Mechanism for the Deployment in Switched Ethernet Networks

Hela Mliki[1], Lamia Chaari Fourati[2], Lotfi Kamoun[1]
[1]Computer Science
National School of Engineers of Sfax
Sfax, Tunisia
[2]Computer Science
Higher Institute of Computer Science and Multimedia of Sfax
Sfax, Tunisia
mliki.hela@gmail.com, lotfi.kamoun@isec.rnu.tn, lamia.chaari@tunet.tn

ABSTRACT: *In order to define new switched Ethernet functions for data center use, the project IEEE802.1Qau [1] for congestion management is underway in the IEEE 802.1 standards committee. The aim of this project is to develop Ethernet congestion control algorithm for hardware implementation. Our focus is on Quantized Congestion Notification (QCN) algorithm, which is being considered by the IEEE802.1Qau standards for deployment in switched Ethernet networks.*

In this paper, based on simulation results, we show how QCN can dynamically control traffic flow, and we study QCN performances properties related to congestion control. In addition, we compare QCN behavior and performances to another backward congestion notification algorithm, which is Ethernet Congestion Management (ECM). The comparison between these two mechanisms is based on simulation results.

Keywords: Component, QCN, Congestion Notification, Metro_Ethernet

Received: 31 August 2011, Revised 1 November 2011, Accepted 7 November 2011

1. Introduction

The common response to congestion is to drop frames and the common method of avoiding the need to drop frame is to make in place very large switch buffer or to utilize backpressure mechanism [12], which is defined in the IEEE802.3x standard and allows pausing the link feeding the buffer. This type of mechanism leads to no packet losses but, potentially, spread congestion to upstream buffers and eventually to the sources. To mitigate this, it is desirable to notify sources in order to decrease thier-sending rate, thereby preventing buffer overflow. The Quantized Congestion Notification (QCN) mechanism is such a mechanism that has been developed to provide congestion control at layer 2. It has been retained for the IEEE 802.1Qau standard for deployment in switched Ethernet [11].

The QCN mechanism is a way for mitigating the congestion spreading effects due to link-level pause. During severe congestion it is necessary to ensure fast reduce of flow rate in source points. Then, when the congestion problem is over and an extra bandwidth is available, sources should be allowed to grab it, so for stable recovery of available bandwidth, we need a stable indication of "*available bandwidth*". However, we cannot increase transmission rate based on instantaneous value of positive feedback (or Qoff or Qdelta) as it is with ECM (Ethernet Congestion Management mechanism) [2], [3] because they will all swing from positive to negative as RTT (Round Trip delay Time) increases . To probe an available bandwidth, QCN follow the following steps [9]:

- Multiplicative Decrease.
- Fast Recovery (FR).
- Active Increase (AI).

The congested queue that causes generation of message notification is called congestion point. Sources that receive the congestion notification messages from congested point are called reaction points or rate limiters.

The main idea of QCN mechanism consists in sending periodic pings from rate limiter to probe for extra bandwidth. A switch which "*has no extra bandwidth*", responds indicating this; else, it does not respond. The rate limiter infers the availability of a bandwidth whenever a ping elicits no "*echo*".

In our simulation, we have considered congestion point as buffer shared between reaction points. We have not applied per priority pausing of traffic at congested links. In addition, we put into practice a burst traffic transported from sources to destination. The echoes of pings sent from rate limiters traverse the same path by the opposite direction from congested point to the sources.

The goal and the contribution of this paper are to describe QCN (Quantized Congestion Notification) algorithm, then based on simulations results QCN performances analysis arededuced. In addition, we propose comparative analysis between the Quantized Congestion Notification and the Ethernet Congestion Management mechanism basing on simulation results.

This paper is organized as follows: In the next section, we present a description of QCN algorithm. In the third section, we study a scenario of QCN mechanism to analyze its behavior. In section IV, we review performances of QCN mechanism via simulation analyses. In section V, we compare QCN with another backward congestion notification, which is Ethernet Congestion Management and the last section is to conclude this paper and to suggest some prospects.

2. QCN: Quantized Congestion Notification

QCN assumption consists in pushing congestion from the core of the network to the edge and use rate limiters at edge to shape flows causing congestion. Thus, QCN tries to eliminate congestion by reducing the sending rate at the sources.

It operates by monitoring switch queue length with respect to a predefined equilibrium threshold (Qeq). The Rate Limiter (RL) is named also the Reaction Point, which represents the network traffic machine senders. To probe an extra bandwidth, the rate limiter sends a periodic ping. The switch that did not have an extra bandwidth reply to indicate no available bandwidth otherwise it did not sends any reply. By the absence of a reply from the switch, the rate limiter deduces that the path have an available bandwidth, the RL deduces this for every ping that have no echo [7].

2.1 Rate adjustement in reaction point

As shown in Figure 1, rate limiter performs each time one of three states: Waiting to Probe (WP), Waiting for Echo (WE) and Short Fuse (SF). The Rate Limiter goes to WP state whenever it receives a negative feedback signal ($Fb < 0$). If the WP timer expires, the next packet sent by the Rate Limiter is a "*special packet*". A "*special packet*" is a data packet with one bit to indicate special. After launching the special packet, the rate limiter releases waiting for echo timer. If the rate limiter receives an echo for the special sent packet, the rate limiter comes back to the waiting for probe timer and continues executing Fast Recovery and Active Increase operations. If the waiting for echo timer expires and no echo is received, the rate limiter goes to short fuse and executes the Hyper Active Increase operations [8].

In addition to timers, QCN uses a byte counter to adjust throughput in reaction points.

2.1.1 Byte Counter

Byte counter is used to compute the number of cycles in Fast Recovery and in Active Increase. When the rate limiter receives a negative feedback, a timer is set off and a Fast Recovery is processed. The byte counter computes five Fast Recovery cycles where 150k byte is sent per cycle for example.

After that, an Active Increase cycle is performed where 75k byte have to be sent for example. Every time when there is no available bandwidth, sources perform Fast Recovery or Active Increase cycles; bring on an endless loop of FR and AI [8].

When negative feedback signals arrives:

During first cycle of FR, Current Rate (CR) goes down with every negative feedback signal and Target Rate (TR) remains unchanged.

For the next FR cycles:

$TR \leftarrow CR$ (just before ping) and $CR \leftarrow CR(1\text{-}Gd\,|\,Fb\,|)$.

At the end of each FR cycle:

$CR \leftarrow (CR + TR) / 2$; and TR does not change.

At the end of each AI cycle:

$TR \leftarrow TR+Ri$; $CR \leftarrow (CR + TR) / 2$.

At the end of each HAI cycle:

$TR \leftarrow Ri{*}\,Cycle\_Counter$; $CR \leftarrow (CR + TR) / 2$.

At the end of the first cycle of FR, if $TR > 10 * CR$ then

$TR \leftarrow TR / 8$; $CR \leftarrow (TR + CR) / 2$.

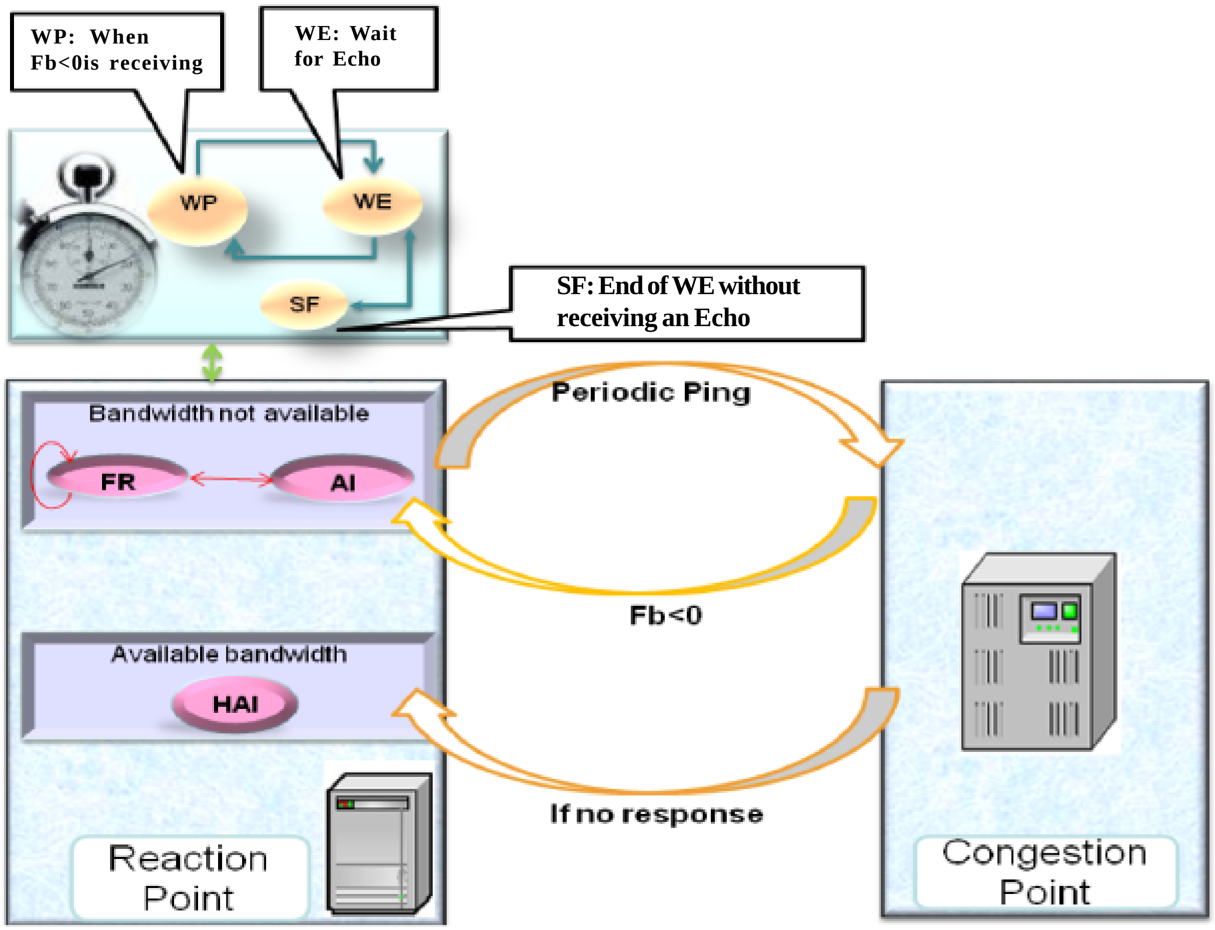

Figure 1. QCN mechanism

2.1.2 Congestion detection in congestion point

We suppose a timer, which marks out a non congestion period.

We assume the following parameters at the congestion point queue: Qeq = equilibrium threshold and Q_len = queue current length. If we suppose that Qeq = 6 packets, we pick up that bandwidth is available if queue length is lower than 6 packets. So when Q_len < Qeq it means that input rate is lower than output rate. Every time Q_len < 6 packets, congestion point starts congestion timer. If timer expires, bandwidth is available else timer is restarted when Q_len < 6 packets gain.

3. QCN Evaluation

In order to analyze performance QCN congestion control mechanism, our implementation was under the discrete event simulator OMNeT++ [4][5][6]. We have modelled the network as several users generating messages (Jobs) to a destination. These messages are queuing before achieving their destination. We have modeled flow as unidirectional transfer from the source to the destination, and a feedback pass over the network by the opposite direction on the same path. Furthermore, we assume a FIFO discipline in queuing network. We choose to delete the arriving packets when the queue is full. For all simulations, the link bandwidth is set to 10Mbits/s. Senders sent packets, which size are fixed at 1522 byte. Packets achieve their destination by passing through a switch queue with capacity equal to 100 packets. The simulation duration is set to 10 hours (36000 seconds). Several scenarios are used to carry Ethernet packets across network. For all simulations scenarios we have assumed that queue equilibrium threshold is set to 25 packets. Parameters used for simulations in reaction points are giving in Table I.

3.1 Network with twelve reaction points

As illustrated in Figure 2, we have supposed a network with one queue inside a congestion point, which is the switch, twelve Ethernet packets senders and a destination point. We assume two Fast Recovery cycles and two packets are sent per cycle. In addition, we plan out to send three packets in Active Increase cycle and ten packets in Hyper Active Increase cycle.

Simulation results are shown in Figure 3 and Figure 4. We note that the queue is often overloaded during simulation, as shown in Figure 3. It oscillates always over the queue equilibrium threshold and exceeds in many times the queue capacity, so that lead to packets drop. Consequently, rate limiter cannot proceed executing the Hyper Active Increase cycle as long as it receives negatives feedbacks as response to its specials packets already sent during Waiting for Echo timer (WE). Then the rate limiter processes a loop of Fast Recovery cycles and Active Increase cycle. This loop causes decreasing of throughout which stabilizes afterward around 1300 bit/sec as given in Figure 4. The Target throughput performs also decreasing while being upper than current throughput values. We point out the absence of Hyper Active Increase cycle, which could be processing after expire of Waiting for Echo timer. Since the queue swings always over the queue equilibrium threshold, there is always negatives feedbacks responses for pings sent by the rate limiter. Thus, we notice the absence of important increases in throughput curve.

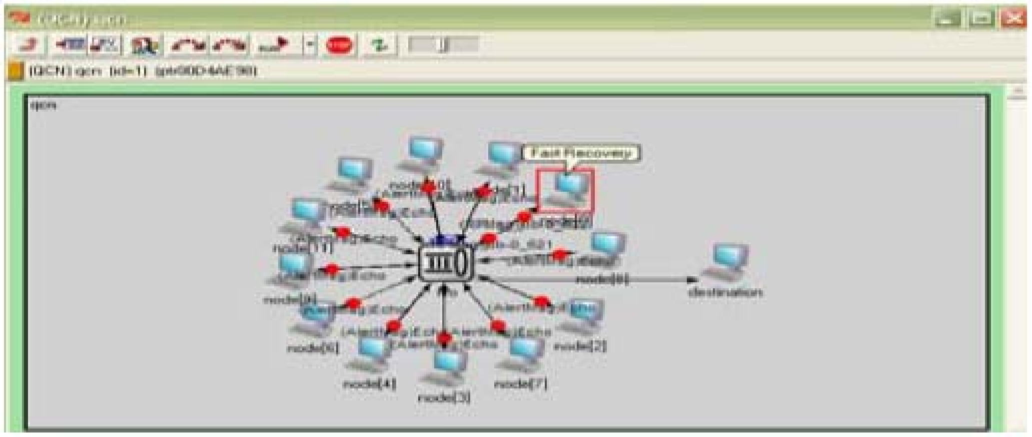

Figure 2. Network with twelve reaction points

4. QCN Performance Analysis

This section presents simulation results obtained for the stability, fairness and scalability features.

Parameters	Definition	Values
CR	Current throughput	106 bit/sec
TR	Target throughput	107 bit/sec
W	Weight of derivate component	2
alpha	Maximum throughput decrease	0.5
beta	Maximum throughout increase	0.5
Gd	Decrease gain	0.0078125
Gi	Increase gain	0.53333
Ri	Increasing throughput amount	1000 bit/sec
nb	CycleHAImax Cycle number of Hyper Active Increase	1 cycle
nb	CycleAImax Cycle number of Active Increase	1 cycle

Table 1. Rate Limiter parameters

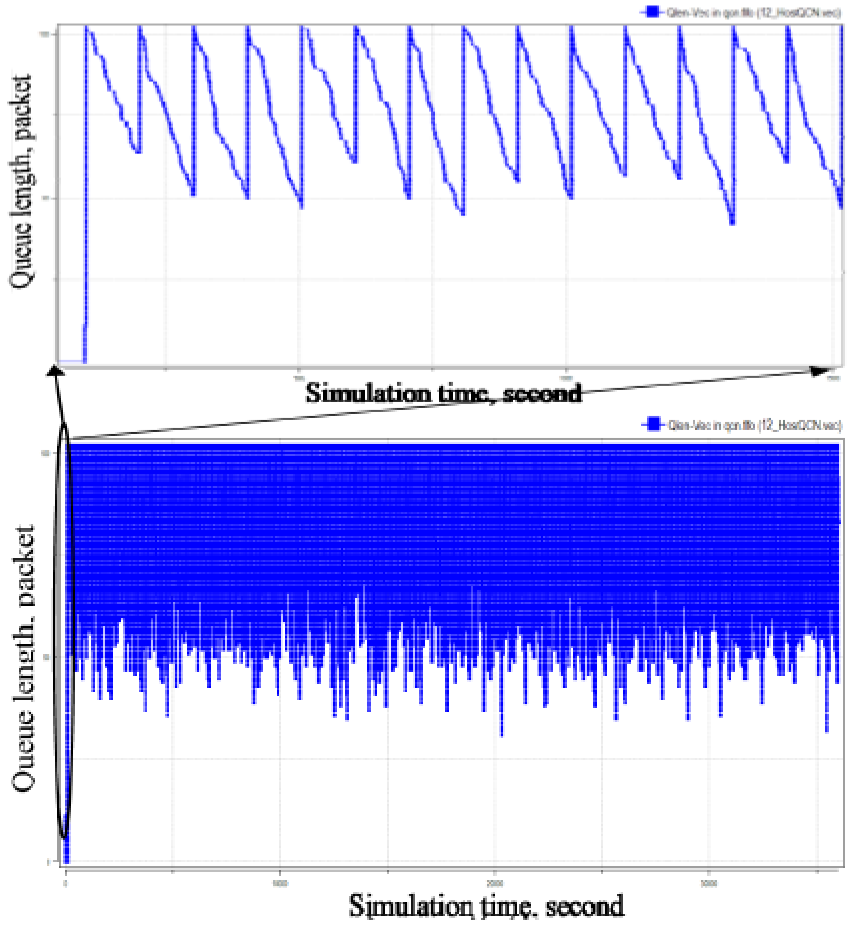

Figure 3. Queue length variation with QCN scheme

4.1 Stability Analyses

Stability represents the absence of queue fluctuation caused by under utilization or an overutilization of resource. This kind scenario leads to packets drop or bandwidth underuse.

When we raise the number of reaction points, queue length curve oscillates over the equilibrium threshold, and that bring to

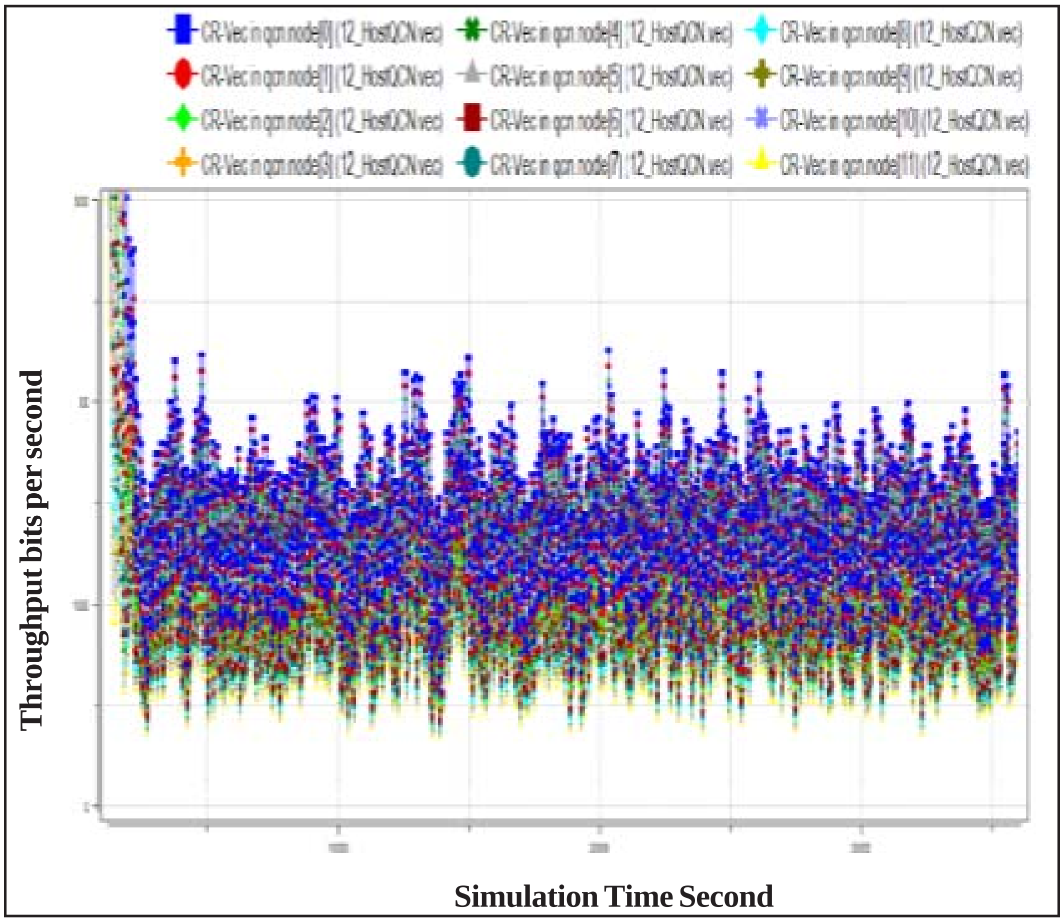

Figure 4. QCN throughput variation

packets drop. We note also that during simulation time the queue oscillation could not succeed to stabilize around equilibrium threshold as shown in Figure 3. Hence, we can deduce that stability is not well maintained with QCN mechanism.

4.1.2 Fairness Analyses

We can say that an algorithm is "fair" if it takes the completion times of similar flows equal Figure 5 below, shows the number of packets sent during simulation from different reactions points to the same destination.

Although increased number of traffic senders sharing the same path in different scenarios, we notice that numbers of packets sent by reaction points are fair. For example in a simulation with nine reaction points, all nodes achieve to send 2852 packets. In addition, we pick up from Figure 4 that throughput variation curves for different reaction points are superimposed, bring on to deduce that all reaction points participate in the network by sending traffic with the same throughputs. Thereby we point out that fairness with QCN mechanism is guaranteed.

4.1.3 Scalability Analyses

It indicates the ability of a mechanism to handle the increasing sum of flow in smooth way. We found that packets error rate for a scenario with one reaction point is equal to 0% and it is equal to 49% for a scenario with twelve reaction points. Figure 6 depict packets error rate for different scenarios. To compute the packets error rate we have used this formula:

Packet Error Rate = [1 - (Number of packets delivered to destination / total packets sent to destination)] * 100 (1)

By increasing number of reaction points, the curve oscillation of queue length gets close to queue maximum capacity bring on packets drop. As shown in Figure 4.

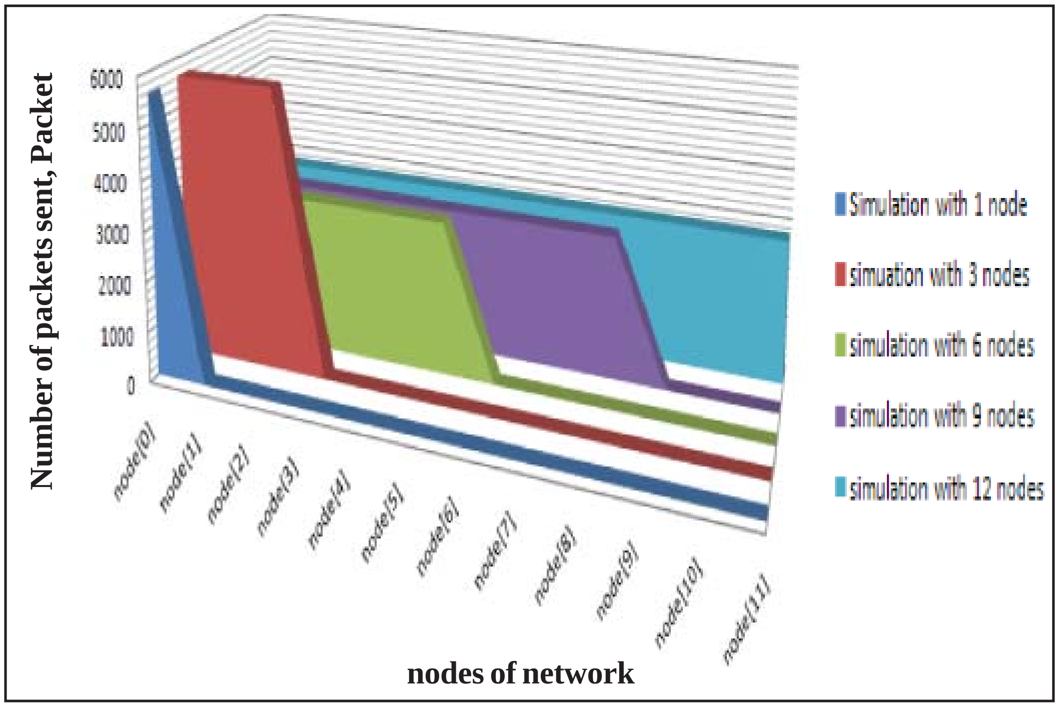

Figure 5. Packets sent from reaction points for different scenarios with QCN mechanism

By increasing number of reaction points, the curve oscillation of queue length gets close to queue maximum capacity bring on packets drop. As shown in Figure 4.

Throughput decreases is depending on ping sent to investigate queue's state. When congestion point receives a ping from reaction point it responses by negatives feedbacks in case of congestion state. The absence of feedback from congestion point gives an implicit signal to reaction points to increase their throughputs.

We can envisage the case of loss or increasing delay of both the special packet and the feedback packet, which could cause throughput increases even when the queue is overloaded. This fact as a result quenches more and more network resource.

Furthermore, the congestion point cannot communicate frequently its congestion stat to reaction points. It has to wait for the special packets (ping) to be delivered before sending feedback signals in order to decrease reaction point's throughput and thus reduce the overload. This fact could strengthen the congestion state for some while before taken the adequate reaction from the rate limiters. Hence, we can deduce that scalability is not well maintained with QCN mechanism.

5. QCN Versus ECM

This section draws parallels between two congestion control mechanisms: Ethernet Congestion Manager (ECM) and Quantized Congestion Notification (QCN).

ECM is a layer two congestion management mechanism. Its assumption consist in pushing congestion from the core of the network to the edge, use rate limiters at edge to shape flows causing congestion and control injection rate based on feedback coming from congestion points. For an overview of relevant work on ECM scheme, see [2] and [3].

5.1 QCN and ECM performance Analyses

Both ECM and QCN mechanisms control congestion by adjusting the injection rate based on feedback coming from congestion point; the feedback contains status and variation of queue length.

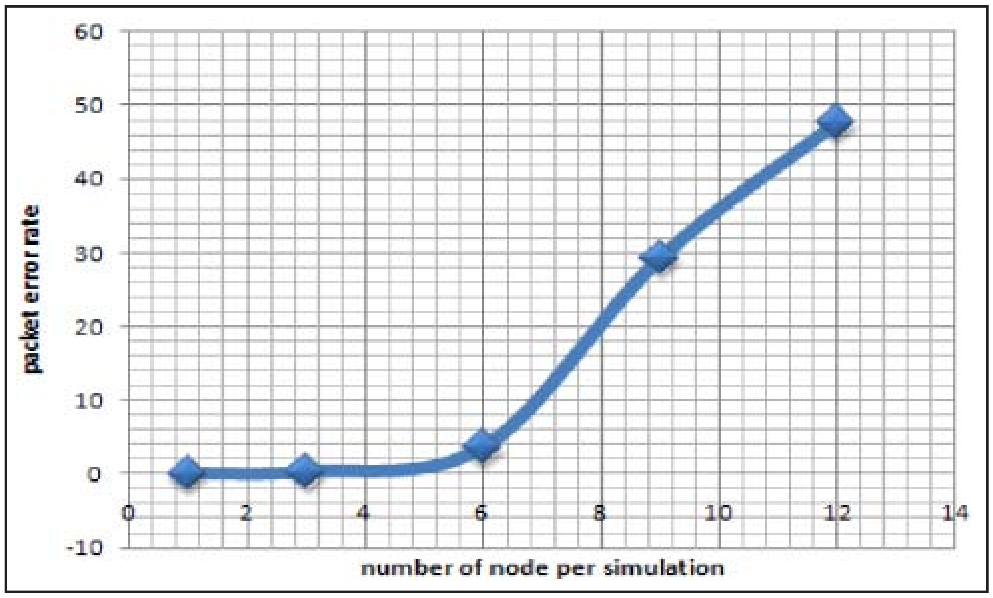

Figure 6. Packet error rate with QCN scheme

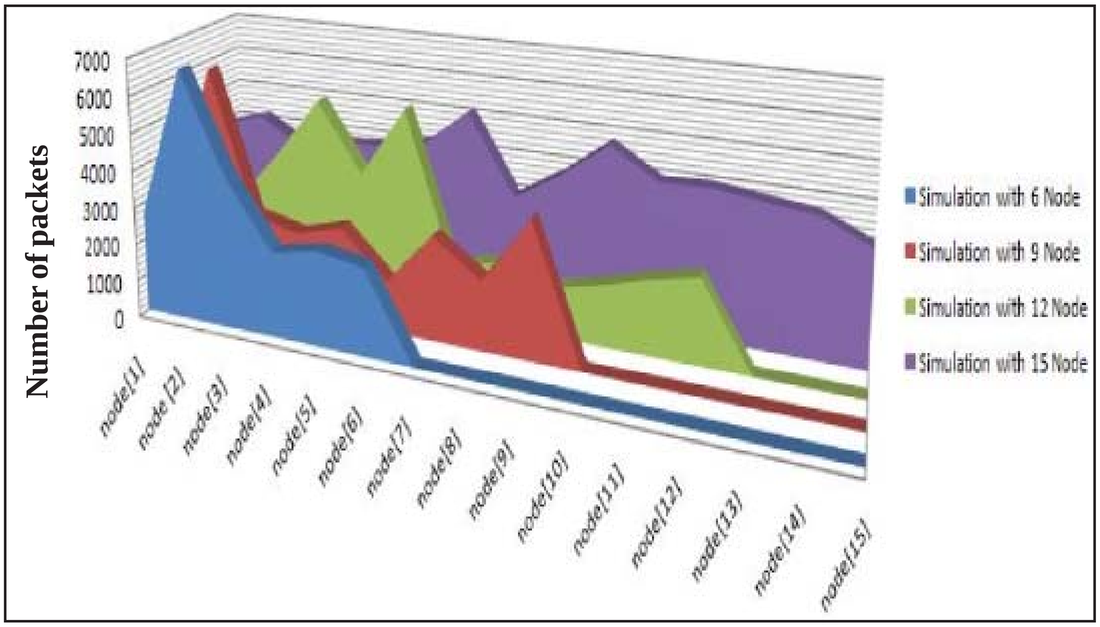

Figure 7. Packets sent from reaction points for different scenarios with ECM scheme

The main components of these mechanisms are:

• Congestion Point: it represents the switch's queue which detects and signals congestion state. It samples an incoming packet, computes the current length and then determines feedback value according to the following formula:

$$Fb = Qoff - W * Qdelta \qquad (2)$$

Depending on the feedback's value, the congestion point sends back congestion notifications to the traffic sources.

• Reaction point adjusts the sending rate according to the received feedback messages. The reaction point may increases its sending rate periodically and voluntarily. Such a rate is typically small and serves to safely probe for extra bandwidth and to ensure fail-safe operation in the case of feedback losses.

Ethernet Congestion Management (ECM) [10] signals positive and negative feedback. Negative feedback is generated only if the queue level exceeds the queue equilibrium threshold (Qeq). Positive feedback is generated only if the sampled packet is marked 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.

The main feature of the Quantized Congestion Notification (QCN) is the total absence of positive feedback. A source increases its flow throughput after a time interval "*WETime*" during which it has not been received any negative feedback. Moreover, the Rate Limiter uses a byte counter and a Timer to pass from one cycle to another in order to control congestion.

As it is, embody from simulations results both ECM mechanism and QCN mechanism allow an efficient control of throughput in reaction points.

From simulations, we raise the problem of lack of stability for both ECM and QCN mechanism. Queue level could not stabilize around equilibrium threshold. Therefore, we pick up queue overuse and then packets drop or queue underuse and then resource waste.

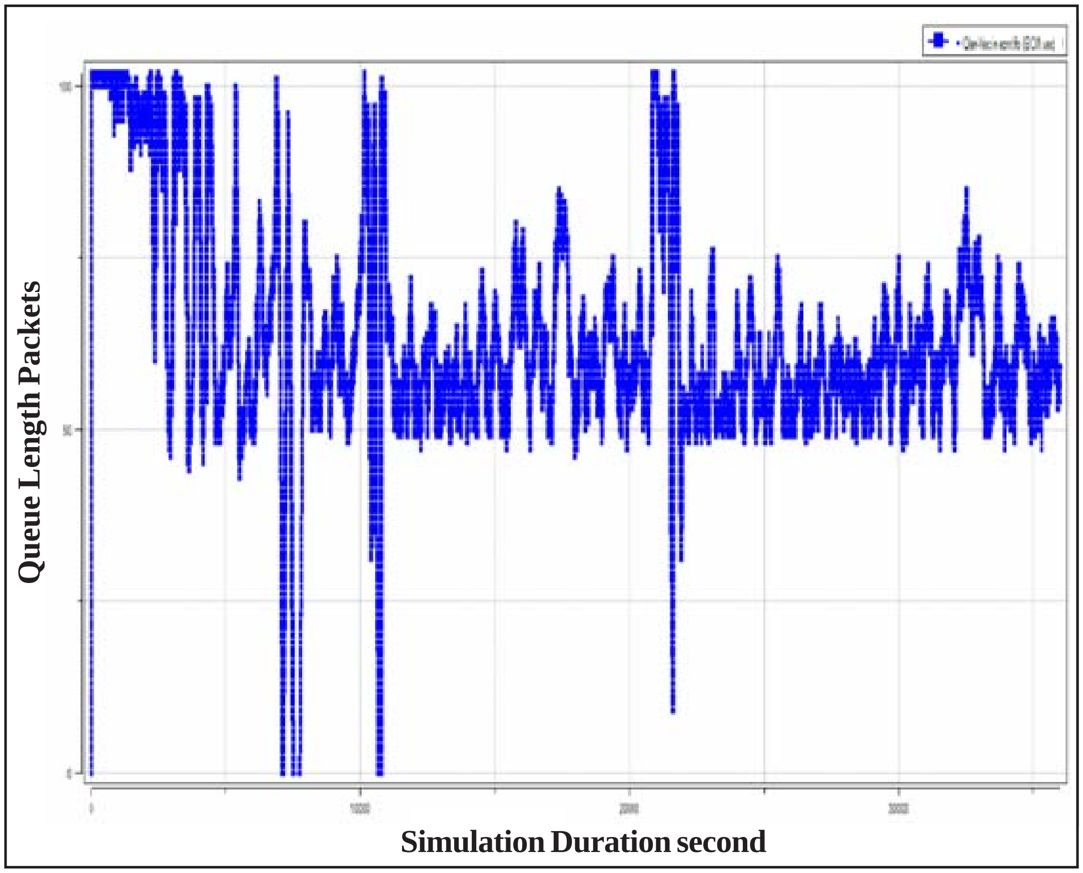

Figure 8. Queue length variation with ECM scheme

Packets error rate for ECM and QCN are close; packets loss is increasing exponentially by increasing reaction point's number as shown in Figure 9 and Figure 6.

Moreover, fairness feature seems to be well maintained with QCN as illustrated by the Figure 5. QCN bring about quicker convergence to fairness. However, traffic flow share unfairness the bandwidth with ECM mechanism as shown in Figure 7.

Furthermore, ECM seems to be more scalable than QCN. In fact, the queue oscillation is not always raised to queue maximum capacity as it is found with QCN and illustrated by Figure 8 and Figure 3. This due to updating of reaction points by the queue's state with positive and negative feedbacks. Congestion notification with ECM mechanism is carried out for each incoming packet that causes queue thresholds excess. However, with QCN mechanism congestion notification depend on special packet, which is sent to explore for an extra bandwidth, and a waiting timer.

We deduce from above that both ECM and QCN mechanism have advantages and disadvantages while controlling traffic.

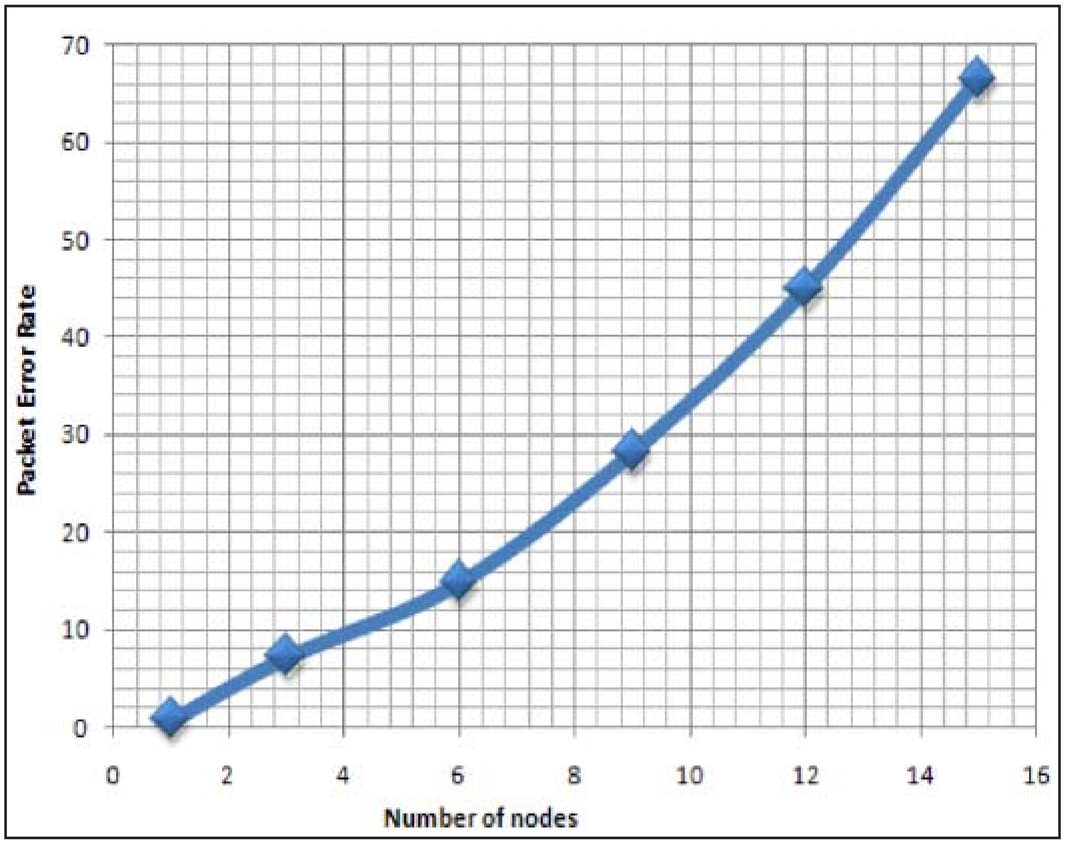

Figure 9. Packet Error Rate variation with ECM scheme

6. Conclusion

This paper presented the QCN congestion control mechanism being considered by the IEEE standardization process. This work takes place to cover overcharge Meto- Ethernet network problematic. In fact congestion management is substantive to ensure that applications have sufficient network resources to communicate effectively and appropriately for the variety of traffic.

This paper also describes simulation implementation and analyses simulation results. Notably, it studies the QCN behavior and performances. Then, it elaborates a comparative analysis between ECM and QCN congestion control mechanism. There is much ongoing and future work. It would be interesting to analyse QCN and ECM behaviour with other complex topology and to study other congestion control techniques for Metro-Ethernet network.

References

[1] IEEE802.1 Qau- Congestion Notification. (2009). www.ieee802.org/1/pages/802.1Qau..html.

[2] Bergamasco, D. (2007). Ethernet Congestion Manager (ECM), IEEE802 PlenaryMeeting.

[3] Jiang, J., Jain, R. (2006). Simulation Modelling of BCNV2.0 Phase1: Model Validation, IEEE 802.1 Congestion Group Meeting, Denver.

[4] Van Foreest, N. (2003). Simulation Queueing Network with OMNeT++.

[5] Tabi, G. Get into GNED, An introduction to the GNED editor of OMNEST/OMNeT++.

[6] Varga, A. (2005). OMNeT++ Descret Event Simulation System version 3.2 User Manual.

[7] Kabbani, A., Lakshmikantha, A., Pan, R., Prabhakar, B., Seaman, M. (1997). QCN:Transience, Equilibrium, Implementation, Telecom Center Workshop.

[8] Kabbani, A., Pan, R., Prabhakar, B., Seaman, M. (2007). QCN: Stably improving transient response, www.ieee802.org/1/files/public/docs2007/au-prabhakar-sonar-fbpause- 1107.pdf

[9] Kabbani, A., Pan, R., Prabhakar, B., Seaman, M. (2007). QCN: Algorithm for Pcode, www.ieee802.org/1/files/public/docs2007/auprabhakar- qcn-with-timer-0711.pdf

[10] Lu, Y., Pan, R., Prabhakar, B., Bergamasco, D., Alaria, V., Baldini, A. Congestion Control in networks with no congestion drops, *www.stanford.edu/~balaji/papers/au-Lu-et-al-BCN-study.pdf*

[11] McAlpine, G. (2005). Congestion Control for Switched Ethernet, Intel Corporation, 8 pages.

[12] Lit, S., Chen, J. G., Ansarit, N. (1998). Fair Queuing For Input-Buffered Switches with Back Pressure, ATM, 22-24 ,first IEEE International Conference.