QCN: Problems, Solutions, and a Convergence Proposal

Davide Bergamasco

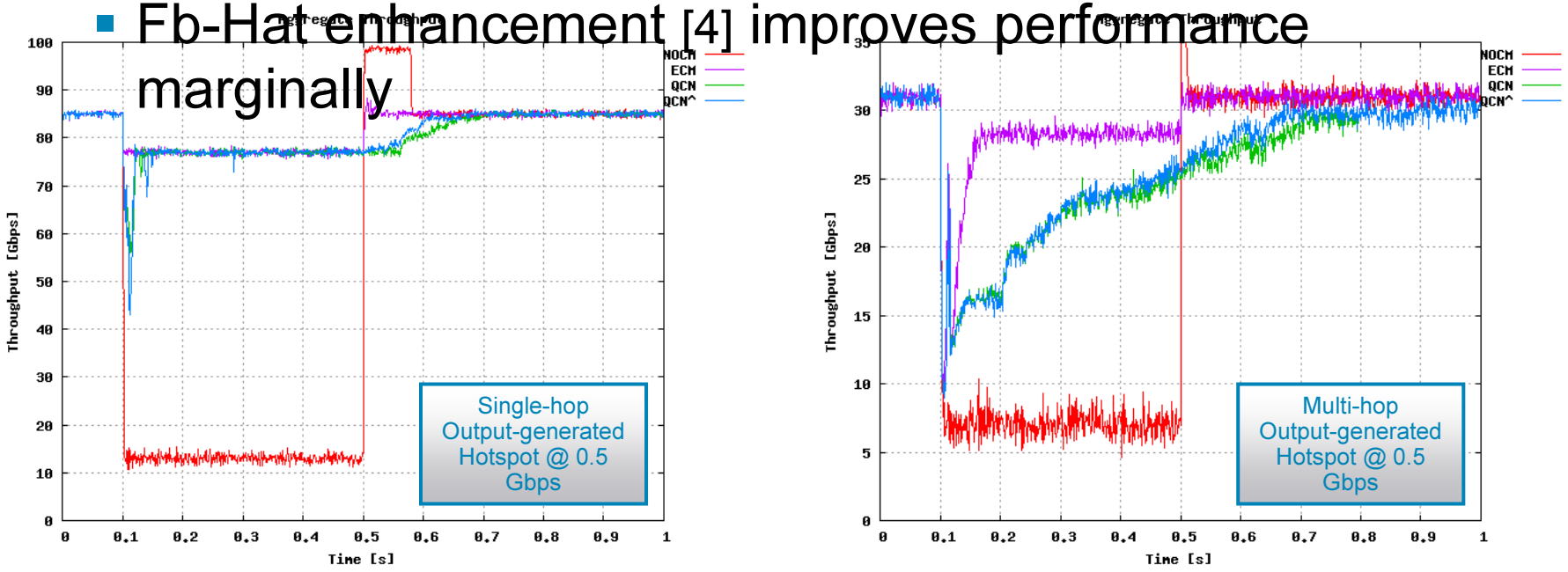
IEEE 802.1 Interim Meeting
Stockholm, Sweden
September 5th, 2007

Agenda

- Analyze QCN issues
- Provide a possible solution
- Present converged proposal
- Details

QCN Issues

- A recent body of simulation work [1,2,3] shows that QCN does exhibit poor performance in the post-transient recovery
- Fb-Hat enhancement [4] improves performance marginally



[1] http://www.ieee802.org/1/files/public/docs2007/au-roeck-simulation-results-071707.pdf,

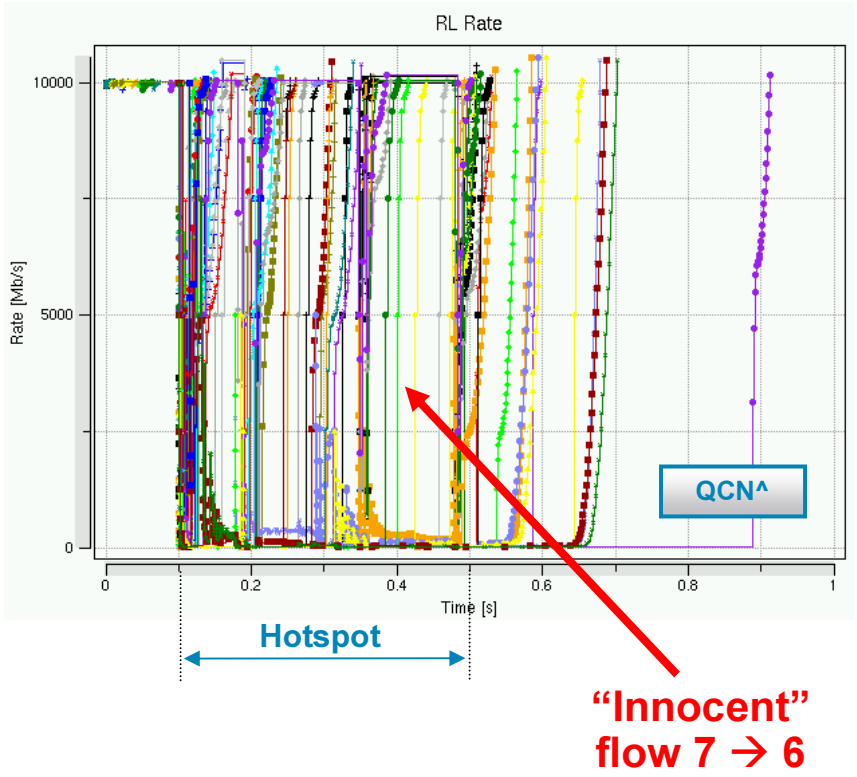
[2] http://www.ieee802.org/1/files/public/docs2007/au-ZRL-prelim-QCN-r1.01.pdf,

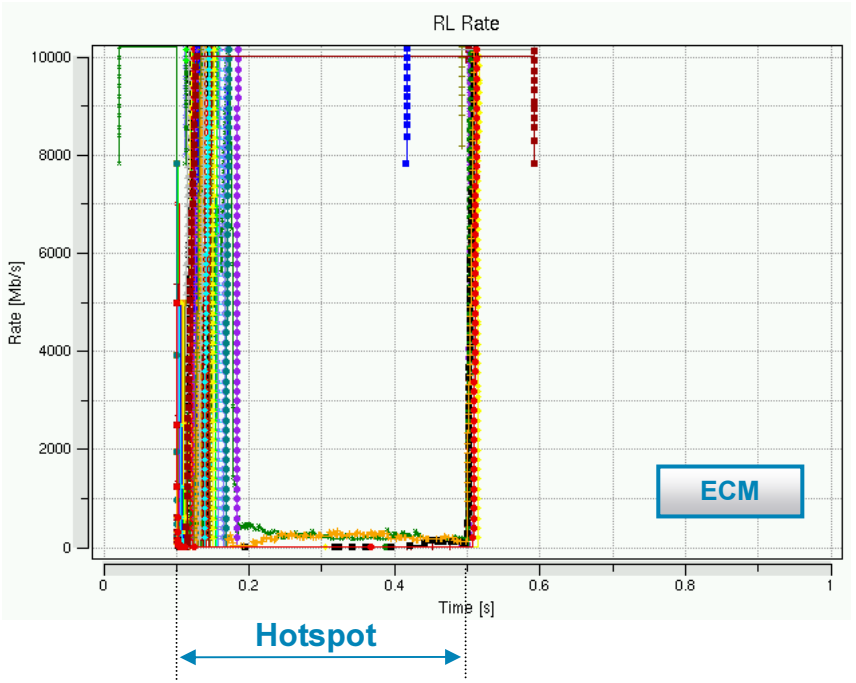
[3] http://www.ieee802.org/1/files/public/docs2007/au-bergamasco-ecm-qcn-benchmarks-20070717.pdf

[4] http://www.ieee802.org/1/files/public/docs2007/au-prabhakar-improving-recovery-v1-1.pdf

QCN Issues

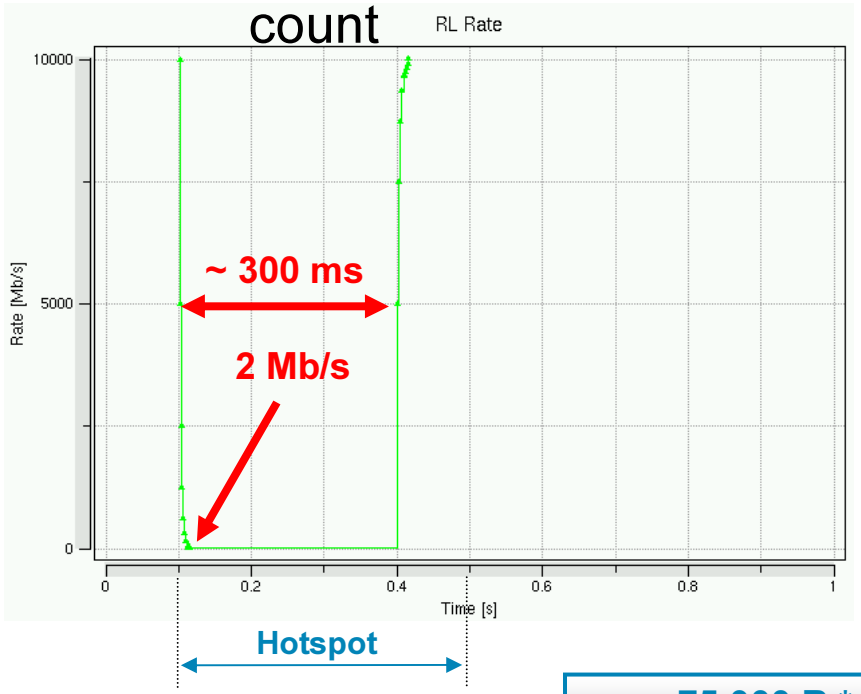
- Why is QCN(^) recovery so slow?
  - Because of lack of positive feedback
  - Self-increase (EFR/AI) is open-loop because driven by byte-count

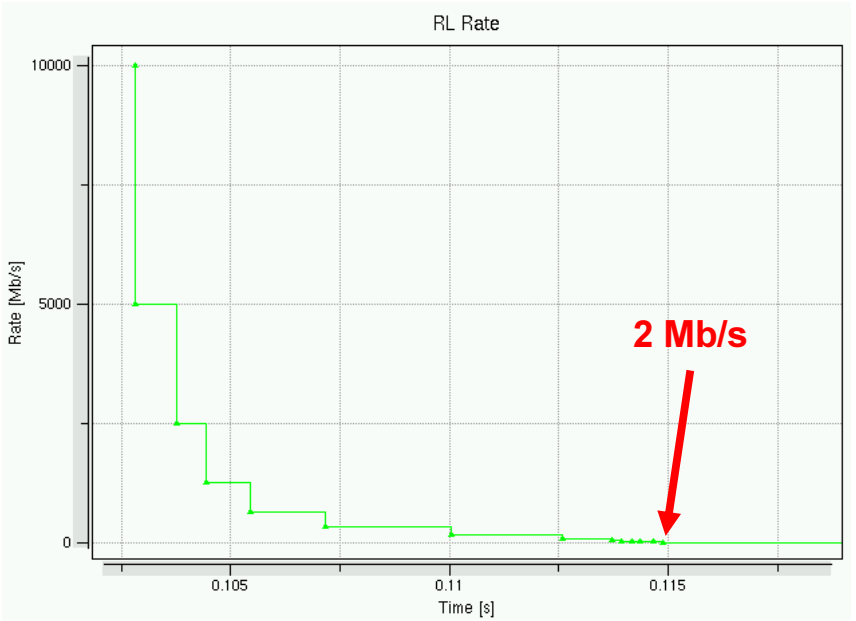




QCN Issues

- Why is QCN(^) recovery so slow?
  - Because of lack of positive feedback
  - Self-increase (EFR/AI) is open-loop because driven by byte-count

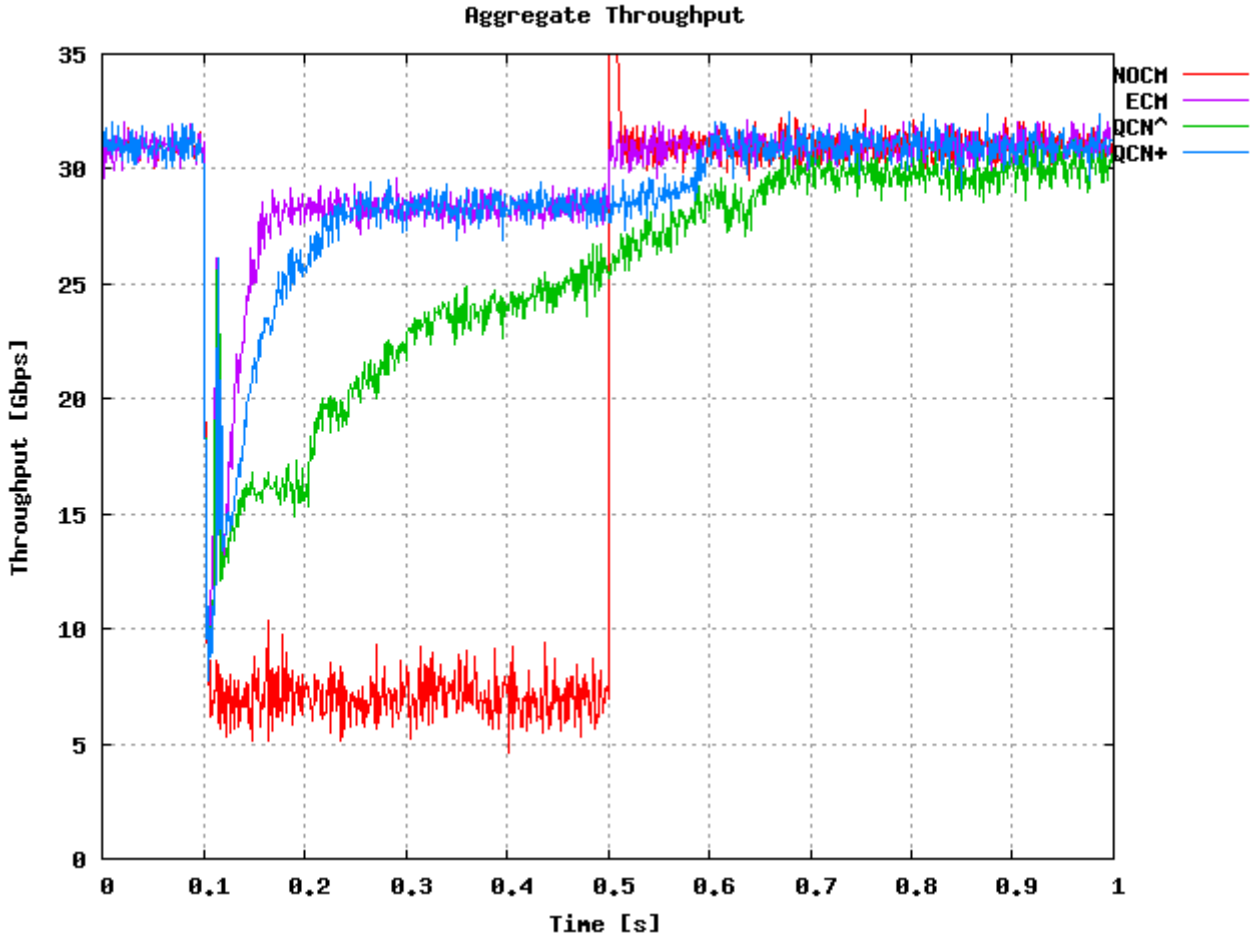




75,000 B * 8 / 2 Mb/s = 0.3 s

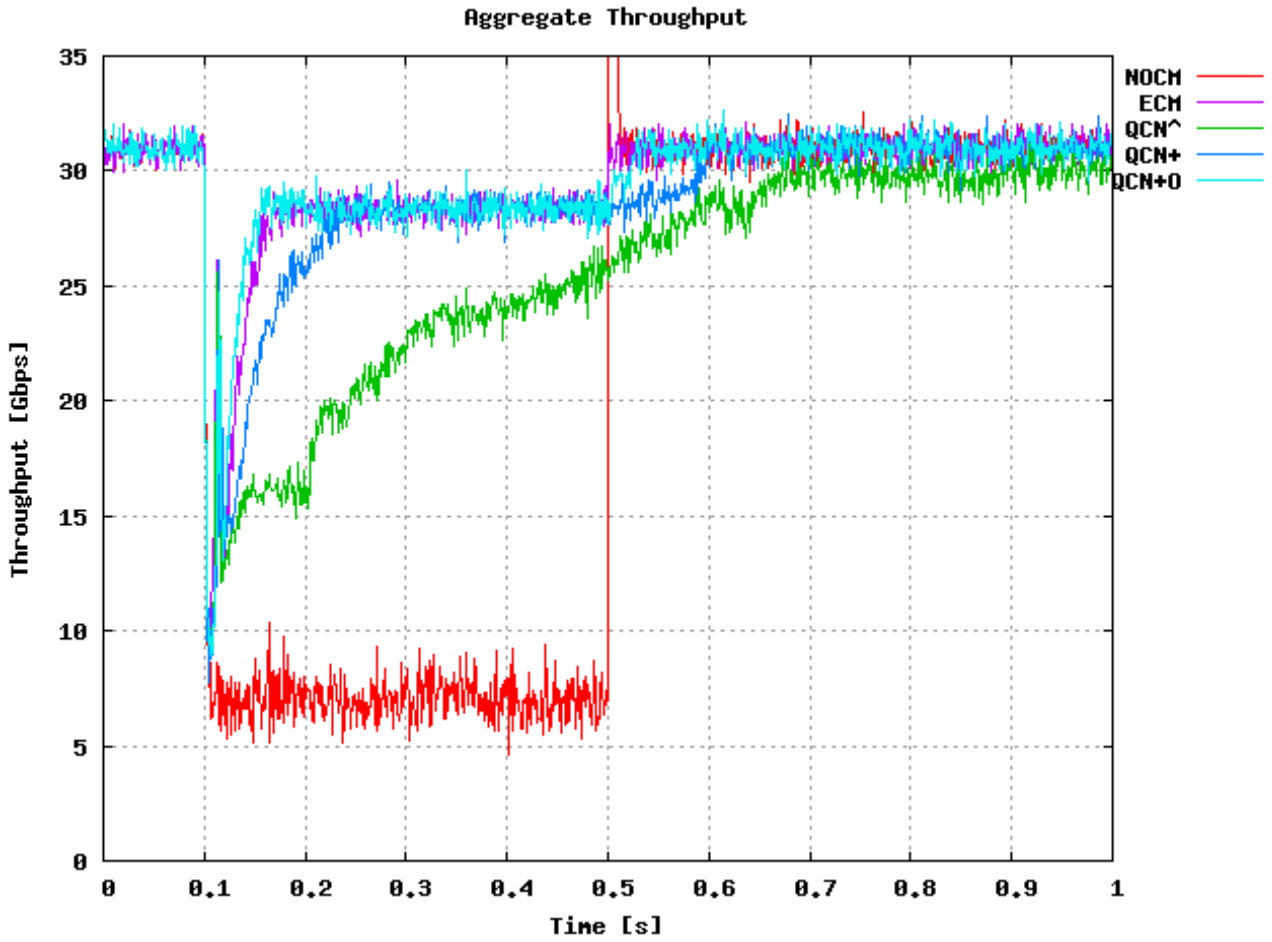
A Possible Solution

- Generate positive feedback (Fb+)
- Use Fb+ to drive the (E)FR/AI recovery phases in lieu of the byte counters



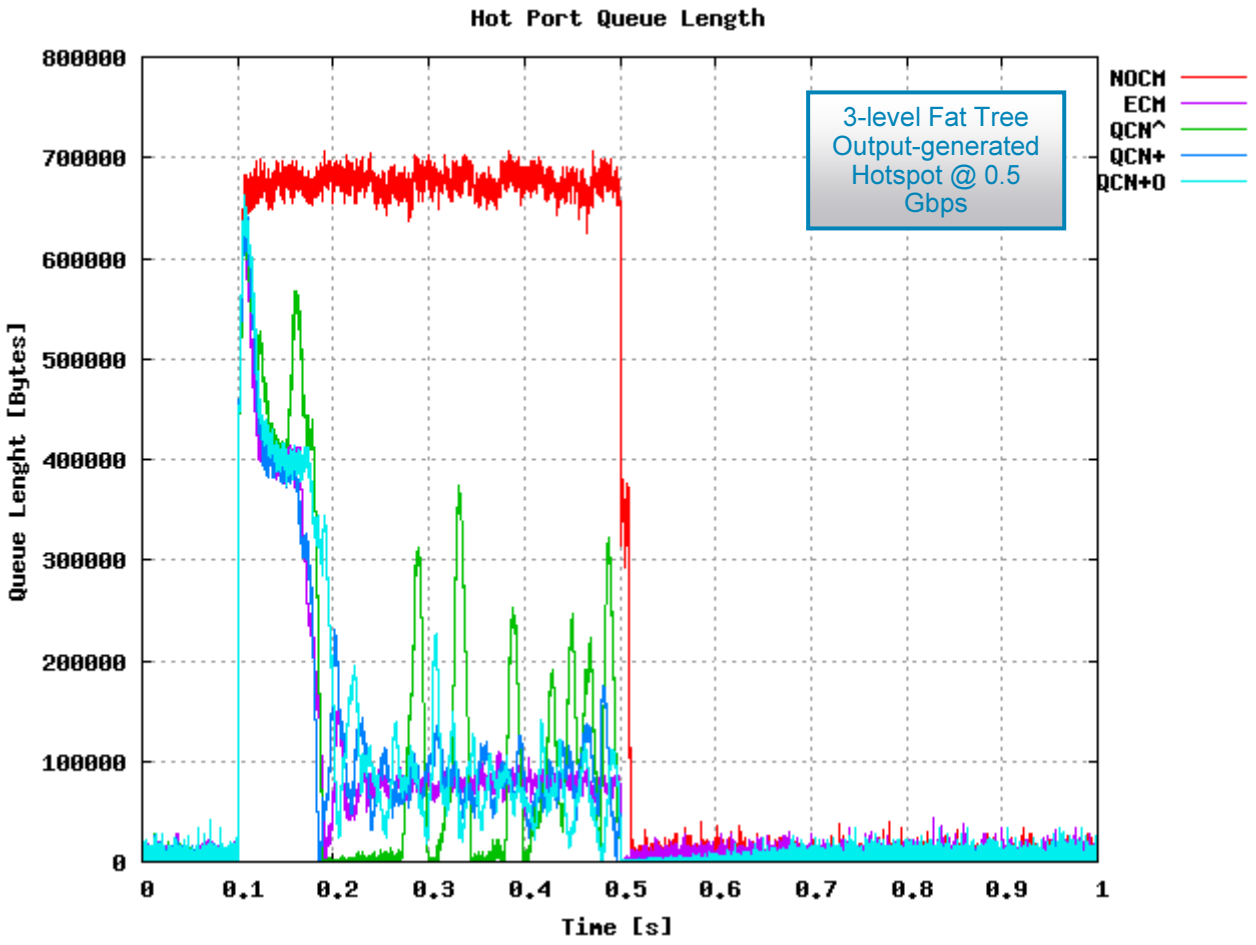
A Possible Solution

- Performance improves even further if ECM-style OverSampling is used



A Possible Solution

- Most importantly, positive feedback brings back stability



A Converged Proposal: QECM

- The suggested solution to QCN issues is really a hybrid of the ECM and QCN mechanisms: **QECM?**
- QECM inherits the advantages of both mechanisms
  - Performance
  - Stability
  - Simplicity
- Proposal
  - .1Qau to consider QECM as the foundation on which to build the Ethernet Congestion Management framework

QECM: The Details

QECM CP Details

- Symmetrical feedback as in ECM

  $Fb = -(q_{off} + w * q_{delta})$

  $-Fb_{Max} \leq Fb < Fb_{Max}$

  $Fb = Fb >> shft$

- Sampling probability proportional to |Fb|
- Over-sampling triggered by queue saturation events as in ECM

  Queue going empty

  Queue going above severe congestion threshold

- Fb > 0 messages generated only on rate-limited frames

  Use DE bit to denote such frames

  Timer used to reduce false positives

- Fb messages contain CPID of originating CP

QECM RP Details

- Fb < 0 messages handled as per QCN definition
  - RP stores CPID of last Fb < 0 message
- Fb > 0 honored only if CPID matches with stored one
  - Trigger transition to the next cycle of recovery (E)FR / AI

QECM CP Pseudocode

```
qecmDetection( queueLen, frame )      /* @ every frame arrival */
{
  qOff = queueLen - qecmQeq;
  qDelta = queueLen - qecmQold;
  fb = -(qOff + qecmW * qDelta);

  if ( fb < 0 || ( fb > 0 && frame.DE && !isExpired(fbTimer) ) )
    {
      /* Fedback saturation & quantization */
      if ( fb > qecmMaxFb )
         fb = qecmMaxFb;
      else if ( fb < -qecmMaxFb || queueLen > qecmQmc )  /* ECM-Max */
         fb = -qecmMaxFb;
      fb >>= qecmShift;

     /* Sampling interval generation */
      if ( queueLen > qecmQmc || queueLen == 0 )  /* ECM-style oversampling */
         p = qecmPmax;
      else
         p = (abs(fb) / qecmMaxQFb) * (qecmPmax - qecmPmin) + qecmPmin;
      qecmSampInt = int( ETH_MTU / p );
      qecmByteCnt += len(frame);

     /* Sampling & QECM frame generation */
      if ( qecmByteCnt >= qecmSampInt )
       {
          qecmFrame.srcMacAddr = SW_MAC_ADDR;
          qecmFrame.dstMacAddr = frame.srcMacAddr;
          qecmFrame.fb = fb;
          qecmFrame.cpId = CPID;
          send( qecmFrame );
          qecmQold = queueLen;
          qecmByteCnt = 0;
          if ( fb < 0 )
               restart(fbTimer);
       }
    }
}
```

QECM RP Pseudocode

```
qecmReaction( qecmFrame )  /* @ every QECM frame arrival */
{
  if ( qecmFrame.fb > 0 && qecmFrame.cpId == cpId  && rlState == RL_ACTIVE )
    {
      qecmSIcnt++;
      if ( qecmState == FAST_RECOVERY )
        {
          qecmRd /= 2;
          rate += qecmRd;
          if ( qecmSIcnt >= qecmFRthr )
              qecmState = ACTIVE_INCREASE;
        }
     else // ( qecmState == ACTIVE_INCREASE )
        {
          if ( qecmHyperActiveIncrease )
              rate += qecmRi * (qecmSIcnt - qecmFRthr );
          else
              rate += qecmRi;
        }

     if ( rate > MAX_RATE )
         rate = MAX_RATE;
    }
  else if (qecmFrame.fb < 0 )
    {
       cpId = qecmFrame.cpId;

       /* Negative feedback handled as per QCN pseudocode */
    }
}


qecmMarking( frame )   /* @ every frame departure from RL */
{
   frame.DE = 1;
}
```