Congestion Management Protocol Characteristics in Complex Simulation Scenarios with Updates for QCN-Sonar and large RTT

Guenter Roeck, Teak Technologies
Mitch Gusat, IBM Zurich Research Lab

IEEE 802.1Qau
Atlanta Meeting, November 2007
Update Stockholm presentation with new data for

- ECM with RTT adjustments
- QCN-SP with RTT adjustments (QCN+)
- QCN-Sonar

Present QCN+ details
Simulated Protocols

- **ECM**
  - ECM with $<W, Gi, Gd>$ auto-adjusted for RTT

- **QCN**
  - As specified

- **QCN-Sonar**
  - As understood
  - Not all details included

- **QCN-SP [QCN+]**
  - QCN with Sub-path probes, auto-adjusted for RTT, N
Test Scenarios

- **OG hotspot with oscillating service rate**
  - Simulate transient congestion in higher priority CoS
  - Look for overall throughput

- **Baseline scenario with large forward latency**
  - Simulate network with large BW * latency product
  - Look for stability (throughput, queue length)

- **Large number of hotspots with dynamic load**
  - Simulate complex network with high load and many CPs
  - Look for overall protocol performance (throughput)
Single Oscillating Hotspot, 20 nodes

All nodes (20): Bernoulli distribution, load: 8.5 Gb/s
- From t=0 to 1s

Node 1 (hotspot) service rate: 1Gb/s
- Duration: 800mS from ti=100ms to 900 ms
- Frequency: tOn=2..50ms, tOff=2..50ms

Looking for Throughput distribution and bandwidth loss

Real world scenario: Higher priority CoS with recurring transient congestion
Oscillating Hotspot: Throughput Distribution

ECM

QCN+

QCN

QCN-Sonar
Oscillating Hotspot: Bandwidth Loss

ECM

QCN

QCN+ 

QCN-Sonar
Observations

- QCN-Sonar performs better than QCN and QCN-FbHat
  - Still significant throughput loss

- Best performer is still ECM
Symmetric Topology, Single HS, Large Forward Latency

- 4 - 512 Nodes
- 8 - 1024 uS forward latency from nodes to switch
- Load factor 4 (load adjusted with number of nodes)
- Simulation runtime 1s, with load from 0.1s to 1.0s
- Measure throughput and average queue length
Throughput at Hotspot, ECM

Throughput

RTT

Nodes

Nodes

RTT

Throughput

Nodes
Throughput at Hotspot

QCN-Sonar, no FR1 rate adjustment

QCN-Sonar, FR1 adjustment

QCN+, adjusted for RTT only

QCN+, adjusted for RTT and N
Average Queue Length at Hotspot, ECM
Average Queue Length at Hotspot

QCN-Sonar, no FR1 adjustment

QCN-Sonar, FR1 adjustment

QCN+, adjusted for RTT only

QCN+, adjusted for RTT and N
1ms Latency, 4 Nodes: Throughput at Hotspot

ECM

QCN+, adjusted for RTT and N

QCN-Sonar, no FR1 adjustment

QCN-Sonar, FR1 adjustment
1ms Latency, 4 Nodes: Queue Length at Hotspot

ECM

QCN+, adjusted for RTT and N

QCN-Sonar, no FR1 adjustment

QCN-Sonar, FR1 adjustment
Probe Traffic vs. Number of Nodes

QCN-Sonar (10ms base timer)  QCN+

Chart showing the comparison of probe traffic vs. number of nodes between QCN-Sonar (10ms base timer) and QCN+.
Observations

- Adjusting probing protocols for RTT / N works well

QCN-Sonar

- Works well with large $\langle N \rangle$
- Weak spot with $\langle$large RTT, small $N\rangle$
- FR1 transient adjustment sounds like a good idea
  - Over-adjustment will need improvements
- Probe reply rate proportional to number of queues / flows
  - Can get very large, especially with smaller timer values
N=18 switches; 3 hosts per switch
Node \(<i>\) sends to node \(<i+3>\); Node \(<i+1>\) sends to node \((N-1)*3+2\); node \(<i+2>\) sends to node \(<i+4>\)
Node \(<1,4,7,...>\) sends bursty traffic with interval \(1 + <i>*0.1\) ms
100% load from all nodes
Node \((N-1)*3+2\) receives traffic from \(<N>\) sources
N hotspots
20-stage hotspot: Throughput at last hotspot

ECM

QCN

QCN+ 

QCN-Sonar
20-stage hotspot: Queue length at last hotspot
20-stage hotspot: Switch 2 Throughput

ECM

QCN

QCN+

QCN-Sonar
20-stage hotspot: Per-Flow Throughput

ECM

QCN+

QCN

QCN-Sonar
20-stage hotspot: Total Throughput through all hotspots

- ECM: 18746606764
- QCN-SP: 19718283375
- QCN: 18756144104
- QCN-H: 19308052798
- QCN-Sonar: 18344414462

Total throughput (Bytes)
20-stage Hotspot: Throughput per switch
Observations

QCN-Sonar slightly worse than QCN in this test
  Maybe due to different parameters
QCN-Sonar

- Improvement over QCN and QCN-FbHat
- Introduces positive feedback
  - Lack of negative feedback equivalent to explicit positive feedback
- Introduces gradual positive feedback, especially with low data rates
  - More binary feedback per data rate → impact similar to fewer messages with gradual increase
- Introduces association to RTT
  - Timer setting reflects maximum supported RTT

Problem areas

- Spurious Rate Limiters
- Recovering available bandwidth
- <Large RTT, low N>

Impact of timer-based positive rate adjustment needs further study
QCN+
  Good performance if adjusted for RTT, N

ECM
  Good performance for small # of flows if adjusted for RTT
  Would need adjustment for N (# of flows)
CPID enables association of Congestion Points to Rate Limiters

Thus, CPIDs improve protocol scalability and reduce number of required Rate Limiters

Without CPIDs, the number of required Rate Limiters strictly depends on the number of L2 flows
   - More RLs will be needed to achieve comparable performance

There may be a large number of L2 flows per CP from a single source

As a result, protocol performance will suffer if CPIDs are not available
   - Especially if a CP only supports a few Rate Limiters
   - Protocol scalability will suffer as well
Timer-based feedback introduces unknown elements

- Rate gain inversely proportional to data rate
  - In other words, introduces rate based gain adjustment
  - Protocol favors flows with low datarate

- Timer dependencies on link rate
  - Impact on multi-speed networks (low->high, high->low)?

- Feedback rate depends on # of RLs, not on link rate
  - As a result, # of feedback messages can get large with
    - Large number of flows
    - Multiple CPs, unless DE bit is reset

Introduces RTT dependency

- “hard” RTT limit depending on timer settings
Protocol overhead depends on the # of Rate Limiters
- More RLs → more overhead

As a result, Timer based feedback introduces conflicting objectives
- Reduce # of RLs and use larger timer values to limit overhead
- Increase # of RLs and use smaller timer values to improve protocol performance

In combination with lack of CPID availability, overhead no longer determined by protocol, but by RP implementation decisions
- RP implementation decisions will have impact on CP workload
“No longer congested” condition in CP is determined as “Queue length < Qeq/<factor>” for a period of time

Problem is that even a marginally loaded switch may experience spurious queue length buildup
  - See Stockholm presentation → Spurious rate limiters

Worse, just one or two jumbo packets will create sufficient queue length
  - \(2 \times 9k = 18k > \frac{Qeq}{2}\)

As a result, switches may not exit “congested” condition for a long period of time, even if their average link utilization is well below 100%
Conclusions

- QCN has come a long way
  - QCN-Sonar promises significant improvement over QCN and QCN-FbHat
  - Introduces positive feedback

- QCN-Sonar introduces several new concepts
  - Timer-based feedback loop
    - “No longer congested” condition in CP
    - Must be carefully studied
  - Widest stability range achieved with closed-loop (probing) protocols with RTT and rate based gain adjustment
Still convinced that we need **explicit** probing

- Enables RTT calculation and RTT based adjustments
- Improved transient response
- Achieve acceptable performance in OG hotspot scenarios
  - Faster recovery due to **explicit** positive feedback
- Improved performance with large latencies, low N
QCN+ (QCN-SP) at a Glance

What is QCN+?
- A hybrid between QCN and ECM with Sub-path probing. Architecture, operation, pseudo-code and simulation results are available.

What is mandatory / optional?
- M: Closed loop for increase and decrease controllers.
- O1: Probing → [Robustness]
- O2: Open loop rate increase → [cope w/ failures, ECN loss]

Distinctive feature(s)? Probing, positive feedback, fail-safe.
- Robust and scalable. Self-tuning [w/ probing].
- Closed loop => improved capacity tracking and dynamic response.
- Reliable: failures drive QCN+ into “fail-safe mode” (baseline QCN).

Complexity / performance ratio?
- Algorithm: simple fast rate recovery
- Implementation: comparable to baseline QCN
Probes sent for rate limited flows in regular intervals
Probes sent at same priority as data frames
Probe replies include Qoff, Qdelta
Reaction Point takes RTT into account when adjusting its transmit rate
All Probes replied to

In-path switches adjust Qoff/Qdelta/Fb

To solicit explicit positive feedback

CP sends Qoff/Qdelta in addition to Fb

To enable RTT calculation

Probes sent whenever TO_THRESHOLD expires

Probes sent whenever TO_THRESHOLD expires

Qoff/Qdelta in addition to Fb

Quantized Qoff, Qdelta

Quantization against Qeq

qQoff = Qoff * 64 / Qeq;

qQdelta = Qdelta * 64 / Qeq;

Fb calculation at RP

Fb = -(qQoff - W * qQdelta) / (2^W+1)
RTT based loop gain control in RP
- Accept one negative adjustment per RTT
- Adjust TO_THRESHOLD based on RTT and current datarate
- Adjust W (and calculate Fb) based on RTT and current datarate
- Reduce positive loop gain based on RTT

ToThreshold adjustment
- Set ToThreshold to \text{max}(\text{TO\_THRESHOLD}, \text{RTT} \times 2 \times \text{rate})
- Effect on positive feedback similar to QCN-Sonar’s timer based approach
  - Smaller data rate $\rightarrow$ more probes per amount of data sent

W adjustment
- \( N = \text{<switch link capacity>} / \text{<current rate>} \)
- \( W = \text{baseW} + (\text{RTT} \times \text{<factor>} / \text{N}) \)
Use variable sampling interval instead of sampling probability to generate CM packets

- Next sample interval is calculated when a sample is taken
- More stable than using sampling probability
  - Sampling packet generation is more predictable
  - Integration effect when calculating next sample interval
    - Protocol does not immediately react to short spikes in queue length

Use Qoff to determine if to send negative CM adjustment messages

- Create RL at RP if resulting Fb is negative
- No more spurious rate limiters
Tested (or, rather, played with) several methods

- Qsat as with ECM
- Unlimited $Q_{off}/Q_{delta}$ (not limited to multiples of $Q_{eq}$)
- Rate adjustments based on link capacity reported by switch
  - $F = \frac{<C_i>}{<C_{i-1}>}$
  - $R_i := R_{i-1} \times F$
  - use this rate if lower than $F_{b}$-calculated rate

Needs further study

- QCN-Sonar approach looks promising
Why keep pushing?
Oscillating Hotspot: Throughput Distribution

ECM

QCN, No hyperactive increase

QCN, Hyperactive increase

QCN-Sonar
Oscillating Hotspot: Bandwidth Loss

ECM

QCN, No hyperactive increase

QCN, hyperactive increase

QCN-Sonar
Large RTT/N: Throughput at Hotspot

QCN+

QCN, No hyperactive increase

QCN-Sonar, FR1

QCN-Sonar, No FR1
Large RTT/N: Average Queue Length at Hotspot

QCN+

QCN, No Hyperactive increase

QCN-Sonar, FR1 adjustment

QCN-Sonar, no FR1 adjustment
Since it was introduced, QCN has much improved

Without pushing, improvements would not have happened
  - In fact, each time we keep hearing that my simulation results would not be correct
  - Which is then followed by protocol improvements

QCN, if adopted, still needs significant improvement
  - Still not at par with ECM or QCN+

Either fix, or adopt another scheme
Further study

- QCN-Sonar, after Pseudo-code published
- Transient reaction
  - Improve protocol reaction time for <large N, low bandwidth> scenarios
- QCN-style non-linear increase/decrease may have negative impact on stability w/ large RTT, especially with low N
  - Test ECM with adjustments for N/rate
  - Test ECM-style increase/decrease with Sub-path probing
- Optimize probe traffic
  - Can be reduced significantly, especially with low N (high data rate)
Teak simulation code access

- **OMNET++**
  - Download from [www.omnetpp.org](http://www.omnetpp.org)

- **INET framework**
  - **git access (linux):**
    
    ```
    git clone git://teaktechnologies.com/var/git/INET.git
    cd INET
    git checkout -b my_branch origin/teak
    ```

- **Latest code not yet included**
  - Will be added after cleanup

- **Please keep in mind that this is GPL code**
  - You are expected to publish your modifications
Thank You
Backup Slides
Transforming QCN-Sonar into a Positive Feedback Scheme

1. Send explicit probes instead of tagging data packets
2. Drop probe if node in path is congested
3. At CM domain edge (switch or NIC), echo probe if not congested
4. At RP, increase rate if probe reply was received
Simulation Parameters

- **Traffic**
  - Bernoulli
  - 1500 byte frames

- **System**
  - Switch latency (processing time) = 1us
  - Link latency = 500ns
  - Switch frame capacity = 200kB, 250 packets
  - PAUSE generated by switch
  - RP egress buffer size 100 packets
Simulation Parameters - QCN-xx

- Drift factor = 1.0005
- Timer period = 1ms (or disabled)
- Extra fast recovery enabled
- EFR MAX disabled
- A = 12 Mbit
- Fast Recovery Threshold = 5
- Gd = 1/128
- TO_THRESH = 150 kBytes
- Qeq = 24kB
- QCN packet processing latency = 5uS
- Hyperactive Increase enabled/disabled
- Psample = 1% .. 10%
- W and gain adjusted for RTT
Simulation Parameters - ECM

- Qeq = 375
- Qsc = 1600
- Qmc = 2400
- Qsat enabled/disabled
- Gi = 0.53333 (adjusted for RTT)
- Gd = 0.00026667 (adjusted for RTT)
- Ru = 1,000,000
- Rd = 1,000,000 (10,000 for large N)
- Td = 1ms
- Rmin = 1,000,000
- W = 2.0 ..32 (adjusted for RTT)
- samplingInterval = 150000