Empowered by Innovation NEC

# Data Center Quantized Congestion Notification (QCN): Implementation and Evaluation on NetFPGA

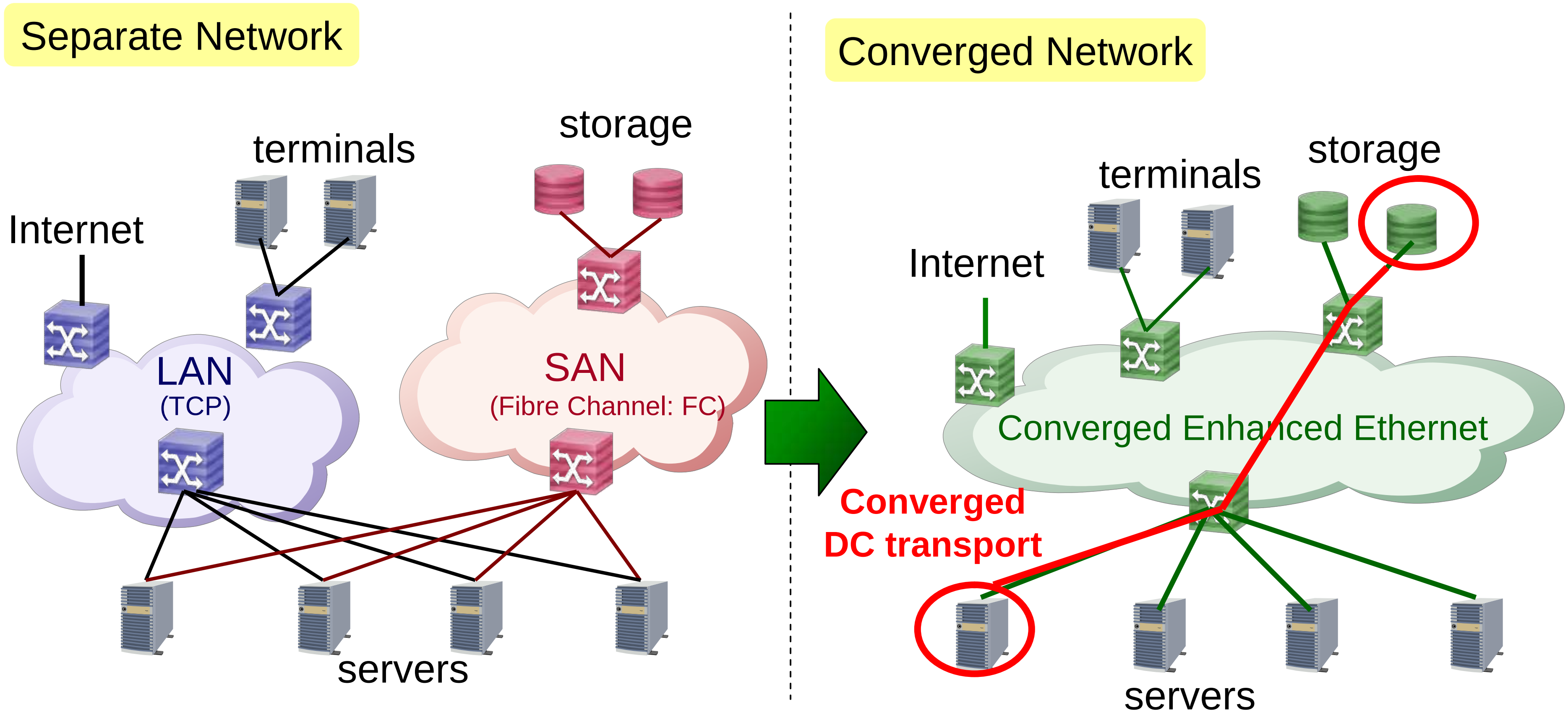
2010 June 14th

Masato Yasuda (NEC Corporation)

Abdul Kader Kabanni (Stanford University)

# Background

- In data centers, there is a movement of Network Convergence
- The spec of Converged Enhanced Ethernet (CEE) has been discussed in IEEE 802.1 standard committees



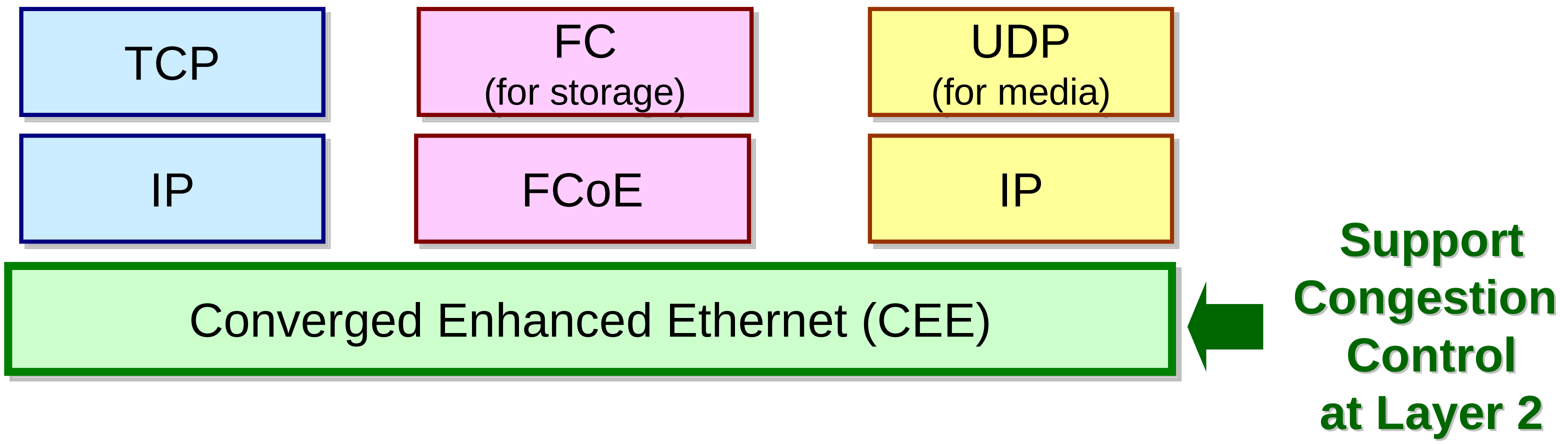

 

Empowered by Innovation NEC

# Congestion Control Mechanism in CEE

## In CEE, it has Congestion Control Mechanism

- Non-TCP traffic (storage, media) can avoid network congestion




## Congestion Notification (CN)

- The spec of Congestion Control Mechanism in CEE
- It is specified in IEEE802.1Qau in Data Center Bridging Task Group
- We proposed QCN (Quantized Congestion Notification) and finally accepted as a standard in March 2010.

# Significant restrictions/requirements for CN

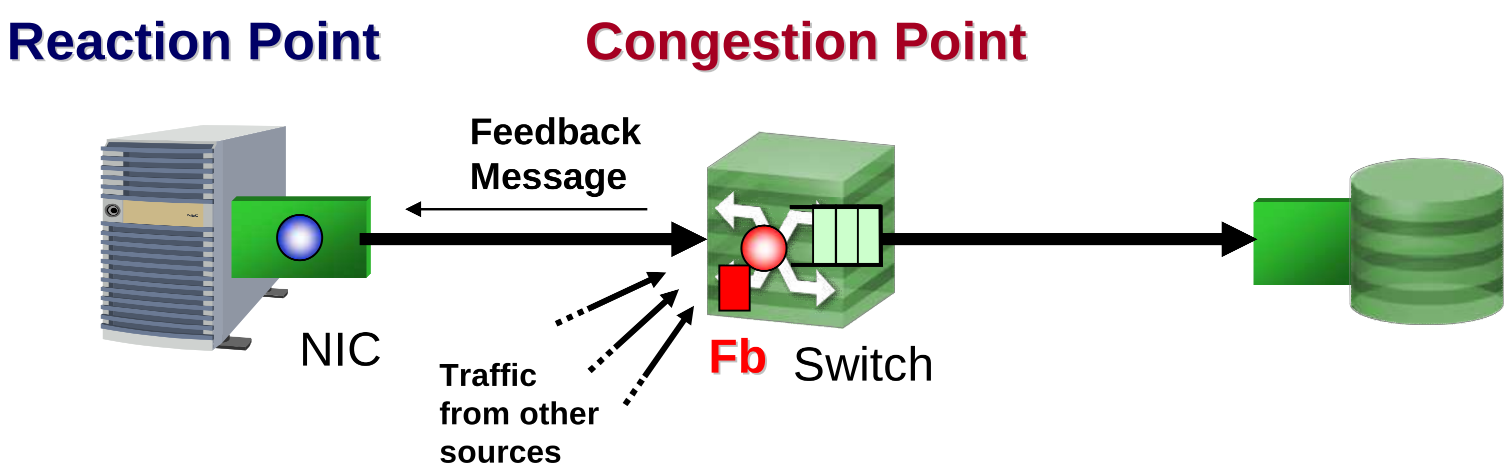
| | |
|---|---|
| **No per-packet ACKs** | No way to know round trip time. Not automatically self-clocked like TCP.<br>(The algorithm need to have counters for self-clocking) |
| **Links can be paused** | Links can be paused by pause mechanism but CN is needed to avoid congestion spreading (pause spreading to innocent paths). |
| **Sources can start at line late** | Unlike TCP slow start, sources can come on at the full line rate of 10Gbps |
| **Stable** | We must avoid queue oscillation which causes overflow (causes link pause) or underflow (lose utilization) |
| **Simple** | The algorithm should be simple enough to be implemented in hardware (for 40G or 100G processing) |

# QCN Overview

# QCN (Quantized Congestion Notification)

- QCN: Congestion Control Mechanism at Layer 2
  - Proposed by our group and accepted in IEEE 802.1Qau

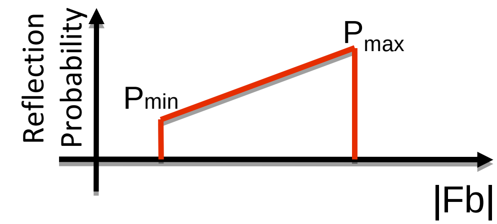
- Terminology:
  - Congestion Point: Where congestion occurs, mainly switches
  - Reaction Point: Source of traffic, mainly rate limiters in Ethernet NICs



# QCN Algorithm: Congestion Point (CP)

## Feedback Processing

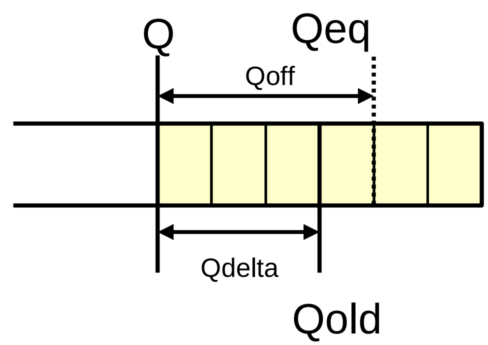
- Calculate Feedback value (Fb) from Switch Queue length
- Send feedback to Reaction Point (RP)




## Fb sampling probability

- No Congestion：Pmin(1%)
- Under Congestion：Pmax(10%) at maximum

## Fb Calculation




$$F_b = -(Q_{off} + w\,Q_{delta})$$

**Qoff: Offset**

$$Q_{off} = Q - Q_{eq}$$

**Qdelta: Variance**

$$Q_{delta} = Q - Q_{old}$$

*Fb* : Feedback (quantized to 6bits)

*Q: Current Queue Length*

$Q_{eq}$ : Qlen in equibrium

$Q_{old}$ : Queue length at previous processing

*w*: fixed parameter (= 2)

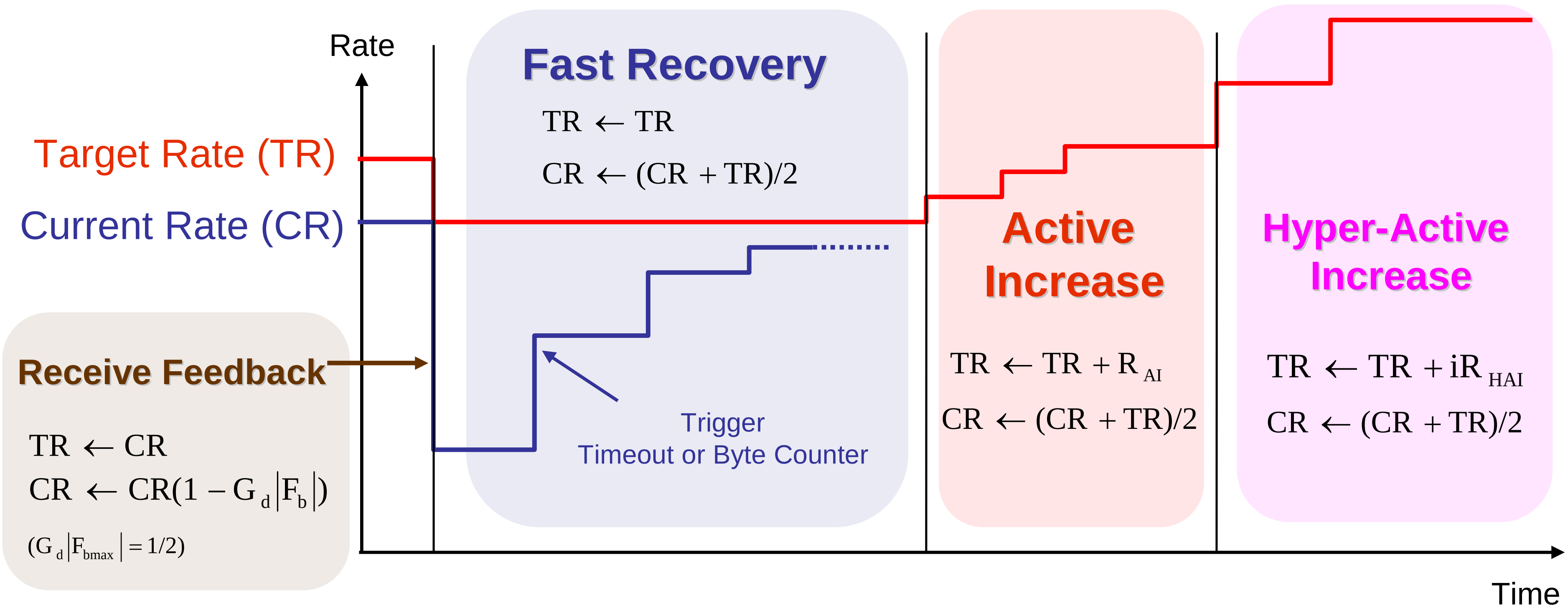
# QCN Algorithm: Reaction Point (RP)

## Rate Control

- Rate Decrease： Feedback from CP
- Rate Increase: Increase by itself

## Rate Increase Algorithm：Averaging Principle

- Based on BIC-TCP：Works without ACK



# Implementation

# Current QCN Implementation

## World's first Hardware QCN Full implementation

- It is fully functional
- We confirmed that the result matches OMNet++ simulation result!

## QCN-NIC

- Multiple Reaction Points
- Improved Token Bucket Rate Limiter

## QCN-Switch

- Multiple Congestion Points

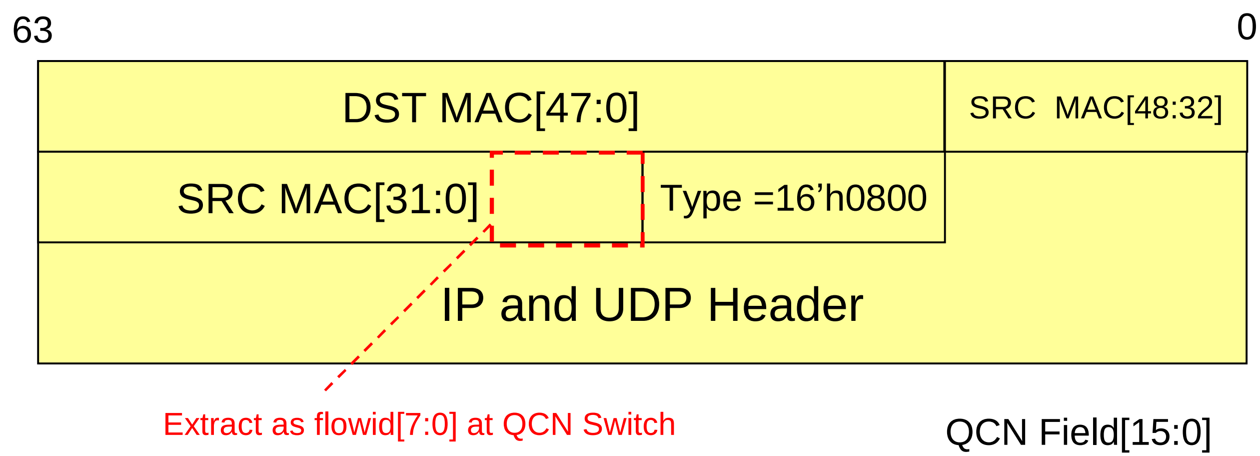
## Compliant with Pseudo Code v2.3 [1]

- 1Gpbs Platform (NetFPGA)

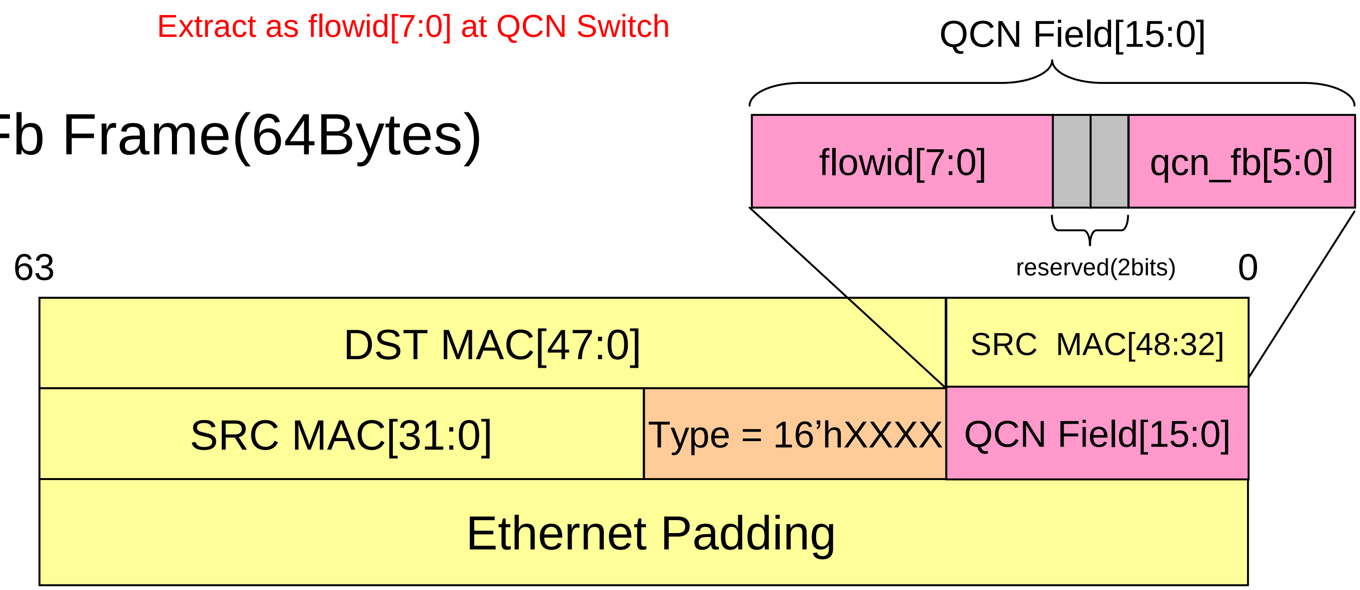
[1] QCN Pseudo Code v2.3: http://www.ieee802.org/1/files/public/docs2009/au-rong-qcn-serial-hai-v23.pdf

# Packet Format

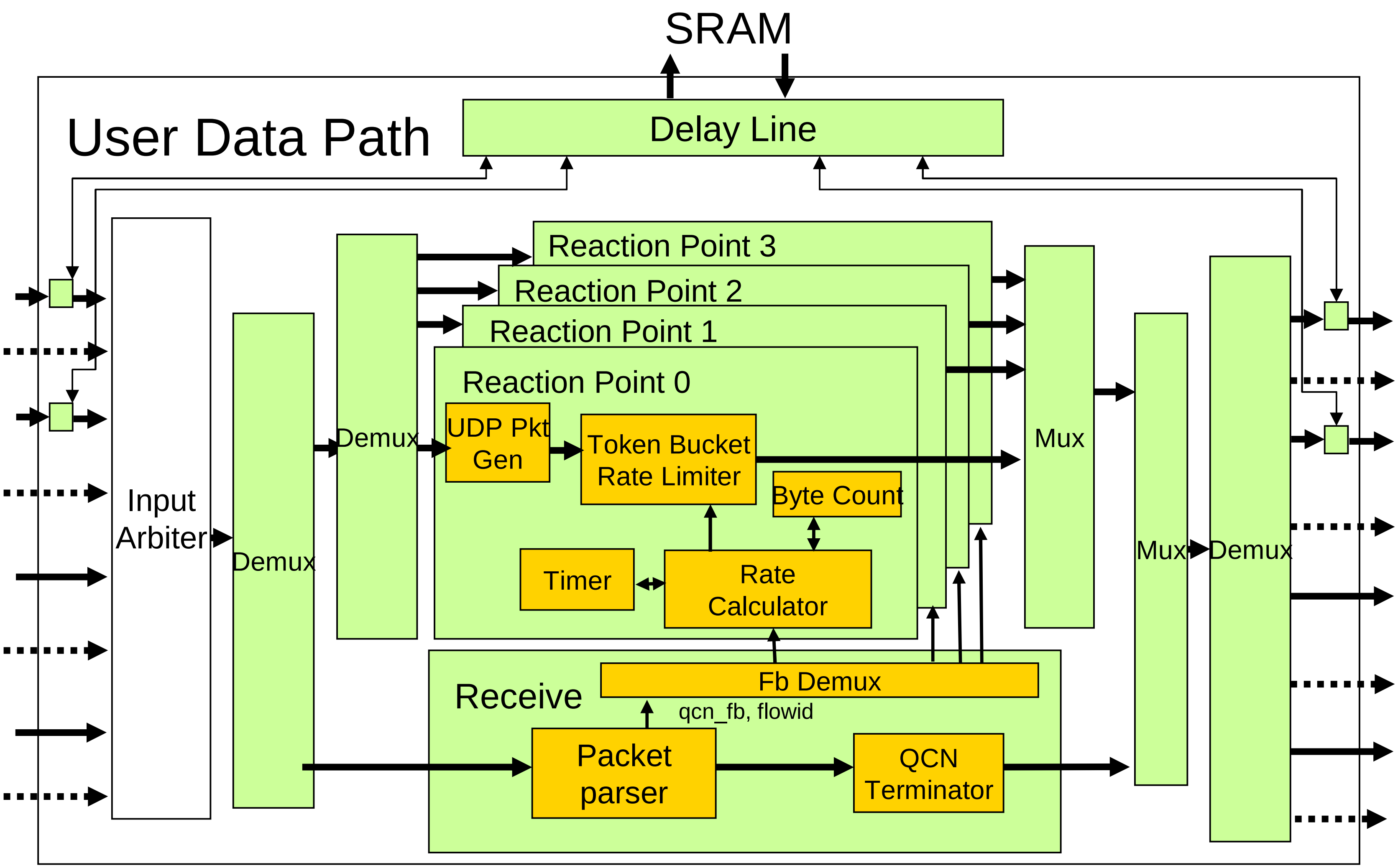
## Data Frame (Normal Ethernet UDP/IP Frame)




## QCN Fb Frame(64Bytes)

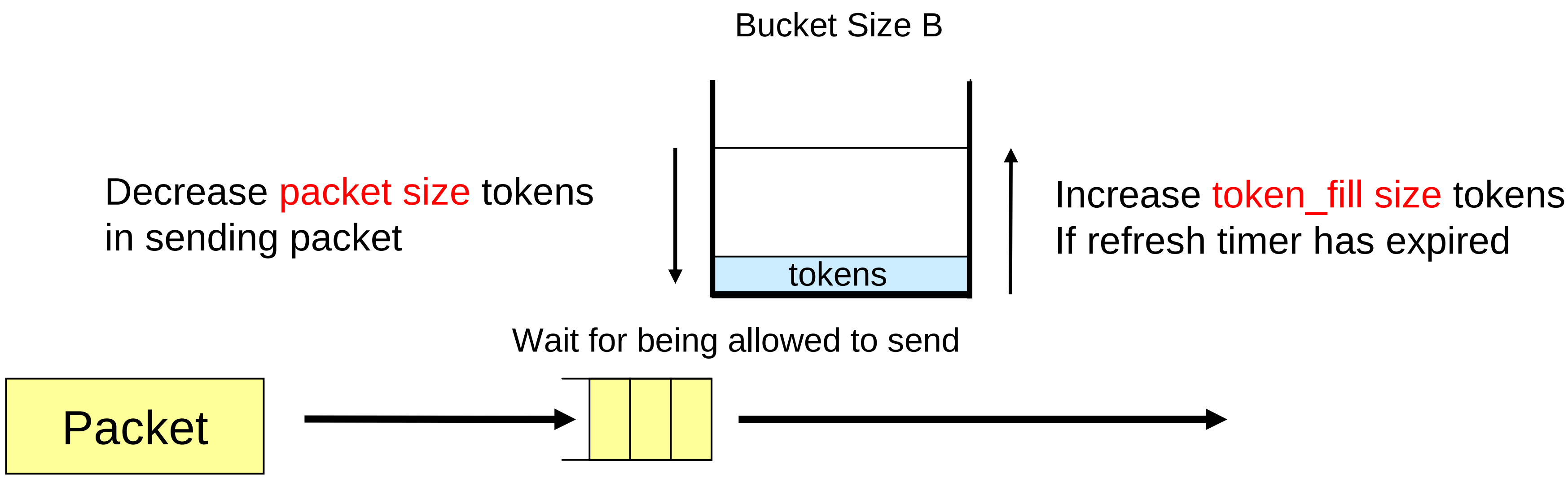


# QCN NIC Block Diagram



## Token Bucket Rate Limiter

### Token bucket algorithm avoiding burstiness




**Previous implementation**

Send Condition:
tokens in the bucket > 0

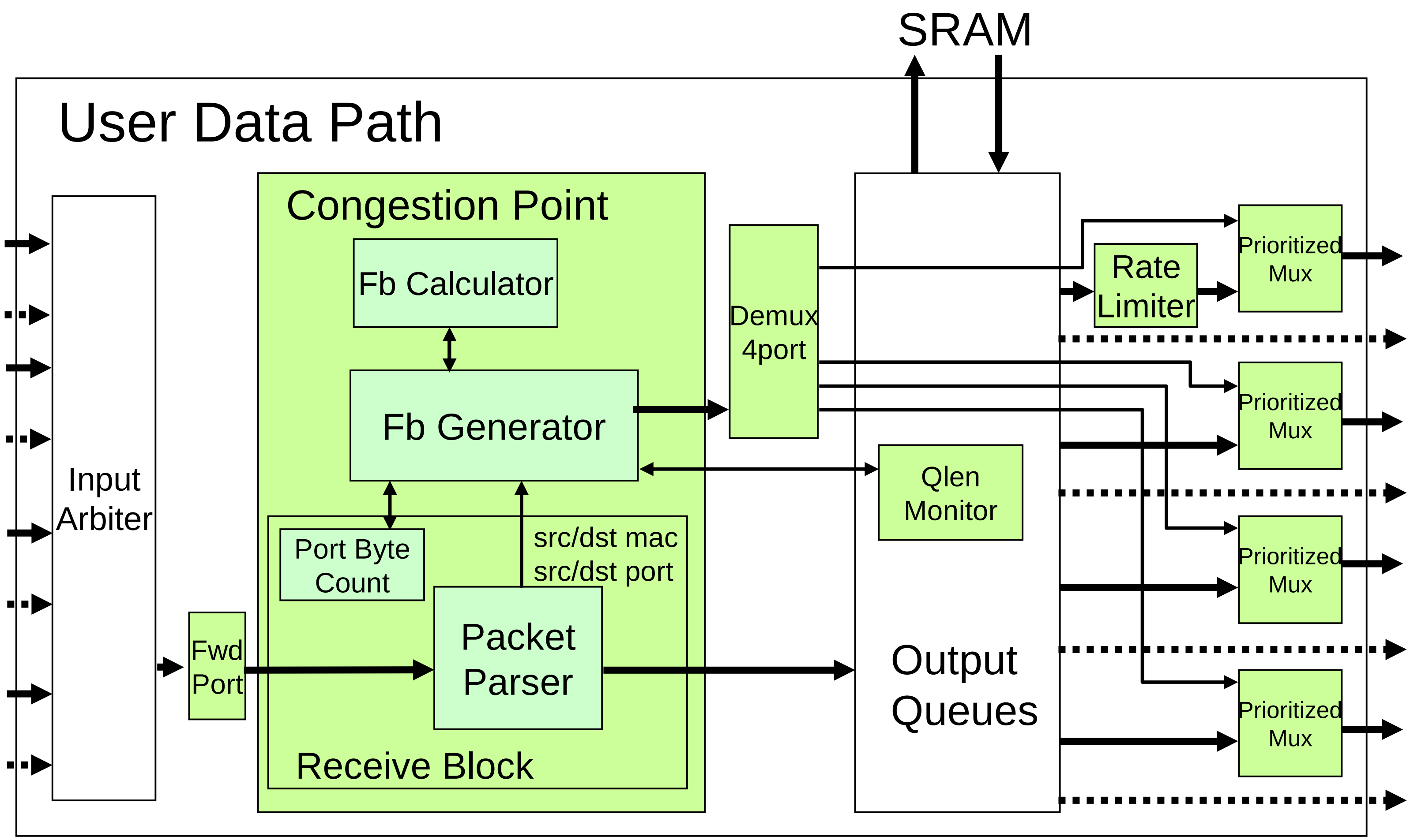
**Causing burstiness!**

**Current implementation**

Send Condition:
tokens in the bucket >= token size

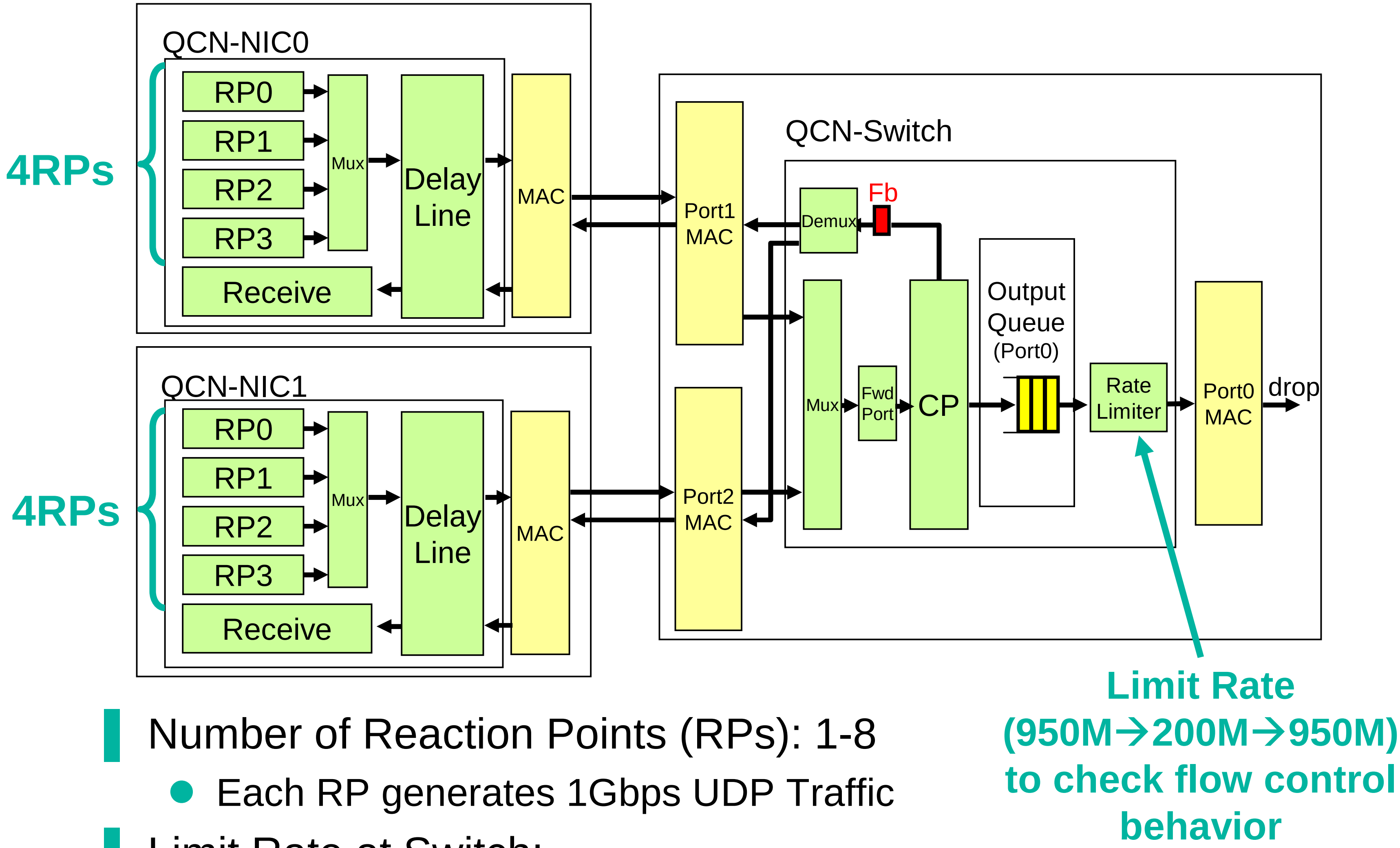
**Working Fine**

# QCN Switch Module Structure



# Evaluation

# QCN Hardware Evaluation System




- Number of Reaction Points (RPs): 1-8
  - Each RP generates 1Gbps UDP Traffic
- Limit Rate at Switch:
  - 950Mbps(3.7sec) → 200Mbps(3.7sec) → 950Gbps(3.7sec)

# Parameters (QCN)

## NIC

- FAST_RECOVERY_THRESHOLD = 5
- AI_INC = 0.5 Mbps
- HAI_INC = 5 Mbps
- BC_LIMIT = 150 KB (30% randomness)
- TIMER_PERIOD = 25 ms (30% randomness)
- MIN_RATE = 0.5 Mbps
- GD = 1/128

Based on QCN Pseudo Code Parameters

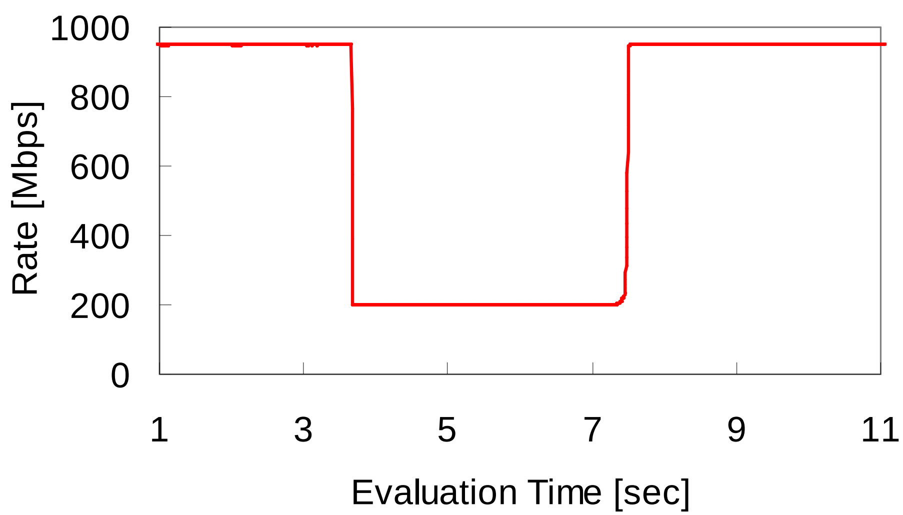
## Switch

- Quantized_Fb: 6 bits
- Q_EQ = 33 KB
- W = 2
- Base marking = 150 KB, and varies according to the lookup table in the pseudo code (30% randomness)

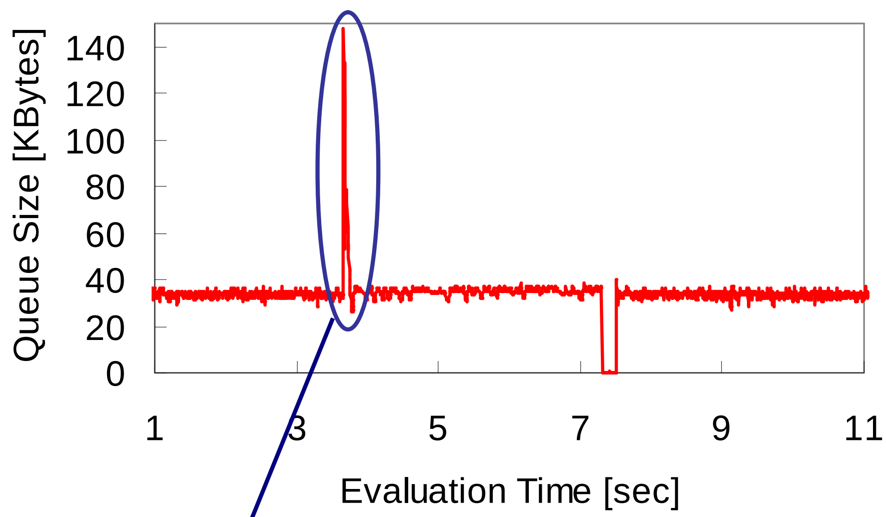
# Evaluation Result

# Hardware Evaluation Result (1RP) , RTT=100usec

NIC: Rate



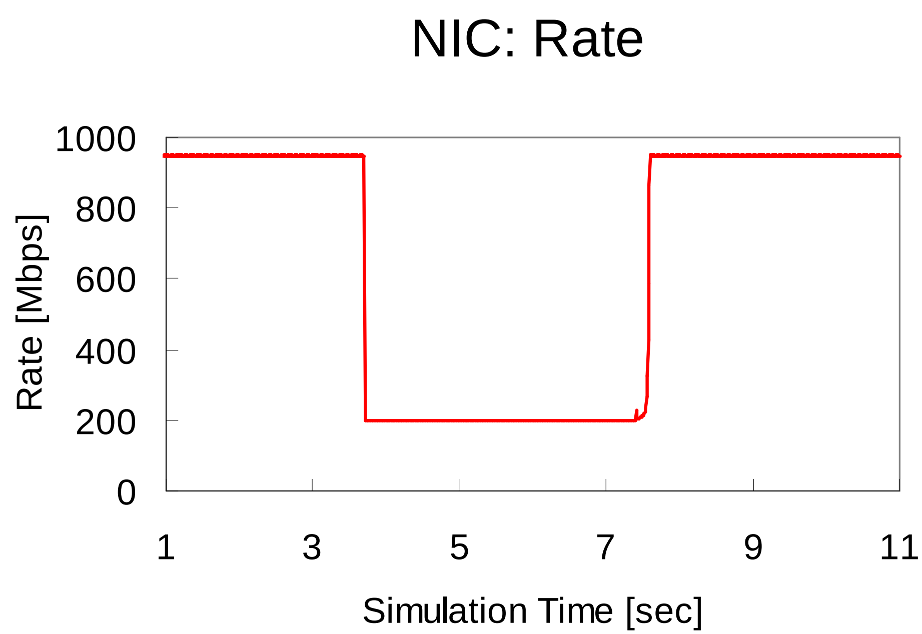

Switch: Queue Length

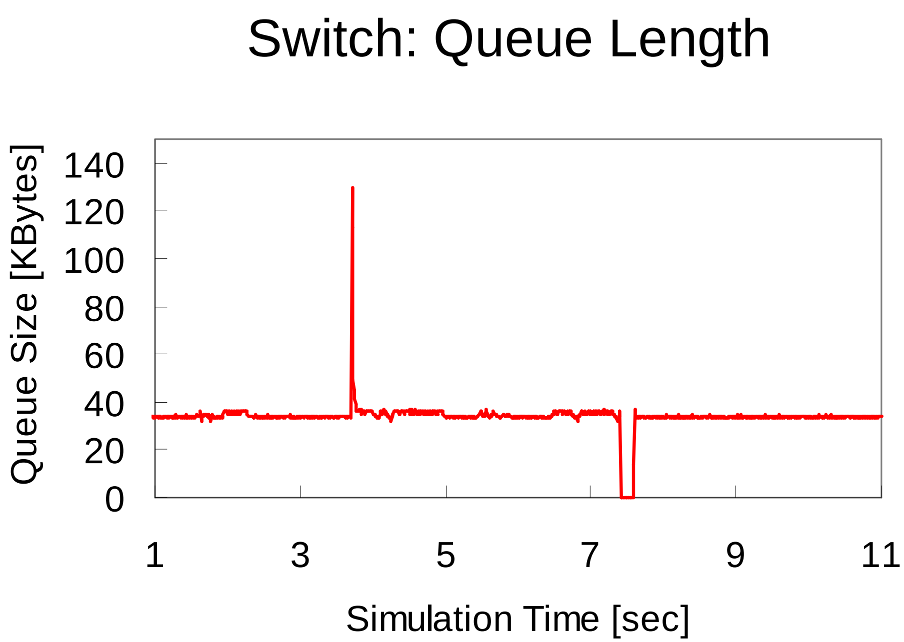



The NIC rate behavior well follows
the limited Switch rate
by QCN Algorithm

Queue length rises temporarily
in limiting Switch rate
but go back to Qeq in shorter time.

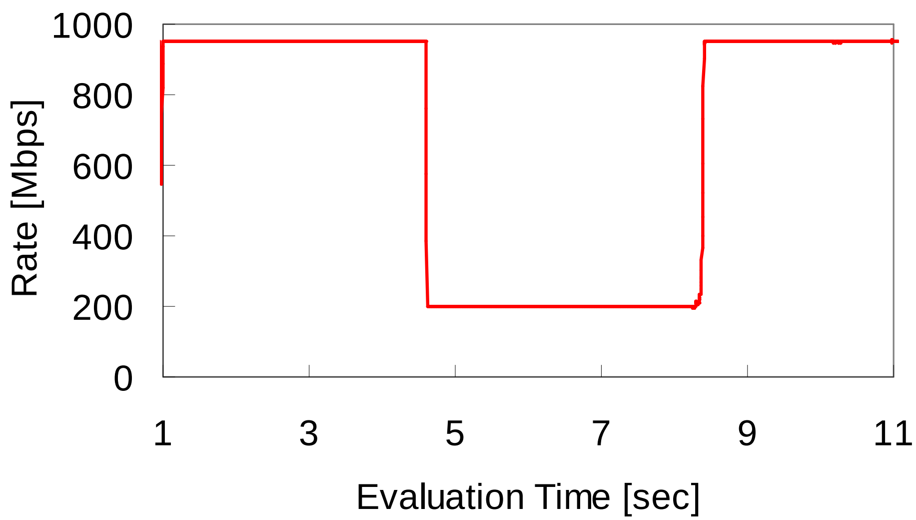
# OMNet++ Simulation Result (1RP) , RTT=100usec

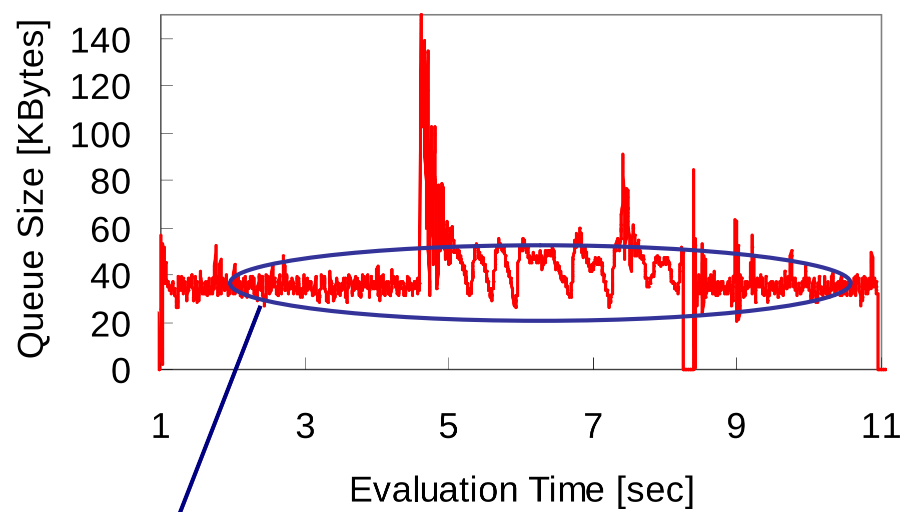



Simulation result well matches the Hardware Evaluation Result !

# Hardware Evaluation Result (8RPs) , RTT=1msec




The aggregate NIC rate behavior still well follows the Switch limited rate

Wiggling because of longer delay but still keeps the same queue length.

It also matches OMNet++ Simulation Result with the same parameter

# Demonstration

# Conclusion

- We presented/demonstrated QCN theory and implementation

- QCN Implementation on NetFPGA Board
  - All QCN Functions are implemented (Pseudo Code v2.3)
  - QCN-NIC
    - Multiple RPs are ready
    - Improved Token Bucket Rate Limiter
  - QCN-Switch
    - Multiple CPs are ready
    - Binary Search Logic for Fb Calculation
- We have proved that our QCN implementation achieves expected performance.
  - The result matches OMNet++ Simulation