CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. 2018; **00**:1–25
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/cpe.4440

Non-minimal Adaptive Routing Based on Explicit Congestion Notifications

Mariano Benito[1,2*], Enrique Vallejo[1], Cruz Izu[3] and Ramón Beivide[1]

[1] *University of Cantabria, Dept. Computer Science and Electronics, Cantabria, Spain*
[2] *Recore Systems BV, Enschede, Overijssel, The Netherlands*
[3] *The University of Adelaide, Adelaide, Australia*

SUMMARY

Low-diameter networks require non-minimal adaptive routing to deal with varying traffic characteristics and avoid pathological performance. Such routing is based on local estimations of network congestion, based on link-level flow control credits. Dragonfly networks based on extensions of commodity Ethernet networks using OpenFlow have been proposed for large HPC deployments with low power consumption. However, this network technology does not implement credit-based flow control.
This work explores a range of routing solutions based on exploiting explicit congestion notification messages (in particular, 802.1Qau) to adapt the number of packets using non-minimal paths. The design (denoted *QCN-Switch*) associates a probability value to each output port. This value is updated to reflect downstream congestion and used to statistically divert traffic away from congested areas when the load is uneven, as in the case of adversarial traffic. A *feedback comparison* variant is designed to separate the cases of uniform traffic at saturation and adversarial traffic at low loads. Evaluation results show that *QCN-Switch* is a competitive design for both uniform and adversarial traffic. Furthermore, it is able to react to changes in traffic conditions in 0.4 ms or less. A sensitivity analysis identifies the best configuration and shows its performance trade-offs.

Received . . .

KEY WORDS: adaptive routing; explicit congestion notifications; dragonfly

1. INTRODUCTION

The current trend towards massive-scale Datacenters and the effort to reach Exascale computers has driven the introduction of low-diameter, highly scalable network topologies, such as Dragonflies [1], SlimFlies [2] or Projective Networks [3]. Such networks sacrifice minimal-length path diversity, common in traditional topologies such as Folded-Clos, for larger number of nodes and lower average distance. Consequently, minimal paths can be relatively easily congested under adversarial traffic patterns, thus requiring the use of non-minimal paths (using Valiant routing [4]) or non-minimal adaptive routing that selects between minimal and non-minimal paths in response to traffic conditions ([1, 5, 6]). These routing mechanisms try to avoid congested areas, so that network delay is minimized and throughput can be sustained by balancing the use of network resources.

Most of the previous proposals rely on comparing congestion indicators between two or more paths to select the most appropriate one. These congestion indicators are typically derived from buffer occupancy, either directly measured from the credit count of the link-level flow control mechanism of its neighbour switches, or notified by the network in case of congestion in remote areas (in some form of Explicit Congestion Notification, ECN) in addition to credit counters.

*Correspondence to: Mariano Benito, Department of Computer Science and Electronic, Facultad de Ciencias, Universidad de Cantabria, Avda. Los Castros s/n 39005, Cantabria, Spain. Email: mariano.benito@unican.es

In this work we explore the design of a non-minimal adaptive routing mechanism based *solely* on explicit network congestion notifications. There are multiple reasons that motivate this approach: firstly, some network technologies, such as lossless Ethernet (which is widely employed in Datacenters and has been considered for large HPC systems [7]), do not implement link-level flow control credits (Ethernet is based on PAUSE messages, 802.1Qbb), which makes direct sensing of buffer occupancy unfeasible. Secondly, even if credit counters are used, remote congestion notification messages send information about the status of remote areas of the network that might suffer congestion, which simplifies source-based adaptive routing. Finally, some form of remote congestion notification is needed in any case for injection throttling to react to endpoint congestion.

Congestion notification messages are received in much coarser intervals than direct measurements such as link-level credit information. For this reason, simply using them to adapt routing tables might generate oscillations between minimal and non-minimal paths. The approach in this work is to route each packet individually, selecting between minimal and non-minimal paths based on a statistical value which is derived from recent congestion notification messages. In particular, the probability of following a given minimal path varies according to the reception or absence of congestion notifications. The approach is implemented in a simulation model of an OpenFlow switch based on the proposal in [7], modeling a Dragonfly network [1] and using Quantized Congestion Notification (QCN, [8]) for ECN messages. Different approaches are considered, including the congestion detection point and its impact on latency and network fairness, the selection of parameters (both in the adaptive routing mechanism and the congestion notification messages) and the use of congestion value averaging to avoid throughput drops at high load. Coarse-grain notifications also impact reaction time when traffic changes; although it is not possible to react in the same time as mechanisms with direct credit sensing, by carefully tuning QCN parameters the system can adapt in a time scale which is acceptable for HPC applications.

A preliminary version of this work was presented in [9]. While that work introduced the idea of randomizing path selection based on congestion notification and presented some preliminary analysis on input-buffer congestion sensing, it did not provide a completely successful mechanism in terms of latency, throughput and fairness. By contrast, this work extends the analysis to output-buffer congestion sensing, observing that such approach eliminates the problem of switches having to react to congestion in their own input ports. In particular, the contributions of this paper are:

- It presents an overall working solution for non-minimal adaptive routing based on congestion notification messages, denoted *QCN-Switch*.
- It presents a thorough evaluation of the proposals, implemented in an Ethernet switch model using QCN, observing that sensing congestion in output buffers works better than sensing at the inputs, and that a comparison of congestion of different ports allows to better identify and manage different congestion situations under uniform and non-uniform traffic loads.
- It analyses the impact of the parameters of the congestion detection mechanism and the reaction policy on router response time, observing that competitive results are feasible without relying on flow control credits information.

The remainder of this paper is organized as follows: first, we describe background and motivation in Section 2. Then, in Section 3 we present our design, Section 4 describes the evaluation methodology and Section 5 presents performance results for a range of loads and configurations. Some related work will be presented in Section 6, concluding this paper in Section 7.

2. BACKGROUND AND MOTIVATION

This section introduces the background in terms of network topology, switch architecture, and the QCN mechanism in which our adaptive high radix network is developed.

DOI: 10.1002/cpe.4440

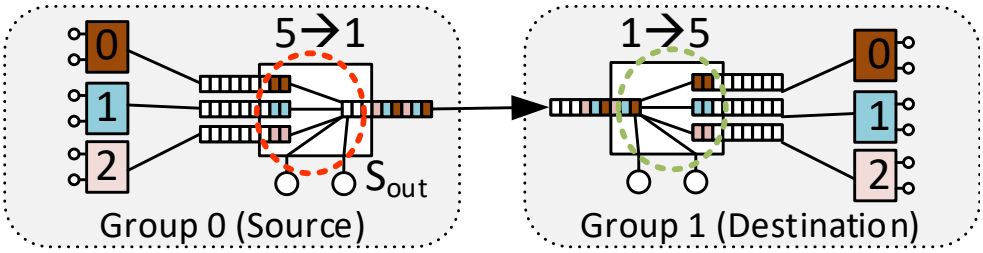


Figure 1. Adversarial (ADV) traffic in a Dragonfly network. All traffic from Group 0 goes to Group 1, making the global link that joins them a bottleneck. Buffers in local ports in S_{out} will get full, reflecting the link's congestion.

2.1. *Dragonfly topology and routing*

This work considers a Dragonfly network [1], which has been employed for commercial products [10, 11], although the proposal could be generalized to other network topologies. A Dragonfly network comprises multiple groups of switches interconnected in a hierarchical direct topology. Each group of a switches (with p nodes per switch) is directly connected by means of local links, while different groups are connected by h global links per switch.

Minimal routing in the Dragonfly is hierarchical, first to the destination group (a local hop, followed by a global hop), and then making a local hop to the destination switch. Under *uniform* (UN) traffic pattern, in which the destination of each frame is any terminal in the network, minimal routing provides optimal throughput and latency. Figure 1 presents a situation of *adversarial* (ADV) traffic. When the nodes in a given group sent all their traffic to the same destination group, the only global link between these two groups, departing from the switch denoted S_{out}, becomes the bottleneck under minimal routing. Such traffic pattern should use non-minimal paths to avoid overloading that global link and increase network performance.

Valiant routing [4] randomizes network traffic by sending each packet first to a random intermediate network switch, and then to its final destination. This doubles the average network distance, but makes a balanced used of network links under any traffic pattern. Therefore, it improves network throughput for adversarial traffic patterns when compared to minimal routing.

Non-minimal adaptive routing adjusts to network conditions by alternating the use of minimal and non-minimal paths: when congestion is low most packets are sent using minimal paths. However, as congestion increases in the minimal paths, a higher percentage of messages should be diverted to less congested non-minimal paths.

2.2. *Switch architecture*

Most current Datacenter and HPC switches employ a combined input-output queued (CIOQ) architecture, often combined with Virtual Output Queues (VOQs). In such architecture, input and output ports employ one or more buffers, typically organized in virtual channels (using individual storage and flow control). A crossbar fabric connects input and output ports, using certain frequency speedup to compensate for inefficiencies in allocation. Other parameters of the considered architecture are presented in Section 4.

Our switch model resembles an Ethernet OpenFlow switch. Per input-buffer flow control is supported by Priority-based Flow Control (PFC), which supports individual pauses per buffer but does not handle buffer credits. Routing is implemented using a flow forwarding table which contains multiple flow entries, which are instantiated proactively. The application of this Ethernet technology to build Dragonfly networks follows the ideas introduced in [7], including hierarchical MAC address rewriting and TCAM compaction, which are required for the mechanism to work but are not relevant for this study. Deadlock avoidance in the Dragonfly topology is guaranteed by leveraging 802.1p classes of service to avoid cyclic dependencies, resembling the ordering in [1].

DOI: 10.1002/cpe.4440

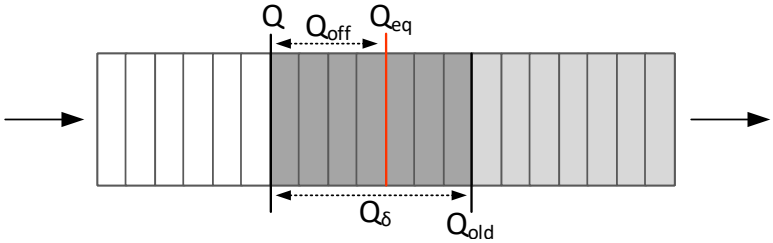


Figure 2. Representation of QCN buffer state variables at the congestion point.

Adaptive routing is implemented using conditional OpenFlow rules, first proposed for non-minimal adaptive routing in [7]. The conditional decision in that original proposal relied on the PAUSE status of each output port, but such implementation presents poor throughput under uniform traffic at high loads and poor fairness and average latency under adversarial traffic, as shown in [7]. Instead of pauses, our proposal characterizes each output port with a percentage value, which defines the probability of minimal routing through that output. This design was proposed and partially evaluated in [9] and will be presented in more detail in Section 3.

2.3. *Quantized Congestion Notification*

Detection of congestion in lossless networks relies on either buffer occupancy values or increases of flow latency. Explicit congestion notifications can be generated by the network switches, typically triggered by the evolution of its buffers occupancy. Such explicit notifications can be piggybacked on the data frames of the flow that suffers congestion, such as IP's ECN bits [12] of Infiniband's FECN/BECN bits [13], or they can be independent messages such as the CNMs from QCN described next.

Quantized Congestion Notification (QCN, [8]) implements congestion notification in Layer-2 Ethernet Datacenter Networks. It is mainly composed of two elements, congestion points (CP, in switches) and reaction points (RP, rate limiters at NICs). Congestion points generate explicit Congestion Notification Messages (CNMs) when a given buffer suffers a congestion situation. In the reference implementation from the standard [8] CPs are located in output buffers, but another work [14] suggests sampling input buffers; both approaches are considered in our evaluation section to drive adaptive routing.

To identify a congested situation, two state variables are combined; position (Q_{off}) and velocity (Q_{δ}) of the sampled buffer occupancy, as illustrated in Figure 2. CNMs do not indicate the mere presence of congestion, but instead carry a feedback value (Fb) that quantifies it. Each buffer is assigned a reference length denoted Q_{eq}, or equilibrium point. Q denotes the instantaneous buffer occupancy sampled every 100 packets and Q_{old} denotes the buffer length when the previous CNM was generated. Considering $Q_{off} = Q - Q_{eq}$, $Q_{\delta} = Q - Q_{old}$, and w a constant weight, set to 2 by default, Fb is calculated according to the following expression:

$$Fb = -(Q_{off} + w \times Q_{\delta})$$

Negative values indicate the presence of congestion and generate CNMs. The Fb value in the CNM is quantized to 6 bits, ranging from 0 to 63, saturating with an occupancy of $2 \times Q_{eq}$. This value is sent to the source of the packet sampled in the switch buffer, this is, a source NIC. RPs at NICs implement injection throttling, using a mechanism based on an Additive-Increase, Multiplicative-Decrease (AIMD) policy. When negative congestion notifications are received, the feedback value Fb is used to divide the injection rate, adjusted by a factor L_f. QCN does not implement positive notifications (lack of congestion), so the additive increase policy is applied when no notifications are received for a period corresponding to 100 frames.

DOI: 10.1002/cpe.4440

3. ADAPTIVE ROUTING DESIGN BASED ON EXPLICIT CONGESTION NOTIFICATIONS

This section introduces statistical source adaptive routing relying on explicit congestion notifications, extending the preliminary version introduced in [9].

Many previous mechanisms implement table-based adaptive routing that performs load balancing. In previous proposals, balancing is typically performed with the granularity of a packet [1, 5, 6], flowlet [15] or flow [16, 17, 18, 19]. However, those proposals either employ a congestion-oblivious balancing function (so they do not really adapt to changing network congestion, and are not feasible for non-minimal adaptive routing), do not support non-minimal adaptive routing or rely on packet-by-packet network information such as credit counters, with minimal feedback delay.

Our design adapts routing to network traffic by capitalizing on QCN congestion messages generated when transit buffers are heavily used. Congestion notifications update network status with a coarse granularity, in the order of hundreds of packets. This is affordable for injection throttling, because injection rate is modulated in an almost-continuous range, from 0 to the maximum interface rate. By contrast, table-based adaptive routing needs to select the destination between a small set of forwarding entries, so large notification delays would fix the selection of a given path and overload some network area. Indeed, a large feedback delay combined with a deterministic selection function in the forwarding table may cause abrupt traffic oscillations.

Our proposal does not implement the Reaction Point in the NICs. Instead, source switches (those that are directly connected to the destination NIC) intercept QCN notifications and modify the probability of forwarding traffic minimally through each of its output ports, as detailed in subsection 3.1. As previously stated, the *base* mechanism modulates the probability of each output according to an Additive-Increase, Multiplicative-Decrease (AIMD) policy, and is denoted *QCN-Switch*. The base design is extended in subsection 3.2 with *feedback comparison* targeting the issue of throughput fall at uniform high loads. Finally, section 3.3 discusses the implications of sampling location of the congestion point within network switches.

3.1. Statistical non-minimal routing based on congestion notifications: QCN-Switch

3.1.1. Forwarding tables with probabilities. To provide a smooth balance in the utilization of different network paths between the reception of consecutive congestion notifications, this work relies on conditional forwarding entries [7] with *statistical* path selection. Path selection is modulated with a set of control values derived from the congestion status reported by the network. Such control values are calculated to set the *probability* of forwarding traffic minimally using a given output port, so one value is required per transit port. These values are denoted "minimal port probability" (or simply, port probability), and their range is 0-100%.

To implement the statistical path selection, we consider routing table entries with priorities. Priorities are used to discern minimal and non-minimal forwarding table entries: high-priority conditional rules are used for minimal routing and low priority rules are used for non-minimal routing. Routing is calculated so that, when path selection is possible, one high and at least one low priority entries match a given packet. In our evaluation in Section 4 we consider source-adaptive routing, so path selection only occurs at injection and non-minimal rules are applied only for injection ports, but the model might be applied to in-transit adaptive routing. When only one rule match occurs, such rule is applied regardless of its probability value.

A random number N [†] is employed when multiple rules with different priorities match a packet, as follows. If N exceeds the probability of the output port of a matching high-priority conditional rule, this rule is not executed. Instead, the matching lower-priority rule is followed.

Figure 3 shows an example of the proposed routing table and associated port probabilities for one switch in a Dragonfly group. The switch has five transit ports: ports 3-5 connect to other switches in the same Dragonfly group, and ports 6-7 connect to remote groups. Table entries for minimal routing have *high* priority and employ a condition that is associated to the probability of their output port;

[†]The random value N may be generated from the incoming frame (for example, by selecting some bits from its CRC) or derived from a pseudo-random sequence.

DOI: 10.1002/cpe.4440

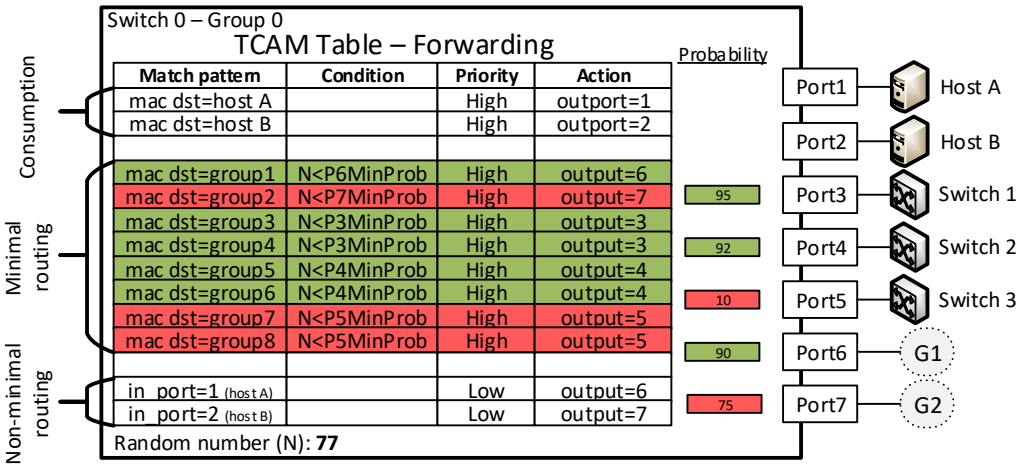


Figure 3. Switch architecture with a generic QCN-Switch proposal. This diagram portrays Switch 0 of Group 0 under adversarial traffic pattern for a $(p = 2, a = 4, h = 2)$ Dragonfly network. The condition or a rule fails when N is higher than the probability of sending minimally by its associated port. In such case, high-priority minimal routing rules are ignored (highlighted in red), leading to the use of a low-priority rule.

table entries for non-minimal routing have *low* priority and are not conditional. In this example, port 5 has received large congestion notifications and has reduced its probability to a low value (10%).

The random value generated for the query shown in figure 3 is $N = 77$. Thus, probability values lower than N (the associated table entries are highlighted in red) will disable minimal routing paths via ports 5 and 7 which are known to be congested. When the condition is false an alternative non-minimal path is selected (using the two entries in the lower part of the table) based on the input port of the packets[‡]. If no alternative entry exists, the condition will be ignored; this avoids non-minimal forwarding of in-transit traffic and creating forwarding loops.

The probability of each conditional rule needs to be adapted to the congestion level in the corresponding path, which is estimated from the feedback values received in the QCN congestion notification messages. Different policies may be considered to update the probability values of each port; we present the base mechanism next, and one extension in the following subsection.

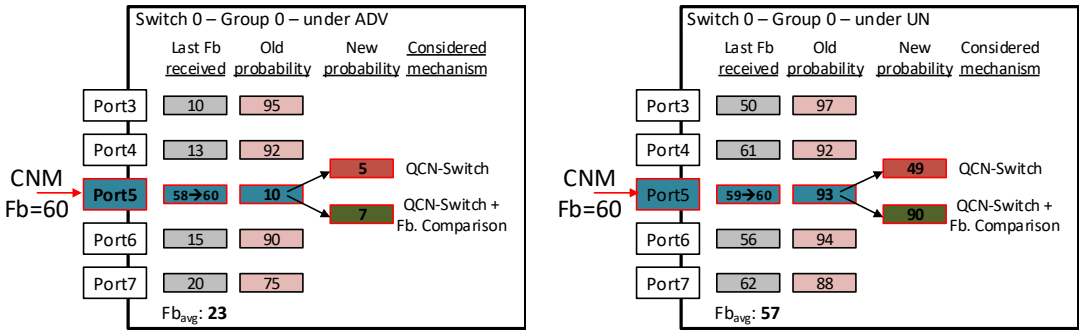
3.1.2. QCN-Switch base*: AIMD probability management driven by QCN messages*

The *base* policy employs an AIMD policy to reduce the probability associated to an output port when a CNM is received through it, modulated by the Feedback value Fb indicated in the message. Since QCN only contains negative congestion notification messages, but it does not notify the absence of congestion, the protocol is implemented as follows:

- Upon reception of a CNM with feedback Fb, the probability value is multiplied by a factor R, calculated as:
$$R = 1 - |L_f \times Fb|,$$
where L_f is a limiting factor that determines the extent to which the probability decreases and Fb values range from 0 to 63 as presented in Section 2.3. For example, with $L_f = 1 \,/\, 128$, R will be in the range between 0.5 and 1.
- A counter tracks the packets transmitted by each transit port between the reception of CNM messages. The counter is reset every time a CNM message is received and it is incremented for every transmitted packet. If it reaches a threshold PC_l, the probability value associated with that port is increased by $PI\%$.

[‡]Note that in certain cases the alternative rule might overlap the minimal path.

DOI: 10.1002/cpe.4440



(a) Adversarial traffic pattern. (b) Uniform traffic pattern with high load.

Figure 4. Example of probability values update when CNM with Fb equal to 60 arrives, under two different traffic scenarios for a switch in a ($p = 2, a = 4, h = 2$) Dragonfly network. Under uniform traffic the port probability remains high when using the *feedback comparison* variant. Changing values are shown in red.

Parameters L_f and PI regulate the reaction time of this mechanism to changes in congestion: L_f determines how fast traffic is sent non-minimally after congestion is detected; analogously, PI indicates how quickly routers revert to minimal routing after congestion disappears. The impact of these parameters is evaluated in subsections 5.3.2 and 5.3.3.

3.2. Feedback comparison

Non-minimal adaptive routing needs to react to network traffic conditions to avoid congestion under adversarial traffic patterns. However, the information received from a single individual port is not enough to accurately identify an adversarial traffic pattern. We should note that an adversarial traffic pattern will cause congestion in one or a low number of ports while a heavily loaded uniform load will cause congestion in all the ports.

When the network is heavily loaded under a uniform traffic pattern, all network buffers get similarly loaded and all switches generate some congestion notifications. In this case, switching to non-minimal routing because notifications are received from an individual port is counterproductive: non-minimal routing increases the load in the network, which eventually generates more congestion notifications and further non-minimal routing, in a positive control loop. The effect is that all traffic tends to be forwarded non-minimally, with reduced throughput and increased latency.

Feedback comparison calculates an average feedback value Fb_{avg} which represents the average congestion of all transit ports of a switch. This value is calculated from the most recent feedback values $Last\ Fb_i$ received on each of its transit ports according to the following expression:

$$Fb_{avg} = \frac{\sum_{i=transit\ ports} |Last\ Fb_i|}{i}$$

Upon reception of each QCN message, the switch recalculates Fb_{avg} and compares this average with the feedback received in the CNM. If $|Fb| < Fb_{avg}$, the probability associated to that port is increased by $PI\%$, same as occurs when no CNM is received in an interval in *QCN-Switch*. If $|Fb|$ is greater, the port's probability is reduced by the factor:

$$R = 1 - L_f \times (Fb - Fb_{avg})$$

Figure 4 shows an example of the update of probabilities using both *QCN-Switch* or *QCN-Switch + feedback comparison*, under uniform or adversarial traffic. The use of Fb_{avg} to calculate the reduction factor R in the latter reduces the impact of congestion notifications on uniform loads, when all ports experience similarly high congestion values.

Buffer occupancy naturally suffers transient variations, which is denoted transient congestion in the literature [20]. Therefore, congestion notification feedback values may change significantly

DOI: 10.1002/cpe.4440

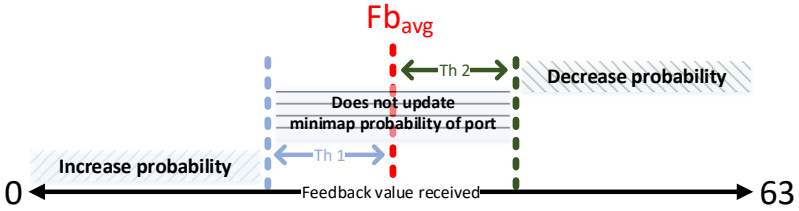


Figure 5. Thresholds used in the *feedback-comparison* variant.

even in a stationary state, which may increase variability. The *feedback comparison* model employs two thresholds Th_1 and Th_2 to eliminate minor changes due to such transient variations. Port probabilities are not modified when the feedback value received in a notification lays within the range $(Fb_{avg} - Th_1, Fb_{avg} + Th_2)$, as depicted in Figure 5. The impact of these threshold values is evaluated in subsection 5.3.4.

When network congestion disappears CNMs might be no longer received given the lack of positive (lack of congestion) notifications in QCN. In this case, Fb_{avg} might remain with a high value with the previous implementation. While this potential issue has not been observed in our simulations (a case of congestion disappearing is presented in Figure 10b), a realistic implementation would need to adapt Fb_{avg}, for example reducing each $Last\ Fb_i$ according to a negative exponential decrease when no CNMs are received.

3.3. Alternatives for buffer sampling and congestion detection

QCN samples the occupancy of switch buffers according to the definition presented in subsection 2.3. When considering a CIOQ switch architecture, occupancy sampling can be performed at input or output buffers. These two alternatives are described next.

- *Congestion detection at switch input buffers*: Neeser *et al* [14] suggested the use of input-buffer occupancy as a congestion indicator, which provides better fairness and improved detection of offending and victim flows. When generating a CNM, they also selected a victim packet randomly from the complete buffer, not necessarily the packet in the head, in order to improve fairness.
- *Congestion detection at switch output buffers*: the original implementation of QCN [8] considers occupancy sampling in the output ports of the switch. As discussed in the next subsection, one advantage of this alternative is that generated CNMs can be sent to all sources in the network regardless of their source switch.

The use of input-buffer sampling to generate CNMs that modify routing has been evaluated in our previous work [9]. That work identified unfairness issues, particularly under adversarial traffic. The next subsection explains the source for such unfairness and proposes a modification to the base mechanism under input sampling to address this issue. The evaluation of both sampling alternatives are presented in Section 5.1.1, including a complete implementation which combines *QCN-Switch* with *source processing* and *feedback comparison*.

3.3.1. Source processing

Depending on the switch speedup, congestion trees can be rooted at input or output ports [21], but with sufficient speedup this typically occurs at the outputs. When congestion is detected, the notification message is sent to the source NIC of one packet in the congested buffer.

In the adaptive routing implementation introduced in Section 3.1, switches intercept CNMs in order to adapt routing. However, a pathological case of unfairness can occur, specifically when congestion sensing is implemented at the input buffers. Consider the situation of adversarial traffic presented in Figure 1. Congestion occurs in the minimal path, saturating input buffers in local ports and the output buffer in the global port of S_{out}.

DOI: 10.1002/cpe.4440

With output-buffer sampling, S_{out} generates CNMs indicating congestion associated to the output buffer of its global port; these CNMs are sent to all the eight nodes in the example, and they are intercepted by their associated access switch (0, 1, 2 or S_{out} itself). In the case of S_{out}, this CNM reduces the probability of the output port that generates the CNM, whereas in the neighbor switches the probability that is reduced is associated to the port that receives the CNM.

By contrast, with input-buffer sampling, CNMs are generated based on the occupancy of the three buffers associated to local ports, and they are sent only to the neighbor switches. No CNM targets the two computing nodes associated to S_{out}, because their input buffers are not associated to transit ports. Additionally, if the message was generated, it would need to be processed locally without information about which is the congested output port. Therefore, S_{out} never modifies its own output port probabilities and always sends traffic minimally, suffering much higher congestion. For this reason, nodes in S_{out} may inject a much lower amount of traffic.

The *source processing* variant attacks this limitation by making all switches process their own CNMs generated based on congestion at input ports. When a CNM is generated by a QCN Congestion Point in a switch, it is sent to the source as usual but it is also processed in that same switch, as if it had been received from a neighbor switch. In this case, the feedback of this QCN message is used to decrease the probability of the output port used to forward the "victim packet" selected in the sampling process. This allows all switches in a group to detect congestion and adapt their tables in response.

4. EVALUATION METHODOLOGY

This section introduces the simulation tool used and its configuration parameters. The network simulator described in subsection 4.1 is used to implement and compare the different QCN-base adaptive routers with previous proposals as listed in subsection 4.2 under a range of network loads, as described in subsection 4.3.

4.1. Simulation environment

We employ the FOGSim network simulator [22] to evaluate the performance of non-minimal adaptive routing using the mechanisms proposed in Section 3. We model a Dragonfly network with $p = 6$ terminals per switch, $a = 12$ switches per group and $h = 6$ global links per switch, leading to 5,256 terminals, which is representative of realistic HPC systems. We select 1 KB for packet size as an intermediate value between minimum and maximum packet size for Ethernet technology. Link latencies model cables of 8 or 80 meters for local and global links respectively. The cycle-accurate simulator models an input-output-buffered router. Four Ethernet CoS levels are considered, implemented as virtual channels (using per-priority flow control for a lossless implementation), and used for deadlock avoidance and to prioritize QCN control messages.

QCN *Congestion Points* (CPs) have been implemented following the standard [23], including the feedback calculation and timing parameters indicated in Section 2.3. We use the recommendation to set Q_{eq} to 20% the size of the physical buffer and we maintain the selection of 1 packet from each 100 for the sampling in the *Congestion Point* (denoted Congestion Point Cycle, CPC); this parameter is part of our sensitivity analysis during the evaluation. Another parameter for this analysis is the percentage of samples with a negative feedback value that actually generates a CNM; we consider a default value of 30%, and justify this selection in subsection 5.3.1. QCN CPs are implemented at the input ports of the switches, as suggested in [14], or at the output ports [23]. Preliminary explorations found that a parameter $w = 0$ led to better results than the default $w = 2$ in QCN. QCN *Reaction Points* (RPs), which implement injection throttling in the NICs, are disabled.

Table I shows the network parameters, the QCN standard parameters and the parameters applied to the QCN-Switch as described in Subsection 3.1.2. Any changes of the default values shown in this table will be clearly stated during the evaluation. We employ a reduction limiting factor $L_f = 1/64$, which allows for output port percentage reductions in a factor up to $R = 1 - 63/64 = 0.0156$, this is, very fast transition from minimal to non-minimal routing. By contrast, we employ a default

DOI: 10.1002/cpe.4440

Table I. Simulation parameters.

	Parameter	Value
Network configuration	Total end terminals	5,256 hosts
	Topology	Dragonfly
	Groups	73 groups
	Switches per group	12 switches
	Switch degree	23 ports
	Link speed	40 Gbps
	Packet size	1,000 bytes
	Switch frequency	1 GHz
	Internal crossbar speedupd	$2\times$
	Switch latency	200 ns
	Local/Global link latency	40/400 ns (8/80 m)
	CoS levels	4
	Injection queues size	200 KBytes
	Transit queue size	100 KBytes
QCN	Queue's reference point (Q_{eq})	20% of queue size
	Weight value (w)	0
	Congestion point cycle (CPC)	100 packets
	% of samples which actually generate a CNM ($\%CNMs$)	30%
QCN-Switch	Reduction limiting factor (L_f)	1 / 64
	Packet counter limit (PC_l)	100 frames
	% Probability increase (PI)	1 %
	Feedback-compare threshold 1 (Th_1)	0
	Feedback-compare threshold 2 (Th_2)	30

probability increase of $PI = 1\%$, which corresponds to a slow return to minimal routing when congestion disappears. Such design is sensible, since minimal routing in presence of adversarial traffic in Dragonflies is much more limiting than non-minimal routing under benign (i.e. uniform) traffic. This is explored in sections 5.3.2 and 5.3.3.

4.2. Routing mechanisms

We evaluate our non-minimal conditional routing based on QCN snooping, *QCN-Switch*, plus the three different variations denoted *QCN-Switch + Source processing*, *QCN-Switch + Feedback comparison* and *QCN-Switch + Source P. + Fb. Comparison*. We compare these adaptive routing schemes with *Minimal* (MIN) and *Valiant* (VAL, [4]) routing as they provide best response under UN and ADV traffic respectively. Additionally, we include *Piggyback* routing (PB) [5] as an adaptive reference that implements per-packet adaptive routing relying on state information for every global channel in its group distributed among switches. As PB routing relies on per-port flow control credit information, it is *unfeasible* in certain technologies such as Ethernet.

4.3. Traffic patterns

We feed the network with synthetic traffic. Each node injects frames according to a Bernoulli process with a variable load, alike other network simulators [24]. Firstly, two traffic patterns have been considered as they represent the best and worst case for this topology:

DOI: 10.1002/cpe.4440

- Uniform (UN), in which the destination of a frame is any other network node,
- Adversarial (ADV), in which the destination of a frame is randomly selected from all nodes in the consecutive group. ADV traffic patterns concentrates the traffic on a single global link between two groups, so non-minimal routing is required to obtain a good performance.

Furthermore, we evaluate the network response, using the average of 10 simulations, under two different types of network load:

- Steady loads: performance evaluations inject traffic at a given injection load, ranging from 0 to 100% following one of the patterns. After a warm-up phase of $2,4$ ms, we measure average latency and throughput for the next $2,4$ ms, also including fairness metrics. Note that this is the standard load used for most network evaluations under synthetic traffic.
- Transient loads: we also run simulations for a fixed offered load of 40% at which no traffic pattern saturates, starting with a warm-up phase of $2,4ms$ with *uniform* traffic, then changing the pattern to *adversarial* (or vice versa) for the next $2,4ms$. We denote the point at which traffic pattern changes as time 0, and record packet latency as well as port probability values and missrouted packets before and after that point in intervals of $20\mu s$. Such simulation allows us to determine how quickly each adaptive routing mechanism reacts to load changes; in other words, it measures its response time.

Two additional patterns are considered in Section 5.4 to further evaluate the best *QCN-Switch* configuration:

- Hotregion (HR), in which 25% of the traffic goes to the first 12.5% endpoints, and the remaining is distributed as UN (including the first 12.5% endpoints). This is the only considered pattern that generates endpoint congestion.
- Random permutation (RND), in which each node sends traffic to a given random destination, which remains fixed for the simulation, such that only one source sends traffic to a given destination.

5. EVALUATION RESULTS

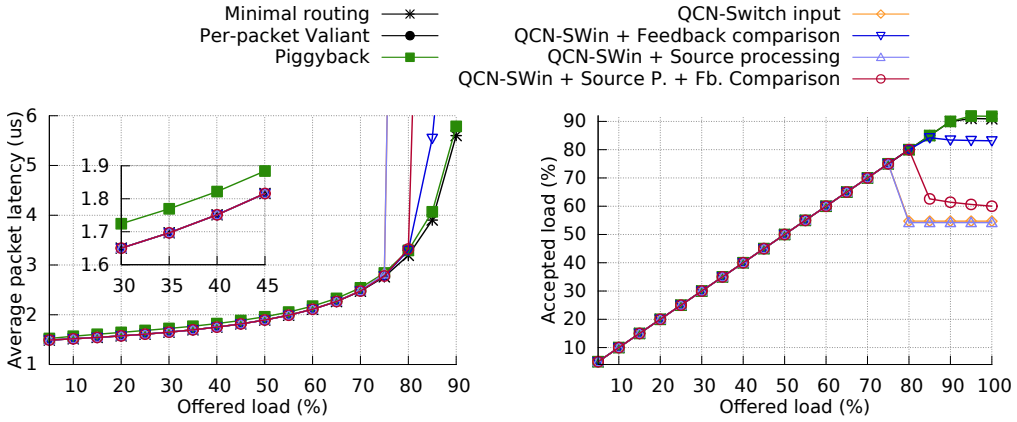
This section presents the performance results of our proposals. First, it considers steady traffic, measuring throughput, average packet latency and fairness. This first evaluation shows that *QCN-Switch with feedback comparison* doing the sampling at output queues exhibits the best results; thus, later subsections focus only on that configuration. Next, we evaluate the response time to traffic changes from *uniform* to *adversarial* traffic pattern, and vice versa. A sensitivity analysis is presented next for the most important parameters of our implementation. Finally, performance under alternative traffic patterns is evaluated.

5.1. Performance under steady loads

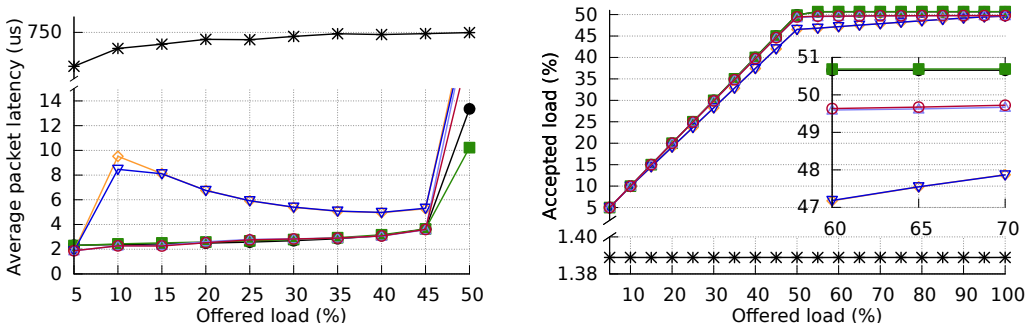
This subsection presents average latency and throughput results under two steady loads: *uniform* (UN) and *adversarial* (ADV) traffic patterns Minimal routing is presented as a reference for UN traffic and Valiant routing as the reference for ADV. We separately evaluate QCN sampling at input and output queues, considering all possible variations.

Note that throughput under ADV traffic is limited to $1/72 = 0.01389$ phits/node/cycle when using minimal routing (because only one of the 72 global links per group is used) and to 0.5 phits/node/cycle when using non-minimal routing (because of the randomization of traffic which increases the load in the network). This performance limitation under minimal routing is an effect of the topology, not of congestion. For this reason, using the original implementation of QCN for injection throttling under adversarial traffic would result in similar or worse performance than MIN, and therefore it is not plotted for simplicity.

DOI: 10.1002/cpe.4440



(a) Average Latency and Throughput under uniform (UN) traffic



(b) Average Latency and Throughput under adversarial (ADV) traffic.

Figure 6. Average latency and throughput of the proposed mechanisms with input-port sampling under (a) uniform and (b) adversarial traffic patterns.

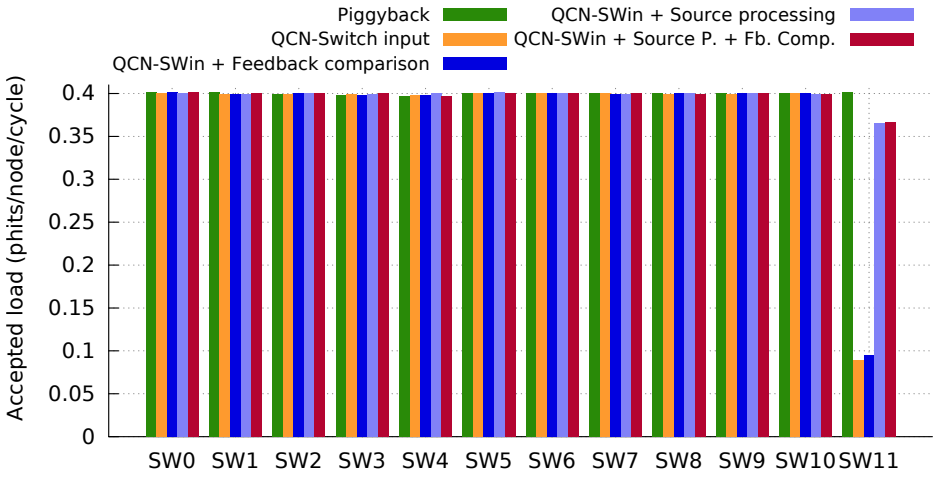


Figure 7. Throughput injected in each switch of group 0 of the proposed mechanisms with input-port sampling under ADV traffic with load 0.4 phits/node/cycle.

5.1.1. QCN-switch with input-buffer sampling. Figure 6 shows results when QCN-switch implements occupancy sampling at the input buffers. Under uniform traffic in Figure 6a, all the QCN-based mechanisms obtain optimal latency, similar to the reference *MIN*. *Piggyback* sends part

DOI: 10.1002/cpe.4440

of the traffic non-minimally, which slightly increases its latency, as observed in the inset plot. At high loads, both *QCN-Switch input* and *QCN-SWin + Source processing* decrease their minimal probability values after receiving CNMs, diverting more traffic towards non-minimal paths. Such strategy does not reduce overall congestion; on the contrary, it causes throughput at saturation to take a nosedive. However, *QCN-SWin + Feedback comparison* corrects this problem and presents good throughput at saturation; while there is a slight drop, it is relatively small and obtained throughput is competitive with the adaptive reference *Piggyback*. On the other hand, when *feedback comparison* is combined with *source processing*, the throughput loss at saturation reappears.

In the context of ADV traffic in Figure 6b, our base proposal *QCN-Switch input* and *QCN-SWin + Feedback comparison* have a large latency at low load because most switches are not aware of the congestion caused by the adversarial pattern; indeed, all hosts connected to the respective switch S_{out} of each group try to send all traffic following a minimal path. *QCN-SWin + Source processing*, alone or combined with *feedback comparison* eliminates this rise on packet latency, since it allows all switches in a group to detect congestion and to deal with it by using the non-minimal paths. This effect is also visible in Figure 7 focusing on throughput fairness. Unfortunately, the latency of *QCN-SWin + Feedback comparison*, which was the only competitive one in terms on throughput under even loads (see Figure 6a), is worse than *QCN-Switch*. The comparison with the average feedback value Fb_{avg} makes minimal routing more frequent, which increases average latency under adversarial traffic.

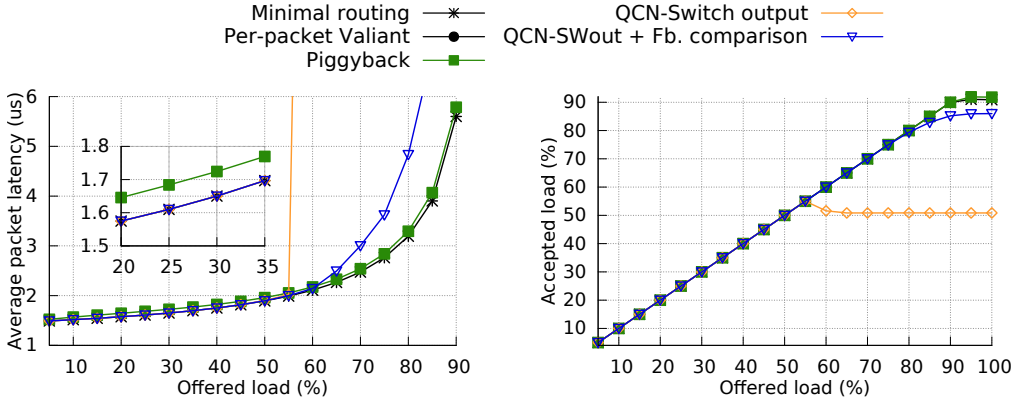
Throughput under adversarial traffic are consistent with the latency values analyzed above. *QCN-SWin + Source processing*, alone or with *feedback comparison* obtains the maximum saturation throughput, and despite suffering some congestion at high loads, its throughput is competitive with the oblivious *Valiant*. Note that both *QCN-Switch input* and *QCN-SWin + Feedback comparison* have a minor loss of throughput, around 2%, which occurs below the saturation point. This is observed easily in the slope of their throughput curves before saturation, lower than the expected 45º and this is indicative of unfairness issues. Figure 7 explores this problem in more detail, comparing the throughput obtained by each switch of Group 0 at load 0.4 phits/node/cycle. Switch SW11 corresponds to the switch S_{out} in Figure 1. Using *QCN-Switch input*, SW11 injects significantly less traffic than the other switches of the group. As discussed in Section 3, SW11 does not receive a significant amount of congestion notification messages from the global link, because all traffic received in the destination group is quickly dispatched to its destination switch, so the queues do not get full; recall that during the current evaluation subsection we implement the detection point at the input buffers. As expected, this unfairness is resolved when *source processing* is used, as SW11 sends to itself the same notification that would be sent to other switch when congestion is detected, which means SW11 also changes its minimal probability for congested ports.

5.1.2. QCN-switch with output-buffer sampling. Figure 8 shows average packet latency and throughput under *uniform* and *adversarial* traffic patterns using occupancy sampling at the output queues, versus the reference routing schemes. Note the *source-processing* variant is not necessary because the base case is already fair when using output-port sampling.

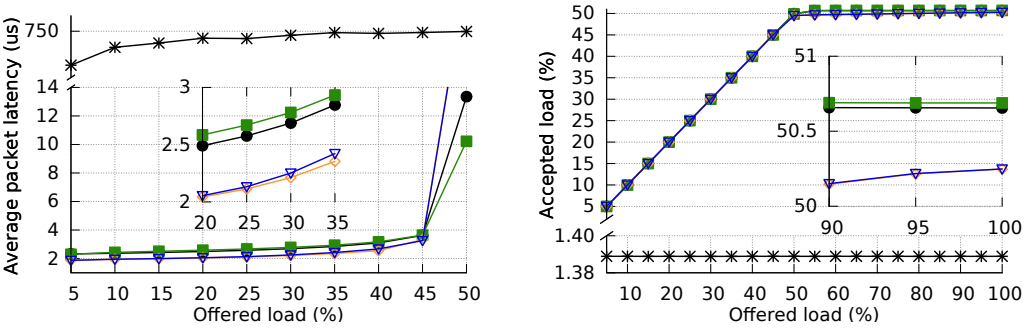
Under uniform traffic, *QCN-Switch output* suffers a lost of peak throughput which is due to the "positive control loop" described in Section 3.2, which starts when congestion is detected and results on most traffic using non-minimally paths. However, *QCN-SWout + Feedback comparison* corrects this problem and presents good throughput at saturation, which is competitive with the adaptive reference *Piggyback*. At medium loads, *Piggyback* sends part of the traffic non-minimally, which increases latency as shown in the inset plot; by contrast, the two QCN-based mechanisms obtain optimal latency, similar to the reference *MIN*.

In the context of ADV traffic, our base proposal *QCN-Switch output* reduces network latency over Piggyback reference because it eliminates the local hop on the intermediate group. Latency results are also better than when using input-port sampling in Figure 6b, because congestion detection is more fair in this case. Throughput results under adversarial traffic are consistent with the latency values analyzed above, resulting in almost ideal throughput just below the reference mechanisms.

DOI: 10.1002/cpe.4440



(a) Average Latency and Throughput under uniform (UN) traffic



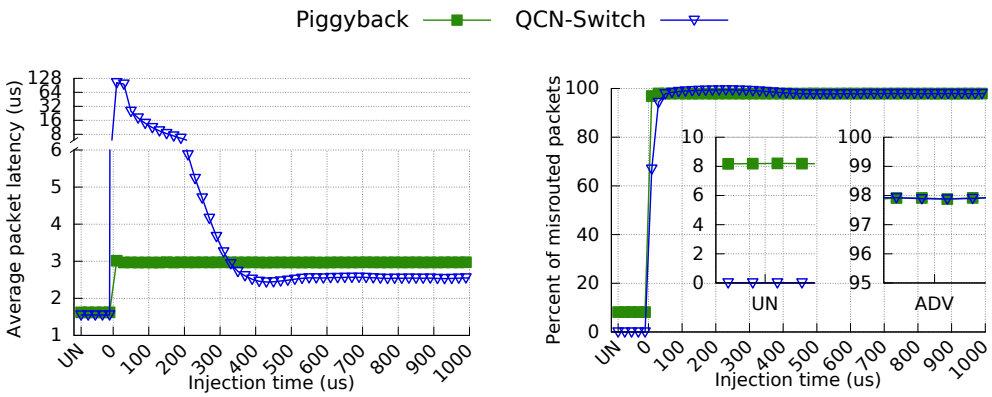
(b) Average Latency and Throughput under adversarial (ADV) traffic.

Figure 8. Average latency and throughput of the proposed mechanisms with output-buffer sampling under uniform (a) and adversarial (b) traffic patterns.

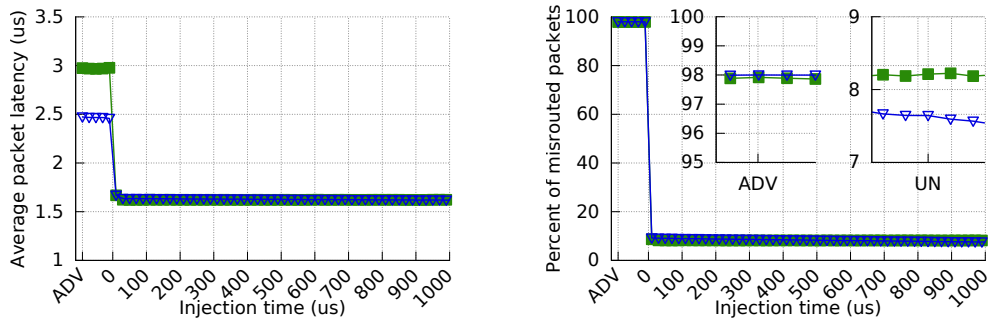
Figure 9. Throughput injected in each switch of group 0 of the proposed mechanisms with output-buffer sampling under ADV traffic with load 0.4 phits/node/cycle.

Figure 9 presents fairness results with sampling at output queues. Output-port sampling can generate CNMs that are intercepted by any switch in the group, including the same switch that generates the message. For this reason, the base proposal is fair and there is no need to implement *source processing*. Adding *feedback comparison* to the base case does not impact on its fairness.

DOI: 10.1002/cpe.4440



(a) Average latency and percentage of misrouted packets during a traffic change from UN to ADV traffic.



(b) Average latency and percentage of misrouted packets during a traffic change from ADV to UN traffic.

Figure 10. Average packet latency and percentage of misrouted packets of Piggyback and the QCN-Switch with output-buffer sampling and *feedback-comparison* changing from an uniform (UN) to an adversarial (ADV) traffic pattern at injection time 0.

In conclusion, sampling at the output offers better and more consistent results than sampling at input queues, and using *feedback comparison* improves throughput. For this reason, the rest of the paper will focus only on that configuration. In upcoming sections all references to the mechanism are denoted simply *QCN-Switch* to refer to *QCN-Switch output + feedback comparison*.

5.2. Performance under transient loads.

This section evaluates the response time to traffic changes by modeling a network load of 0.4 phits/node/cycle that changes from *uniform* to *adversarial* pattern and vice versa. We warm-up the network for $2.4ms$ with the first traffic pattern. At that point the network is stable and then we change to the second traffic pattern (setting this point to be time $t = 0$) and run that second pattern for 100 μs. During this period, the evolution of the latency and misrouted packets are registered, showing how the network transitions from one pattern to the other.

Figure 10 presents the average packet latency and the amount of misrouted packets for both transient loads, UN to ADV and vice versa. The latency of our proposal under *adversarial* is lower than Piggyback; this result is consistent with Figure 8. The response time from UN to ADV is almost $0.4ms$, while from *adversarial* to *uniform* it is much quicker (less than $0.005ms$). This change is faster because UN uses all ports evenly and there is only one rule on each switch (the one associated with the global link) that needs updating to use minimal paths. In other words, sending a small amount of traffic non-minimally under a *uniform* traffic has minimal impact on performance. On the other hand, when we change the traffic from UN to ADV, all the traffic in a switch is funneled towards the same global link until those rules are changed to use non-minimal paths, creating a temporary hot-spot that requires some time to be absorbed by the adaptive network.

DOI: 10.1002/cpe.4440

Comparing the percentage of misrouted packets between Figures 10a and 10b under the *uniform* traffic phase, we notice they are not the same; in the first plot the switch has not detected any congestion during the *uniform* phase, so there is 0% misrouting; on the second plot the percentage of traffic sent minimally is still trending down after the previous *adversarial* phase, and although it initially drops very quickly from 100% to 8%, it progresses very slowly towards 0 afterwards.

5.3. Sensitivity analysis

This subsection studies the impact of the different parameters used in our adaptive routing mechanism, both from the QCN notification part and our implementation in the receiving switches. We use both steady and transient traffic, changing one parameter at a time from the configuration employed in previous evaluations. The QCN-Switch evaluated from this point forward employs the *QCN-Switch + feedback comparison* model, which represents the best configuration as shown in previous subsections. Finally, we evaluate the selected parameter set for different network sizes.

5.3.1. Number of notifications: CPC and $\%CNMs$. The *Congestion Point Cycle (CPC)* parameter represents the frequency of buffer sampling in QCN. In this section we consider values for this parameter from 1 sample every 100 packets (default) to 1 sample every 25 packets. The parameter PC_l, which defines the amount of packets sent following a given routing table entry before its output port percentage is increased, is set accordingly.

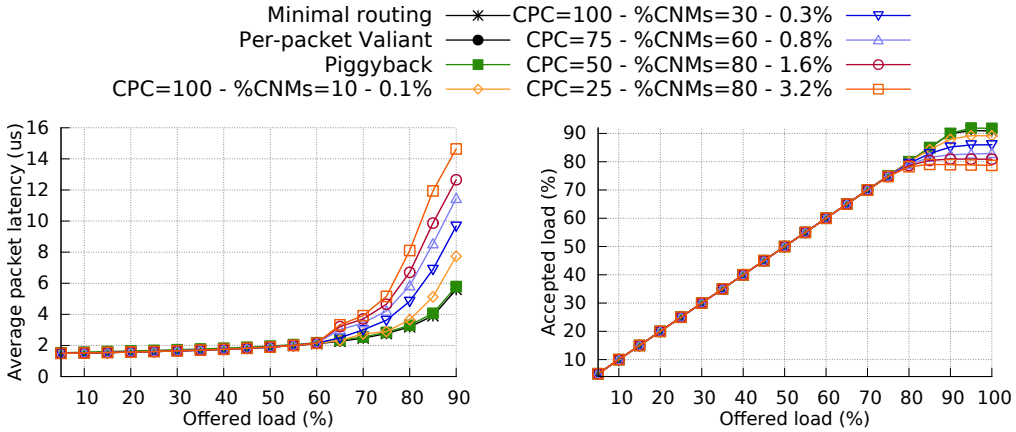
Similarly, $\%CNMs$ is a QCN parameter that indicates how many CNMs are generated from buffer sampling operations with a negative (congested) result. By default, $\%CNMs$ is 10% in QCN, this is, only one message is sent every ten sampling operations which report a negative result. However, evaluations in previous sections employ $\%CNMs = 30\%$, and in this section we consider values ranging from 10% to 80%. Combining *CPC* sampling frequency and $\%CNMs$, we evaluate a set of configurations that generates from 1 to 32 CNMs per 1000 packets forwarded, denoted 0.1% to 3.2% in the legend.

Figure 11 presents average latency and throughput under UN and ADV traffic. Figure 11a shows a trade-off between the amount of messages generated and the latency and throughput at high uniform loads. The most aggressive configuration ($CPC = 25$ and $\%CNMs = 80\% - 3.2\%$), and similar ones, suffer at high loads because the overhead of the CNMs causes some non-minimal forwarding which increases latency. In general, increasing the amount of CNMs has a slight impact on saturation throughput under both traffic patterns. Besides, it increases latency at high loads under *uniform* traffic. The default configuration for QCN ($CPC = 100$ and $\%CNMs = 10\% - 0.1\%$) is too low for our proposed routing, as it produces high latency at medium loads under ADV traffic as seen in Figure 11b. Overall, our selected configuration ($CPC = 100$ and $\%CNMs = 30$) produces competitive performance under UN and ADV traffic, both in terms of low latency at medium loads and peak throughput.

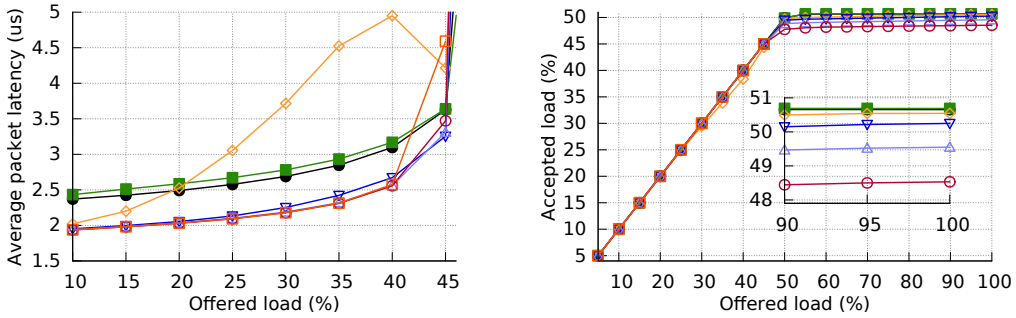
Figure 12a shows the evolution of both average packet latency (left plot) and percentage of misrouted packets (right plot) when the traffic pattern changes from UN to ADV. As expected, increasing the frequency of Congestion Notification Messages has the greatest impact on the response time to traffic changes: the more CNMs received, the sooner the switches are aware of the changes on the network load. We consider the response time as the time spent from the traffic change (at time $t = 0$) to the moment in which average latency approximately converges to the new value. Using the default parameters from QCN presents a large response time, close to 1 ms. Our base configuration presented in Table I offers $0.4ms$, and increasing the amount of messages continues decreasing the response time up to near $0.1ms$. The CNM frequency has no impact on the percentage of misrouted packets, except a minor effect at the start of the transition phase.

Finally, Figure 12b shows the evolution of the minimal probability of the port connected to S_{out} (following the notation in Figure 1) of one switch in the middle of group 0, during the traffic change from UN to ADV. For each configuration, one marker is presented in each instant in which the probability value is updated. Note that only one simulation is analyzed for each configuration, without averaging several executions as in other figures. All configurations reduce the minimal probability rather quickly after the traffic change, and it remains relatively low, under 10%. However,

DOI: 10.1002/cpe.4440



(a) Average Latency and Throughput under uniform (UN) traffic



(b) Average Latency and Throughput under adversarial (ADV) traffic.

Figure 11. Average latency and throughput of the QCN-Switch with output-buffer sampling and *feedback comparison*, for different values of sampling interval and percentage of CNMs sent under uniform (a) and adversarial (b) traffic patterns.

there is a significant variability in the probability values. The evolution of the minimal probability follows a sawtooth pattern caused by the AIMD policy used. The increase stairs are steeper in the configurations with more notifications (such as the square orange marker - 3.2%, whose evolution is highlighted) because a more aggressive CPC value also implies a shorter interval PC_l between probability increases. However, the results in Figure 12b do not present any clear conclusive difference (as is observed later in the similar plot in Figure 14a for the parameter L_f).

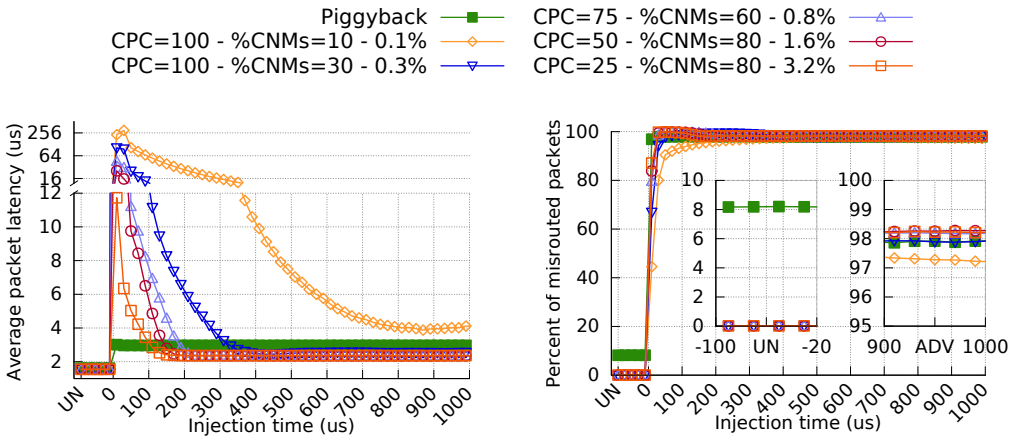
In conclusion, the selection of the parameters CPC and $\%CNMs$ presents a trade-off between latency under high *uniform* traffic loads and reaction time needed to adapt to traffic changes. The default value used in Table I presents a competitive latency and a sufficiently quick response time, according to our previous analysis of HPC applications requirements in [7].

5.3.2. Reduction factor L_f. L_f determines how strongly the switch reacts when receiving CNM by decreasing the probability of sending traffic minimally. Figure 13 presents steady-state results under UN and ADV traffic for values of $L_f = \{1/256, 1/128, 1/85, 1/64\}$, which implies approximately a maximum probability reduction of 0.25, 0.5, 0.75 and 1.

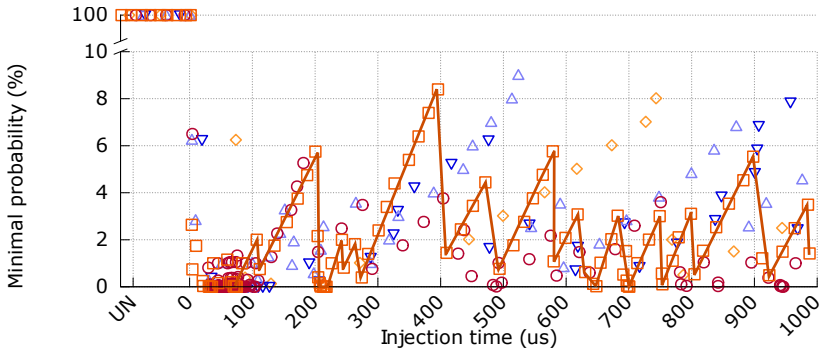
The default value in Table I, $L_f = 1/64$, is the most aggressive one and it produces the lowest latency under adversarial traffic. However, other values present better latency under uniform traffic at high loads. The impact on throughput is very small in both cases.

Figure 14 presents results under transient traffic. Response time is affected considerably by the value of L_f. The base value $L_f = 1/64$ corresponds to a maximum reduction factor of $R = 1 - 63/64$, this is, it can reduce the port percentage maximally after receiving a single CNM. Larger values of L_f present a slower convergence time, which is particularly obvious in the average latency

DOI: 10.1002/cpe.4440



(a) Average latency (left) and percentage of misrouted packets (right).



(b) Evolution of the percentage of minimal routing in an intermediate router in group 0, measured in the congested port that connects minimally to the destination in ADV. Each marker represents one change in the probability.

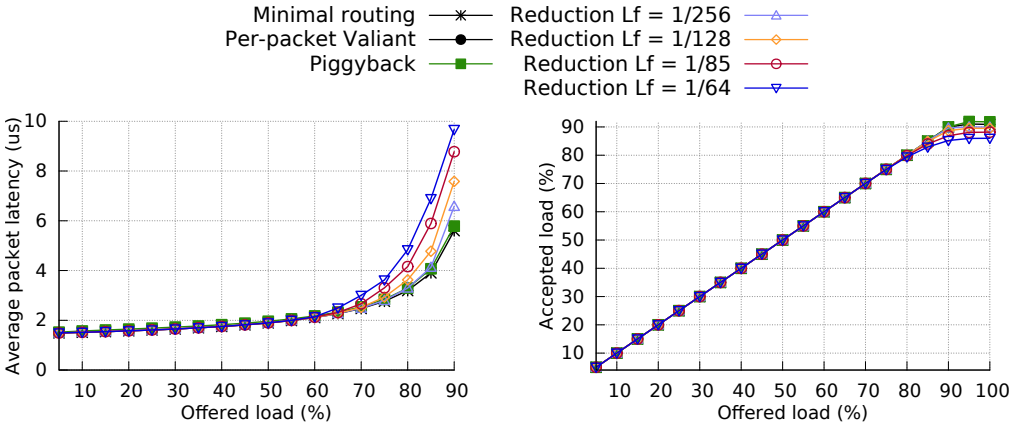
Figure 12. Transient response when traffic changes from UN to ADV, for different values of sampling interval and percentage of CNMs.

in Figure 14a and port probability in Figure 14b. In particular, $L_f = 1/128$, which corresponds to a maximum reduction value of $R \approx 50\%$, is often employed in control mechanisms based on an AIMD policy (such as TCP or QCN). It can be observed that using such value causes significantly slower convergence: it requires about six probability updates and more than twice the time as the base configuration to converge.

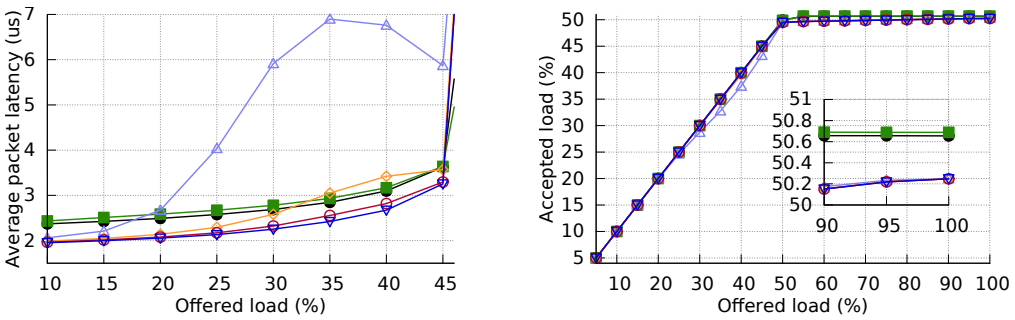
5.3.3. Probability increase PI. The parameter PI determines the increase of the minimal port probability, when no congestion notifications are received during an interval of $PC_l = 100$ frames. Figure 15 presents results for steady-state traffic for different values of PI. With low values for PI (0.1% or 0.5%), throughput and latency results under uniform traffic at high loads, presented in Figure 15a, are poor because the system is slow to reduce non-minimal routing after spurious CNMs are received. By contrast, high values of PI increase latency under adversarial traffic in Figure 15b, because a brief absence of notifications causes the system to revert to minimal routing too soon.

Obviously, this parameter also determines the convergence time when transitioning from ADV to UN traffic. However, as discussed in Section 4.1, this transition time is not as critical as the change from ADV to UN, and Figure 10b already presented an acceptable transient result. Considering this reason, transient results are omitted for brevity, but it is clear that a higher PI would converge to 0% misrouted packets faster.

DOI: 10.1002/cpe.4440



(a) Average Latency and Throughput under uniform (UN) traffic



(b) Average Latency and Throughput under adversarial (ADV) traffic

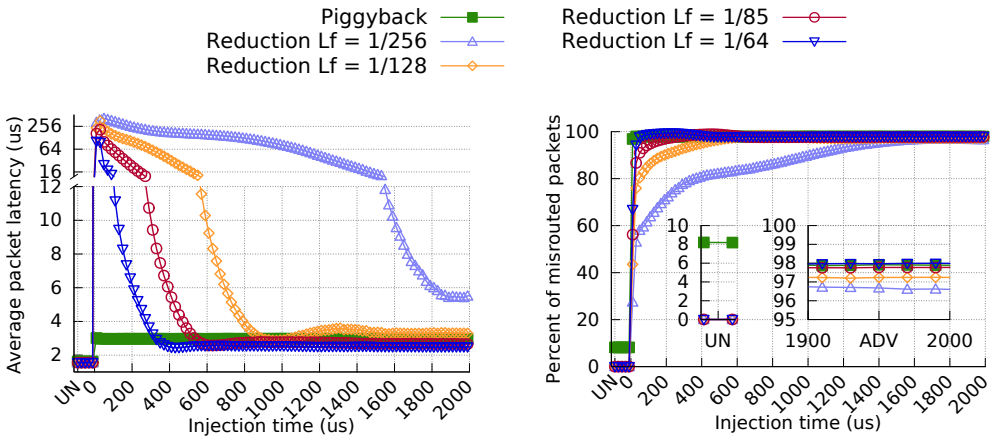
Figure 13. Throughput and latency for different limiting factors L_f under steady traffic.

5.3.4. Feedback comparison thresholds Th_1 and Th_2. Section 3.2 introduces the use of two thresholds to avoid excessive variations in the minimal percentage of the ports. Threshold Th_1 controls the range of feedback values with which the probability is increased when no CNM is received; preliminary evaluations show that it has a negligible impact on performance. For this reason, this section presents an evaluation of the impact of Threshold Th_2, which restricts the reduction of the minimal probabilities of each port. This threshold represents the difference required between the feedback value received in a CNM, which tops at 63, and the average Fb_{avg}, in order to reduce the minimal port probability. Figure 16 presents an evaluation of the impact of this threshold on steady load, for different values of Th_2 from 0 to 50.

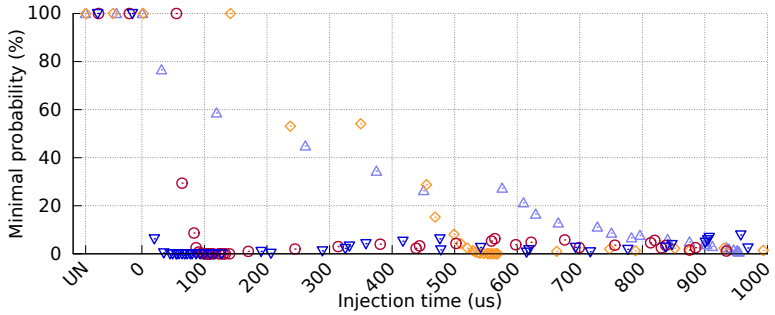
Figure 16a presents average latency and throughput under uniform traffic. A small to medium value Th_2 allows the switch to ignore transient congestion fluctuations. A large value means that only CNMs that report an abrupt change will cause a reduction of the port probability, which is good for uniform traffic. In fact, not using a threshold (i.e., $Th_2 = 0$) produces poor latency and throughput. However, a very high value is bad under adversarial traffic presented in Figure 16b because probability is not properly reduced and throughput goes down. Thus, using the intermediate value of $Th_2 = 30$ is the best option for a network that may support different traffic loads.

5.3.5. Network size. Previous sections have evaluated the parameter selection using a network size of 5256 hosts, which corresponds to a Dragonfly network using $h = 6$ global links per switches and 23 ports used per switch. Figure 17 shows the results for different network sizes, ranging from 1056 ($h = 4$, 15 ports used per switch) to 16512 endpoints ($h = 8$, 31 ports used per switch) under uniform and adversarial traffic patterns. Evaluations show maximum throughput in saturation, and latency at intermediate loads. Except for the aforementioned network size, all the evaluations use the

DOI: 10.1002/cpe.4440



(a) Average latency (left) and percentage of misrouted packets (right).



(b) Evolution of the percentage of minimal routing in an intermediate router in group 0, measured in the congested port that connects minimally to the destination in ADV. Each marker represents one change in the probability.

Figure 14. Transient response when traffic changes from UN to ADV, for different values of the reduction limiting factor, L_f.

same set of parameters presented in Table I; *QCN-Switch* parameters are not modified for different network sizes. Oblivious routing (Minimal and Valiant) is also included as a reference.

Saturation throughput remains fairly stable for different network sizes. Under ADV traffic *QCN-Switch* saturation throughput is in all cases virtually identical to Valiant. Under UN traffic, it is close to the reference MIN, and it is better for larger network sizes. Average latency slightly grows with the network size in both oblivious references. A similar effect occurs for *QCN-Switch*, but latency increases slightly more under adversarial traffic.

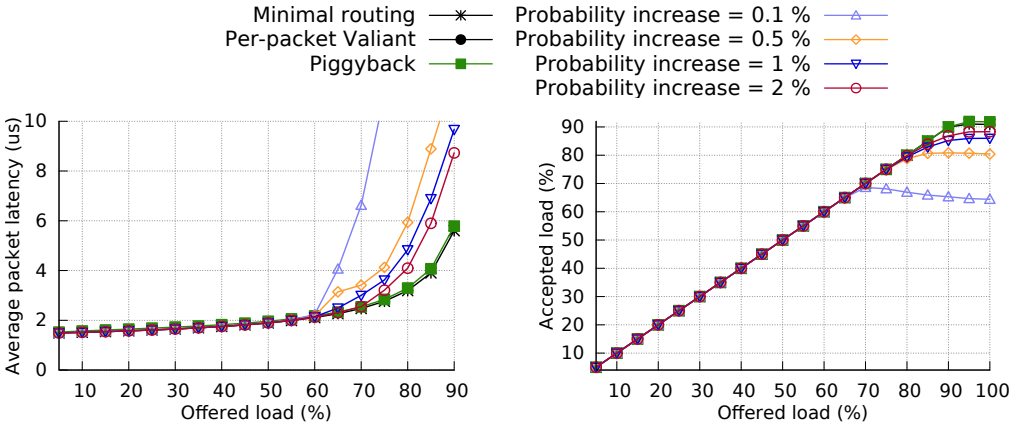
These results suggest that the selected parameters are valid for a broad range of network sizes, but they accept a fine tuning for distant configurations.

5.4. Alternative traffic patterns

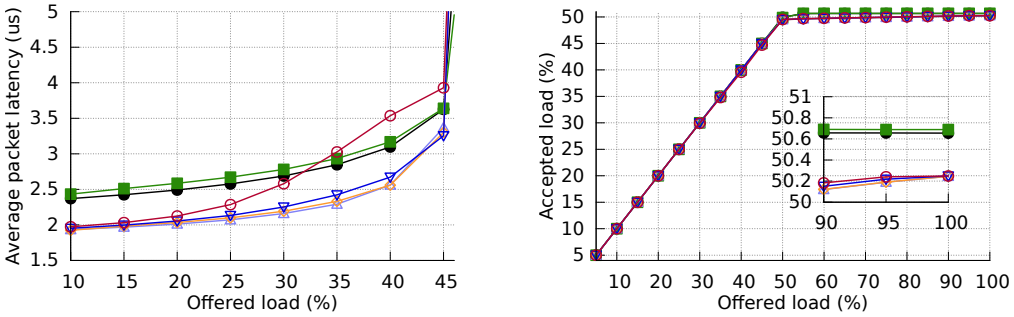
This section presents performance of the proposed mechanism under the two alternative traffic patterns introduced in Section 4.3: Hot-region and Random Permutation. Figure 18 presents steady-state results for them, and shows results of *QCN-Switch* with and without *feedback comparison*.

The latency and peak throughput of adaptive routing mechanisms (PB and both QCN-Switch implementations) under Hot-region in Figure 18a is close to the optimal result of MIN routing. However, Hot-region suffers from endpoint congestion which, without being accompanied with any additional congestion management, spreads through the network [25] decreasing throughput after saturation. The current design of *QCN-Switch* does not explicitly handle endpoint congestion, and the throughput drop is very significant, around 18%. This penalty is very small (around 1-2%) in other non-minimal mechanisms (PB and VAL), possibly because of their faster adaptation time.

DOI: 10.1002/cpe.4440



(a) Average Latency and Throughput under uniform (UN) traffic



(b) Average Latency and Throughput under adversarial (ADV) traffic

Figure 15. Throughput and latency for different values of probability increase PI under steady traffic.

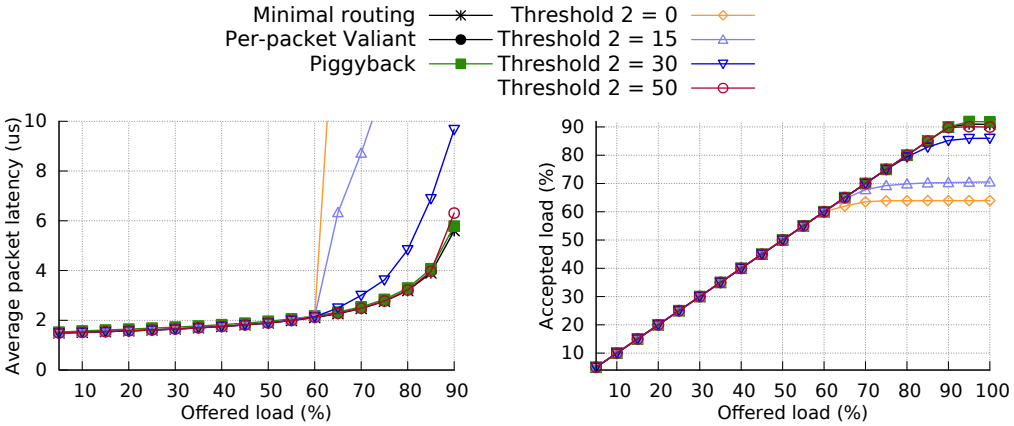
Interestingly, the performance drop of the base *QCN-Switch* without *feedback comparison* is also very small. A study of *QCN-Switch* dealing with endpoint congestion is left for future work.

Figure 18b shows results with random permutations. Interestingly, latency results of both MIN and VAL routing saturate at lower loads than adaptive routing. *QCN-Switch* presents a competitive result, but PB performs significantly better. Maximum throughput of *QCN-Switch* is lower than PB, and MIN also gets higher results.

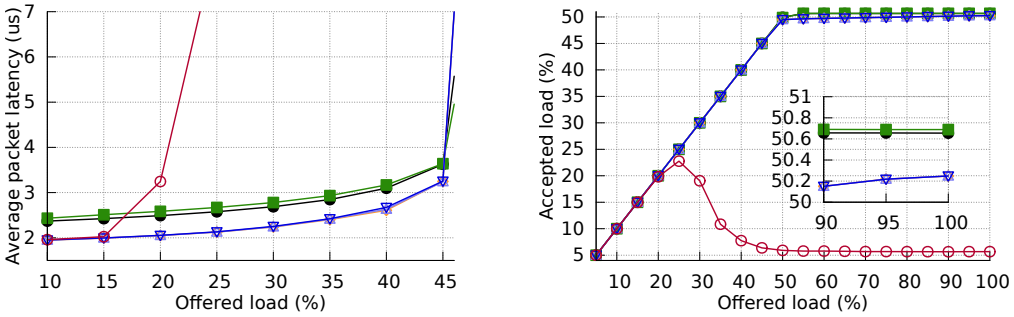
6. RELATED WORK

Explicit congestion notification has been used in commodity networks for a long time. IP has ECN bits for explicit congestion notification [12]. These bits are set when switch queues exceed a given threshold, possibly following a *marking* policy such as RED [26]. TCP may react to packets with the ECN source flag set by decreasing its congestion window, which throttles injection. A similar approach with a single-bit forward notification is used in Infiniband [13], Omnipath [27] and RoCE [28], the latter being encapsulated over UDP/IP. When packets are received with the congestion flag active, the destination generates a response notifying the presence of congestion in the forward path. QCN [8] generates backwards congestion notification messages at layer 2, which include a multi-bit feedback value computed from both the current queue occupancy and from the increase or decrease rate. Datacenter TCP [29] estimates not only the presence (single-bit) of congestion but also its *amount*, based on the count of ECN flags measured during a given interval (typically, a TCP RTT).

DOI: 10.1002/cpe.4440



(a) Average Latency and Throughput under uniform (UN) traffic



(b) Average Latency and Throughput under adversarial (ADV) traffic

Figure 16. Throughput and latency for different feedback-comparison threshold Th_2.

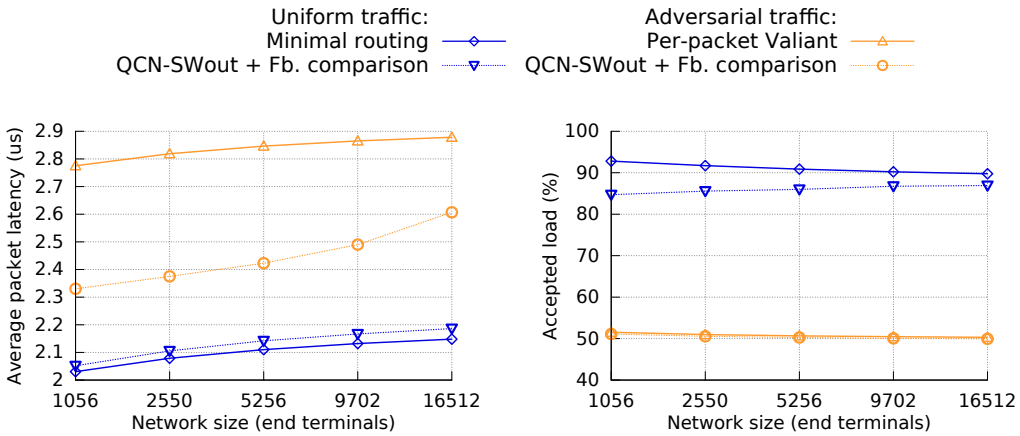
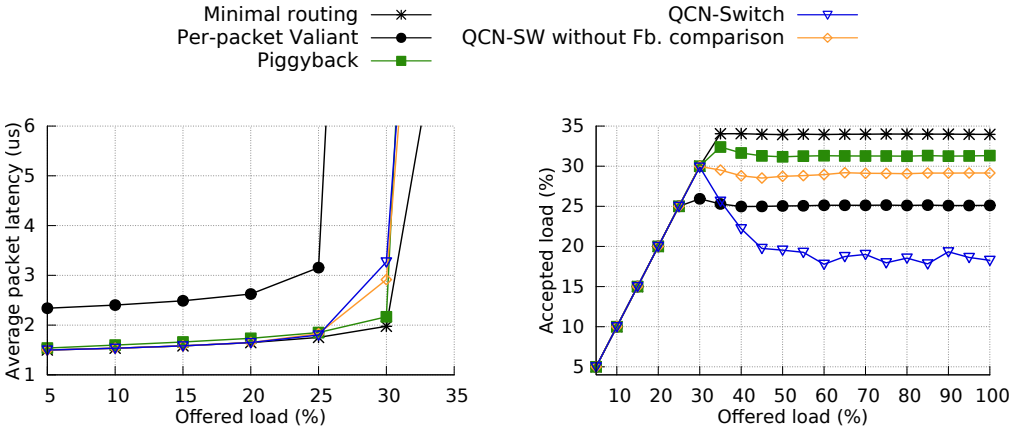


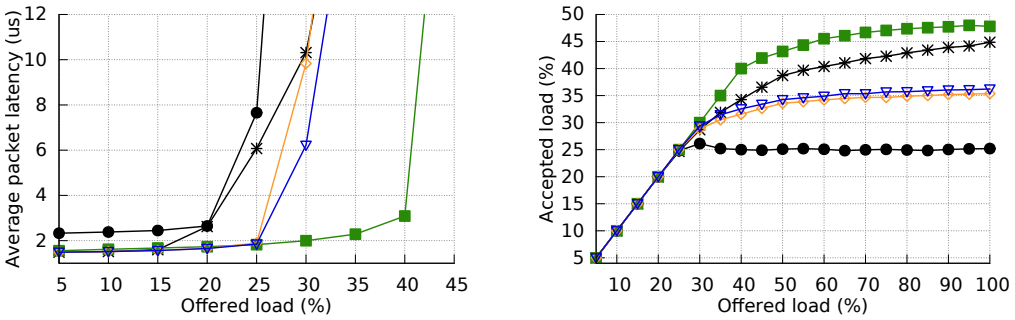
Figure 17. Latency and throughput for different network sizes, under uniform (blue) and adversarial (yellow) traffic. Average packet latency is obtained under 65% (UN) or 35% (ADV) load and throughput is obtained under 100% of load. Results for oblivious routing (MIN and VAL) are also presented as a reference.

Section 3 has already mentioned many proposals for load balancing [1, 5, 6, 15, 16, 17, 18, 19]. In the context of commodity Datacenter networks used in our evaluations, adaptive routing is supported in multiple technologies, at layer-3 such as IP or layer-2 such as TRILL [30], typically using Equal-Cost MultiPath (ECMP). In such mechanisms, forwarding tables include multiple matching entries for a given destination, and the selection process is typically performed using a hash of some fields (addresses) of the packet. While this guarantees that all packets from the same flow follow the same

DOI: 10.1002/cpe.4440



(a) Average Latency and Throughput under hotregion traffic



(b) Average Latency and Throughput under random permutation traffic.

Figure 18. Average latency and throughput of the proposed mechanism and reference mechanisms under under (a) hotregion and (b) random permutation traffic patterns.

path, it does not consider non-minimal adaptive routing nor is modulated by the congestion level. Minkenberg *et al.* [31] introduced the use of explicit congestion notifications to adapt traffic in Datacenters. Their proposal differs from our approach in two fundamental aspects. First, they do not consider non-minimal routing, with the increased load introduced by non-minimal paths and the associated positive feedback loop. Second, they do not consider a probability for each available path, nor a recovery mechanism to restore minimal routing when congestion disappears. Instead, they consider fixed time intervals, and routing information is reset on each interval by discarding its current status and reverting to minimal traffic.

The mechanisms proposed compare congestion on different paths and modify the probabilities of minimal paths in the routing table. UGAL [32] selects between minimal or Valiant routing on a packet per packet way, but it relies on flow-control credit counters and requires global information. More elaborate mechanisms improve this non-minimal routing decision, both at the source or in-transit [5, 6]. Piggyback [5] and the *Averaging* proposal in [20] consider the average occupancy (or credit count) of all the ports in a switch, somehow similar to our *feedback comparison* variant. Contention counters [33] support non-minimal routing using only local traffic demand information (the output ports that would be used under minimal routing), with early detection and almost-immediate reaction time to traffic changes. The ECtN routing based on contention counters distributes explicit messages with port contention information (traffic demand for each global port in a Dragonfly topology), resembling ECN. Unlike ECN, ECtN does not measure buffer occupancy nor reacts to congested buffers: it proactively selects non-minimal routing to avoid the potential bottlenecks that would be caused by the estimated traffic pattern under minimal routing, instead of reacting to congestion after it occurs. Unfortunately, unlike the proposal in this paper, contention

DOI: 10.1002/cpe.4440

counters are not amenable to a straightforward implementation in commodity switches. Finally, our proposal implements source-adaptive routing using a pre-calculated non-minimal intermediate destination per source node; an extension to in-transit adaptive routing is part of future work.

The policies described in Section 3 replace QCN injection throttling at the NICs with adaptive routing in source switches, based on in-network congestion information sensed at in-transit buffers. Endpoint congestion would still need to throttle injection, but it is not evaluated in this work. Injection throttling might be handled using the original QCN implementation or any other ECN mechanism, by identifying CNMs generated by endpoints and not intercepting them in the network switches. Alternative implementations may rely on congestion avoidance at the transport level (such as TCP congestion control) or on the use of proactive reservations to avoid saturating the network, such as SRP [34], CRP [35], SMSRP and LHRP [36].

7. CONCLUSIONS

Most adaptive routing mechanisms proposed for low-diameter networks rely on local congestion information such as credits to decide between using minimal and non-minimal paths to forward each new packet. If the network switch does not support a credits mechanism, we could either blindly choose random minimal paths such as Valiant routing or make use any other available network status information to best balance the traffic load.

This work has explored non-minimal routing mechanisms based on explicit congestion notifications. The proposed *QCN-Switch* uses statistical routing driven by intercepting explicit congestion notification messages generated using commodity QCN.

Preliminary *QCN-Switch* design exploration shows it is best to rely on output-port sampling compared with input-port sampling, as it not only provides better performance but it is critical to provide throughput fairness under adversarial traffic. *Feedback comparison* is key to eliminate costly misrouting that occurs at saturated uniform loads, while still using non-minimal paths to reduce any other congestion caused by uneven loads.

The steady-state performance of the final *QCN-Switch* design is comparable to state-of-the-art sophisticated high-performance alternatives such as Piggyback. Considering transient changes of traffic, while the resulting design does not adapt routing as quickly as other mechanisms that rely on credit counters, it responds in a sub-millisecond time frame which is typically enough for most applications. A sensitivity analysis presents the trade-offs of the design, particularly improving performance for uniform or adversarial traffic patterns or trading off faster response time for lower saturation throughput. Evaluations with endpoint congestion show that QCN-switch suffers in such situation, and future work will focus on identifying and handling separately endpoint and network congestion.

In conclusion, we have explored the trade-offs of relying exclusively on ECN notifications for non-minimal adaptive routing, and provided a feasible switch design for low-diameter networks based on Ethernet.

ACKNOWLEDGEMENTS

We are grateful for the constructive comments received from the anonymous reviewers of the paper. This work has been supported by the Spanish Ministry of Education, Culture and Sports under grant FPU14/02253, by the Spanish Ministry of Economy, Industry and Competitiveness under contract TIN2016-76635-C2-2-R (AEI/FEDER, UE), by the European HiPEAC Network of Excellence and by the European Union's Horizon 2020 research and innovation program under grant agreements No. 671697 (Mont Blanc 3) and No. 671610 (EuroLab-4-HPC).

REFERENCES

1. Kim J, Dally WJ, Scott S, Abts D. Technology-driven, highly-scalable dragonfly topology. *International Symposium on Computer Architecture (ISCA)*, 2008; 77–88.

DOI: 10.1002/cpe.4440

2. Besta M, Hoefler T. SlimFly: A cost effective low-diameter network topology. *IEEE/ACM Intl. Conf. on High Performance Computing, Networking, Storage and Analysis (SC14)*, 2014.
3. Camarero C, Martinez C, Vallejo E, Beivide R. Projective networks: Topologies for large parallel computer systems. *IEEE Transactions on Parallel & Distributed Systems* 2017; **28**(7):2003–2016.
4. Valiant L. A scheme for fast parallel communication. *SIAM journal on computing* 1982; **11**:350.
5. Jiang N, Kim J, Dally WJ. Indirect adaptive routing on large scale interconnection networks. *Intl. Symp. on Computer Architecture (ISCA)*, 2009; 220–231.
6. García M, Vallejo E, Beivide R, Odriozola M, Valero M. Efficient routing mechanisms for dragonfly networks. *The 42nd International Conference on Parallel Processing (ICPP-42)*, 2013.
7. Benito M, Vallejo E, Beivide R. On the use of commodity Ethernet technology in exascale HPC systems. *IEEE 22nd International Conference on High Performance Computing (HiPC)*, 2015; 254–263.
8. IEEE standard for local and metropolitan area networks–virtual bridged local area networks - amendment: 10: Congestion notification, 802.1Qau 2010. URL http://www.ieee802.org/1/pages/802.1au.html,.
9. Benito M, Vallejo E, Beivide R, Izu C. Extending commodity OpenFlow switches for large-scale HPC deployments. *Proceedings of the 3rd IEEE International Workshop on High-Performance Interconnection Networks Towards the Exascale and Big-Data Era*, HiPINEB'17, IEEE Computer Society: Austin, Texas, USA, 2017.
10. Faanes G, Bataineh A, Roweth D, Court T, Froese E, Alverson B, Johnson T, Kopnick J, Higgins M, Reinhard J. Cray Cascade: a scalable HPC system based on a dragonfly network. *SC: Intl Conf on High Performance Computing, Networking, Storage and Analysis*, 2012; 103:1–103:9.
11. Arimilli B, Arimilli R, Chung V, Clark S, Denzel W, Drerup B, Hoefler T, Joyner J, Lewis J, Li J, *et al.*. The PERCS high-performance interconnect. *18th Symposium on High Performance Interconnects*, IEEE, 2010; 75–82.
12. Ramakrishnan K, Floyd S, Black D. RFC3168: The addition of explicit congestion notification (ECN) to IP 2001.
13. Pfister G, Gusat M, Denzel W, Craddock D, Ni N, Rooney W, Engbersen T, Luijten R, Krishnamurthy R, Duato J. Solving hot spot contention using InfiniBand architecture congestion control. *High Performance Interconnects for Distributed Computing Workshop (HPI-DC)*, 2005.
14. Neeser FD, Chrysos NI, Clauberg R, Crisan D, Gusat M, Minkenberg C, Valk KM, Basso C. Occupancy sampling for terabit CEE switches. *20th Annual Symposium on High-Performance Interconnects*, 2012; 64–71.
15. Kandula S, Katabi D, Sinha S, Berger A. Dynamic load balancing without packet reordering. *SIGCOMM Comput. Commun. Rev.* Mar 2007; **37**(2):51–62, doi:10.1145/1232919.1232925.
16. Al-Fares M, Radhakrishnan S, Raghavan B, Huang N, Vahdat A. Hedera: Dynamic flow scheduling for data center networks. *USENIX Conference on Networked Systems Design and Implementation (NSDI)*, 2010.
17. Mogul JC, Tourrilhes J, Yalagandula P, Sharma P, Curtis AR, Banerjee S. DevoFlow: Cost-effective flow management for high performance enterprise networks. *SIGCOMM Workshop on Hot Topics in Networks*, 2010.
18. Farrington N, Porter G, Radhakrishnan S, Bazzaz HH, Subramanya V, Fainman Y, Papen G, Vahdat A. Helios: A hybrid electrical/optical switch architecture for modular data centers. *ACM SIGCOMM Conference*, 2010; 339–350.
19. Rasley J, Stephens B, Dixon C, Rozner E, Felter W, Agarwal K, Carter J, Fonseca R. Planck: Millisecond-scale monitoring and control for commodity networks. *ACM Conference on SIGCOMM*, ACM, 2014; 407–418.
20. Won J, Kim G, Kim J, Jiang T, Parker M, Scott S. Overcoming far-end congestion in large-scale networks. *2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*, 2015; 415–427.
21. García PJ, Flich J, Duato J, Johnson I, Quiles FJ, Naven F. *Dynamic Evolution of Congestion Trees: Analysis and Impact on Switch Architecture*. Springer Berlin Heidelberg, 2005; 266–285.
22. García M, Fuentes P, Odriozola M, Benito M, Vallejo E, Beivide R. FOGSim interconnection network simulator 2014. URL http://fuentesp.github.io/fogsim/.
23. Alizadeh M, Atikoglu B, Kabbani A, Lakshmikantha A, Pan R, Prabhakar B, Seaman M. Data center transport mechanisms: Congestion control theory and ieee standardization. *2008 46th Annual Allerton Conference on Communication, Control, and Computing*, 2008; 1270–1277.
24. García-Haro J, Marín-Sillué R, Melús-Moreno JL. *ATMSWSIM An efficient, portable and expandable ATM SWitch SIMulator tool*. Springer Berlin Heidelberg: Berlin, Heidelberg, 1994; 193–212, doi:10.1007/3-540-58021-2_11.
25. Gran EG. Congestion management in lossless interconnection networks. phd, Faculty of Mathematics and Natural Sciences, University of Oslo March 2014.
26. Floyd S, Jacobson V. Random early detection gateways for congestion avoidance. *IEEE/ACM Trans. Netw.* Aug 1993; **1**(4):397–413.
27. Birrittella MS, Debbage M, Huggahalli R, Kunz J, Lovett T, Rimmer T, Underwood KD, Zak RC. Enabling scalable high-performance systems with the Intel Omni-Path architecture. *IEEE Micro* 2016; **36**(4):38–47.
28. Infiniband Trade Association. *Supplement to InfiniBand Architecture Specification Volume 1 Release 1.2.1. Annex A17: RoCEv2* 2014.
29. Alizadeh M, Greenberg A, Maltz DA, Padhye J, Patel P, Prabhakar B, Sengupta S, Sridharan M. Data center TCP (DCTCP). *ACM SIGCOMM Conference*, 2010; 63–74.
30. Zhai H, Hu F, Perlman R, 3rd DE, Stokes O. Transparent interconnection of lots of links (TRILL): End station address distribution information (ESADI) protocol (RFC 7357) Sept 2014.
31. Minkenberg C, Gusat M, Rodriguez G. Adaptive routing in data center bridges. *17th IEEE Symposium on High Performance Interconnects (HOTI)*, 2009; 33–41, doi:10.1109/HOTI.2009.14.
32. Singh A. Load-balanced routing in interconnection networks. PhD Thesis, Stanford University 2005.
33. Fuentes P, Vallejo E, Garcia M, Beivide R, Rodriguez G, Minkenberg C, Valero M. Contention-based nonminimal adaptive routing in high-radix networks. *Intl Parallel and Distributed Processing Symposium*, 2015; 103–112.
34. Jiang N, Becker DU, Michelogiannakis G, Dally WJ. Network congestion avoidance through speculative reservation. *IEEE International Symposium on High-Performance Computer Architecture*, 2012; 1–12.
35. Michelogiannakis G, Jiang N, Becker D, Dally WJ. Channel reservation protocol for over-subscribed channels and destinations. *SC'13: Intl Conf. High Performance Computing, Networking, Storage and Analysis*, 2013; 1–12.
36. Jiang N, Dennison L, Dally WJ. Network endpoint congestion control for fine-grained communication. *SC15: International Conference for High Performance Computing, Networking, Storage and Analysis*, 2015; 1–12.