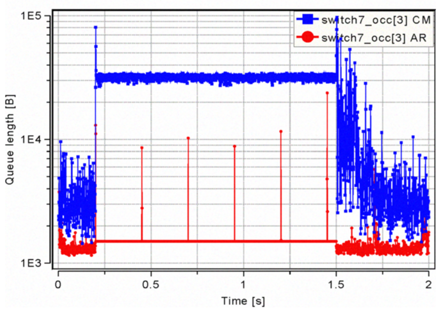
(d) Throughput, AR, T1



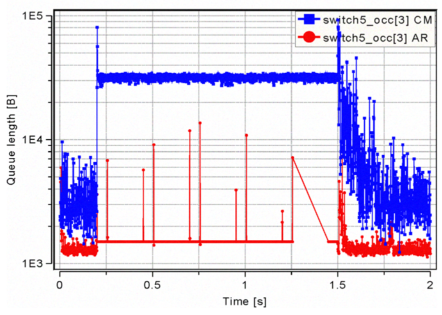
(e) Throughput, AR, T2



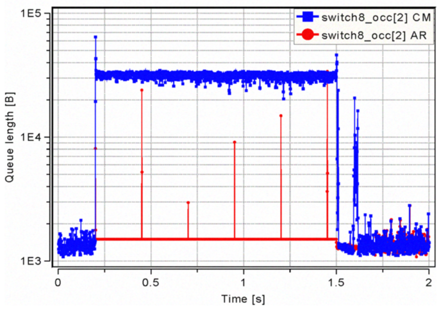
(f) Throughput, AR, T3



(g) Queue length, T1



(h) Queue length, T2



(i) Queue length, T3

Fig. 4. Results for noncontending-flow scenario.

200 ms and the final 500 ms of the simulation. The hotspot occurs from 200 to 1500 ms.

The switch architecture was output-queued but with buffers partitioned per input to support PAUSE. Each adapter used per-node input queuing with round-robin service to ensure that rate limits applied to one flow do not affect other flows from the same adapter. In all simulations, PAUSE was enabled.

VI. Results

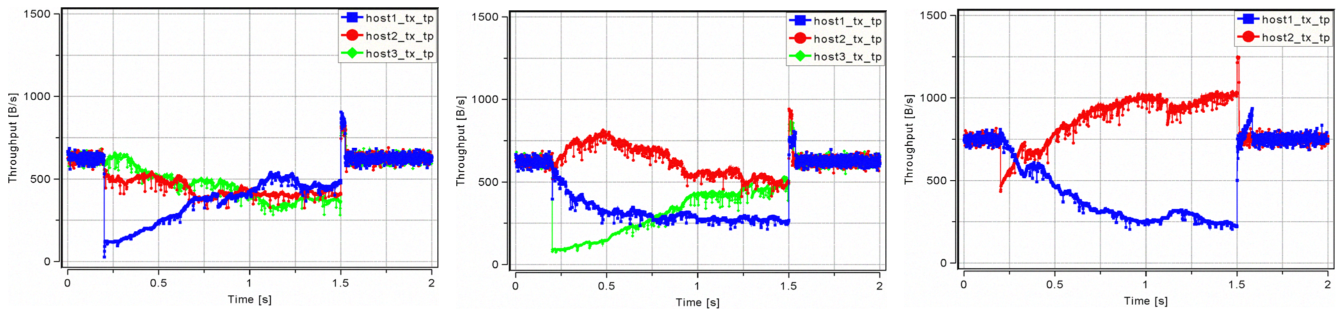
This section presents simulation results obtained for the noncontending- and contending-flow scenarios, as well as for uniform Bernoulli traffic.

A. Noncontending flows

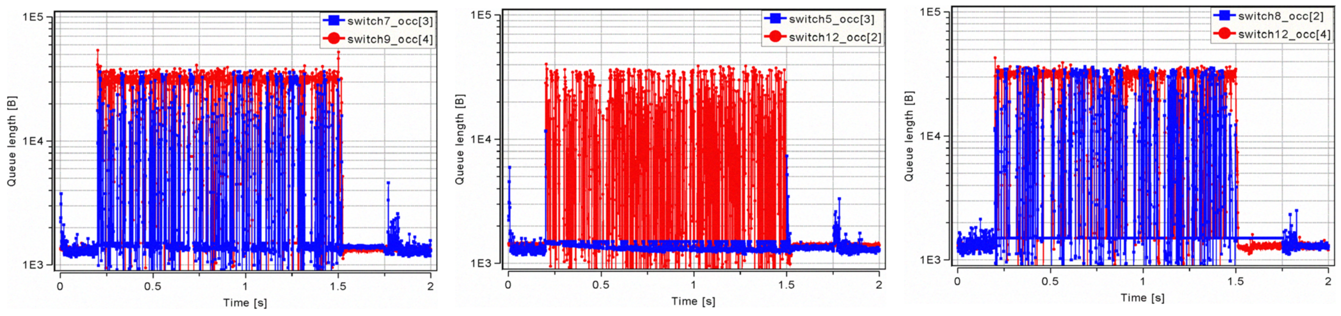
Figure 4 shows the results for the noncontending scenario. Figures 4(a–f) plot the mean transmission rates for **H1**, **H2**, and **H3** as a function of time, with Figs. 4(a–c) showing results using only CM and Figs. 4(d–f) using AR on top of CM. Figures 4(g–i) show the queue length of the hot queues in each scenario with and without CM.

Figures 4(a–c) show that CM by itself was able to adjust the transmission rates of the hot flows such that their sum matches the bottleneck link capacity. In addition, Figs. 4(g–i) show that the hot queue lengths converged on Q_{eq} with high accuracy and stability. However, by enabling AR, the throughput of each flow increased to 1187.5 MB/s, i.e., equal to the offered load. Congestion was avoided by routing each flow on a different path. Correspondingly, Figs. 4(g–i) show that the hot queues were no longer congested. Small spikes in the queue length occurred every 250 ms, because the timers periodically reset the congestion table entries. When a timer goes off, the switch will again route on the default path, which will cause congestion if another flow is also present there. This will lead to new CNs being generated, forcing the contending flow to again reroute onto an uncongested path.

In T3, see Figs. 4(c, f, i), we observed that the flow rate allocation was not fair, with one flow receiving significantly more bandwidth than the other. This, however, was due to QCN itself, which was not designed to converge on a fair flow rate allocation; its main objective is to stabilize the queue



(a) Throughput, AR, T1 (b) Throughput, AR, T2 (c) Throughput, AR, T3



(d) Queue length, T1 (e) Queue length, T2 (f) Queue length, T3

Fig. 5. Results for contending-flow scenario.

length.

Although occasional misorderings occurred, the overall effect on latency was negligible. The mean length of the resequencing buffers was less than one frame.

B. Contending flows

Figure 5 shows the results for the contending scenario using AR on top of CM, with Figs. 5(a–c) showing the per-host transmission rates, and Figs. 5(d–f) showing the hot queue lengths. In T1, these are in **S7** and **S9**, in T2 in **S5** and **S12**, and in T3 in **S8** and **S12**.

These results demonstrate that, although AR was not able to eliminate congestion, the underlying CM behaved as expected in terms of controlling the queue length by adjusting the rates of the contending flows. In these scenarios, two potential hotspots exist in each topology; therefore, the routes frequently changed from one to the other. This effect can also be seen in Fig. 6, which shows the length of the resequencing buffer at the destination node of the hot flows for each topology. The average resequencing buffer length is around 10 frames.

C. Uniform traffic

Figure 7 shows results for uniform traffic, with Figs. 7(a, c) plotting the measured throughput vs. the offered load and Figs. 7(b, d) plotting measured latency vs. offered load. Each figure contains curves corresponding to CM disabled (“None”), only CM enabled (“CM”), and both CM and AR enabled (“AR”).

First, we observed that without CM, throughput saturates at about 55.6%. This is because there were nine hot flows in each direction (the flows from hosts **H1–H3** to **H4–H6** and vice versa) contending for a bottleneck link, hence each

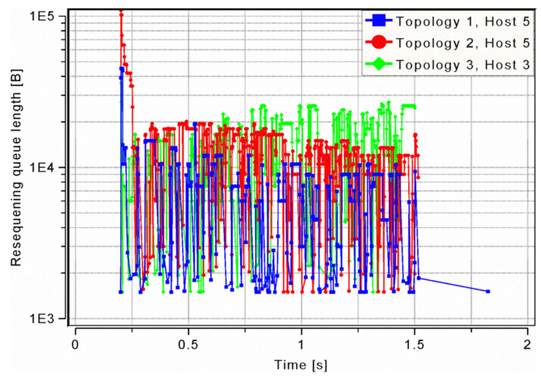
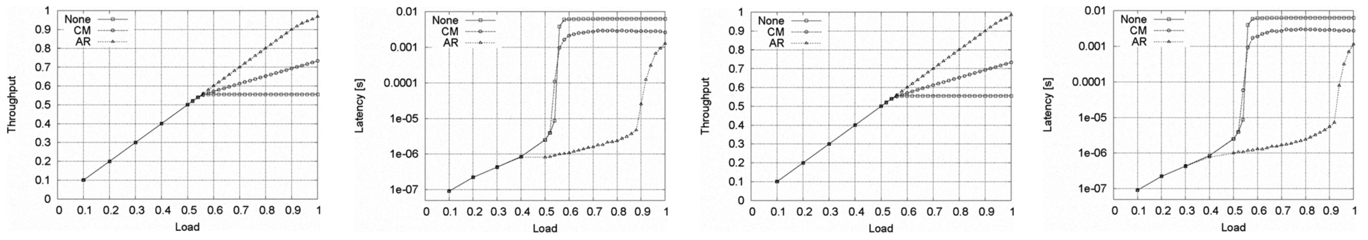


Fig. 6. Resequencing buffer length.

flow received one-ninth of the link rate. As CM was disabled, the input buffers of the congested switches filled up rapidly, causing PAUSE to be applied. Each of the cold flows (those among hosts **H1–H3** on one hand and **H4–H6** on the other) shared a link with at least one hot flow. For instance, in T1, all flows from hosts **H1–H3** traversed a congested link to switch **S7**, and all flows from hosts **H4–H6** traversed a congested link to switch **S9**. As a consequence, the cold flows effectively obtained the same throughput as the hot ones, leading to an aggregate throughput per host of $5 * \frac{1}{9} \equiv 55.6\%$.

Second, for both topologies we observed that enabling CM increased throughput. The CM mechanism controlled the congested queue lengths such that the links leading to the congested switch did not have to be continuously paused. Therefore, the cold flows were unaffected, resulting in an overall throughput of $3 * \frac{1}{9} + 2 * \frac{\lambda}{5}$ for $\lambda \geq \frac{5}{9}$, where λ is



(a) Throughput vs. load, T1 (b) Latency vs. load, T1 (c) Throughput vs. load, T2 (d) Latency vs. load, T2

Fig. 7. Uniform traffic

the offered load ratio.

Finally, we observed that enabling AR increased saturation throughput to 96.8% for T1 and to 98.5% for T2, indicating that our AR scheme optimally exploited the available path diversity. Accordingly, the mean latency is also drastically reduced.

Note that the latencies exhibited a maximum value instead of growing without bound as the load approached saturation, because we limited the length of the queues in the input adapters to prevent the simulation's memory usage from becoming excessively large.

VII. Conclusions

We proposed a fully adaptive routing scheme for IEEE-802.1Qau-compliant CEE networks, which to the best of our knowledge is the first scheme to prove the load balancing potential of the upcoming CM standard for CEE. It takes advantage of the presence of congestion management by modifying the switch routing behavior based on information snooped from congestion notifications. Our evaluation showed that the performance can be improved significantly (increased throughput, reduced latency) under both specific congestion scenarios and uniform traffic. This scheme provides a practical way to exploit the path diversity present in data center networks to its full extent.

The initial results encourage us to further exploit QCN's load sensing capabilities combined with AR for larger HPC fabrics. We plan to study the scheme using more realistic network topologies, e.g., meshes or fat trees, and either execution- or trace-driven traffic patterns.

References

[1] G. Pfister and V. Norton, "Hot spot contention and combining in multistage interconnection networks," *IEEE Trans. Computers*, vol. C-34, no. 10, pp. 933–938, Oct. 1985.

[2] 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 *Proc. HPI-DC 2005*, Research Triangle Park, NC, July 24 2005.

[3] D. Bergamasco, "Data Center Ethernet Congestion Management: Backward Congestion Notification," IEEE 802.1 Meeting, May 2005.

[4] D. Bergamasco and R. Pan, "Backward Congestion Notification Version 2.0," IEEE 802.1 Meeting, September 2005.

[5] R. Pan, B. Prabhakar, and A. Laxmikantha, "QCN: Quantized Congestion Notification," May 17 2007. [Online]. Available: http://www.ieee802.org/1/files/public/docs2007/au-prabhakar-qcn-description.pdf

[6] R. Pan, B. Prabhakar, and A. Laxmikantha, "QCN: Quantized Congestion Notification," May 29 2007. [Online]. Available: http://www.ieee802.org/1/files/public/docs2007/au-pan-qcn-details-053007.pdf

[7] R. Pan, "QCN Pseudo Code Version 2.2," Nov. 13, 2008. [Online]. Available: http://www.ieee802.org/1/files/public/docs2008/au-pan-QCN-pseudo-code-ver2-2.pdf

[8] C. Minkenberg, M. Gusat, "Congestion management for 10G Ethernet" in *Proc. Second Workshop on Interconnection Network Architectures: On-Chip, Multi-Chip* (INA-OCMC 2008), Göteborg, Sweden, Jan. 27, 2008

[9] M. Wadekar, "CN-SIM: Topologies and Workloads", Feb. 8, 2007, [Online]. Available: "http://www.ieee802.org/1/files/public/docs2007/au-sim-wadekar-reqd-extended-sim-list020807.pdf"

[10] W. Dally and C. Seitz, "Deadlock-free message routing in multiprocessor interconnection networks," *IEEE Trans. Computers*, vol. C-36, no. 5, pp. 547-553, May 1987.

[11] J. Duato, S. Yalamanchili, and L. Ni, "Interconnection Networks: An Engineering Approach," Morgan-Kaufmann Press, San Francisco, CA, revised printing, 2003.

[12] J. Duato, "A new theory of deadlock-free adaptive routing in wormhole routing networks," *IEEE Trans. Parallel Distributed Syst.*, vol. 4, no. 12, pp. 1320–1331, 1993.

[13] J. Duato and P. Lopez, "Performance evaluation of adaptive routing algorithms for k-ary-n-cubes," in *Proc. First Int'l Workshop on Parallel Computer Routing and Communication*, 1994.

[14] A. Chien and J. Kim, "Planar-adaptive routing: Low-cost adaptive networks for multiprocessors," in *Proc. Int'l Symp. on Computer Architecture*, J. ACM, vol. 42, no. 1, pp. 91–123, 1992.

[15] C. Glass and L. Ni, "The turn model for adaptive routing," in *Proc. 19th Int'l Symp. on Computer Architecture*, pp. 278–287, 1992.

[16] I. Gopal, "Prevention of store and forward deadlock in computer network," *IEEE Trans. Commun.*, vol. COM-33, no. 12, pp. 1258–1264, Dec. 1985.

[17] X. Lin, P. McKinley, and L. Ni, "The message flow model for routing in wormhole-routed networks," in *Proc. 1993 Int'l Conf. on Parallel Processing*, pp. 1-294-1-297, Aug. 1993.

[18] F. Safaei, A. Khonsari, M. Fathy, and M. Ould-Khaoua, "Performance comparison of routing algorithms in wormhole- switched fault-tolerant interconnect networks," in *Proc. Int'l Conf. on Network and Parallel Computing* (NPC), 2006, Japan.

[19] M. Al-Fares, A. Loukissas, and A. Vahdat, Amin, "A scalable, commodity data center network architecture," in *Proc. ACM SIGCOMM 2008 Conference on Data Communication*, Seattle, WA, Aug. 17-22, 2008, pp. 64-74.