**Edge-to-edge Control: Congestion Avoidance and Service Differentiation for the Internet**

David Harrison  
Rensselaer Polytechnic Institute  
harrisod@cs.rpi.edu  
http://networks.ecse.rpi.edu/~harrisod

---

**Outline**

- QoS for Multi-Provider Private Networks
- Edge-to-Edge Control Architecture
- Riviera Congestion Avoidance
- Trunk Service Building Blocks
  - Weighted Sharing
  - Guaranteed Bandwidth
  - Assured Bandwidth

---

**QoS for Multi-Provider Private Networks**

- **Principle Problems**
  - **Coordination:** scheduled upgrades, cross-provider agreements
  - **Scale:** thousands-millions connections, Gbps.
  - **Heterogeneity:** many datalink layers, 48kbps to >10Gbps

---

**Single Vs. Multi-Provider Solutions**

- ATM and frame relay operate on single datalink layer.
- All intermediate providers must agree on a common infrastructure. Requires upgrades throughout the network. **Coordination** to eliminate heterogeneity.
- Or operate at lowest common denominator.
- **Overprovision:**
  - Operate at single digit utilization.
  - More bandwidth than sum of access points.
  - 1700 DSL (at 1.5 Mbps) or 60 T3 (at 45 Mbps) DDoS swamps an OC-48 (2.4 Gbps).
  - Peering points often last upgraded in each upgrade cycle. Performance between MY customers more important.
- **Hard for multi-provider scenarios.**

---

**Scalability Issues**

- Traditional solutions:
  - Use QoS:
    - ATM, IntServ: per-flow/per-VC scheduling at every hop.
    - Frame Relay: Drop preference, per-VC routing at every hop.
    - DiffServ: per-class (eg: high, low priority) scheduling, drop preference at every hop. Per-flow QoS done only at network boundaries (edges).

---

**Edge-to-Edge Control (EC)**

Use Edge-to-edge congestion Control to push queuing, packet loss and per-flow bandwidth sharing issues to edges (e.g. access router) of the network.
**QoS via Edge-to-Edge Congestion Control**

- **Benefits:**
  - Conquers scale and heterogeneity in same sense as TCP.
  - Allows QoS without upgrades to either end-systems or intermediate networks.
  - Only incremental upgrade of edges (e.g., customer premise access point).
  - Bottleneck is CoS FIFO.
  - Edge knows congestion state and can apply stateful QoS mechanisms.

- **Drawbacks:**
  - Congestion control cannot react faster then propagation delay.
  - Loose control of delay and delay variance.
  - Only appropriate for data and streaming (non-live) multimedia.
  - Must configure edges and potential bottlenecks.

---

**Riviera Congestion Avoidance**

- Implements EC Traffic Trunks.
- EC Constraints:
  - Cannot assume access to TCP headers.
  - No new fields in IP headers (no sequence numbers)
  - Cannot assume existence of end-to-end ACKs (e.g., UDP)
  - Cannot impose edge-to-edge ACKs (doubles packets on network)
- → No window-based control.
- Solution: rate-based control.

---

**Mechanisms for Fairness and Bounded Queue**

- Estimate this control loop’s backlog in path.
  - If backlog > max_thres
    - Congestion = true
  - Else if backlog <= min_thres
    - Congestion = false

- All control loops try to maintain between min_thres and max_thres backlog in path.
  - → bounded queue (Goal 4)

- Each control loop has roughly equal backlog in path → proportional fairness [Low] (Goal 5)

- Well come back to goal 5.

---

**Backlog Estimation and Goal 2**

- Use basertt like Vegas backlog estimation.
- As with Vegas, when basertt is wrong → gross unfairness (violates Goal 2).
- Sol’n: ensure good basertt estimate.

---

**Congestion Avoidance Goals**

- 1. Avoid of congestion collapse or persistent loss.
- 4. High utilization when demand.
- 5. Bounded queue.
- Zero loss with sufficient buffer.
- Accumulation.

---

**Vegas & Delay Increase (Goal 2)**

- Vegas sets basertt to the minimum RTT seen so far.
  - → GROSS UNFAIRNESS!
**Riviera Round-trip Propagation Delay (RTPD) Estimation (Goal 2)**

- Reduce gross unfairness w/ good RTPD estimation.
- Minimum of last \(k\) control packet RTTs.
- Drain queues in path so RTT in last \(k\) RTTs likely reflects RTPD.
- Set \(\text{max\_thresh}\) high enough to avoid excessive false positives.
- Set \(\text{min\_thresh}\) low enough to ensure queue drain.
- Provision drain capacity with each decrease step.

**Increase/Decrease Policy to Drain Queue (Goal 2)**

\[
\begin{align*}
\lambda_i & \rightarrow \sum \lambda_i \rightarrow v_i \\
\lambda_m & \rightarrow \nu_i \\
\end{align*}
\]

- \(r_i\) = rate limit on leaky bucket (\(\sigma, \rho\)) shaper. \(\dot{\lambda}_i = r_i\)
- Increase/decrease Policy
  \[
  \begin{cases} 
  v_i + \text{MTU}/\text{RTT} & \text{if no congestion} \\
  \beta v_i & \text{if congestion} \\
  \end{cases}
  \]
  \(1 > \beta >> 0\)
- Lower \(\beta\) improves probability queues drain at cost to utilization.

**Riviera Achieves Proportional Fairness? (Goal 5)**

\[
\begin{align*}
\max \sum \log \lambda_i \quad \text{with} \quad \sum \lambda_i \leq C, \quad i \in L, \quad \lambda_i \geq 0
\end{align*}
\]

**Weighted Proportional Fairness**

\[
\max \sum w_i \log \lambda_i
\]
Weighted Service Building Block

- Modify accumulation thresholds:
  - \( \text{max\_thresh}_i = w_i \times \text{max\_thresh}_i \)
  - \( \text{min\_thresh}_i = w_i \times \text{min\_thresh}_i \)

Weighted Service Building Block (2)

Guaranteed Bandwidth Allocation

- Maximize: \( \sum w_i \log(\lambda_i - g_i) \)

Quasi-Leased Line (QLL)

- Converges on guaranteed bandwidth allocation.

- Accumulation Modification:
  - Apply Little’s Law
  - If no congestion:
    - \( q_i = \lambda_i t = v_i t \)
    - \( \lambda_i = \lambda \eta_i t = v_i t \)
  - If congestion:
    - \( q_i = \frac{d_{\text{in}} - v_i}{g_i} \)
    - \( g_i = q_i + \frac{v_i}{\eta_i} \)

QLL Increase/Decrease Policy

- Increase/decrease policy:
  - \( r_i = \max(g_i, v_i + \text{MTU/RTT}) \) if no congestion
  - \( r_i = \max(g_i, \beta(v_i - g_i) + g_i) \) if congestion

Quasi-Leased Line Example

- Best-effort VL starts at \( t=0 \) and fully utilizes 100 Mbps bottleneck.

- Background QLL starts with rate 50 Mbps

- Best-effort rate limit versus time

- Background QLL quickly adapts to new rate.
Quasi-Leased Line Example (cont.)

Bottleneck queue versus time

Starting QLL incurs backlog.

Unlike TCP, VL traffic trunks backoff without requiring loss and without bottleneck assistance.

Requires more buffers: larger max queue

Assured Building Block

- Accumulation:
  \[ q_\text{a} = q \left( 1 - \frac{\beta}{\tau} \right) \]
- if \( q_\text{b} > \text{max}_\text{thresh} \) and \( q > w_\text{MAX} \cdot \text{max}_\text{thresh} \)
  congestion = true
- else if \( q_\text{b} < \text{min}_\text{thresh} \) and \( q <= w_\text{MAX} \cdot \text{max}_\text{thresh} \)
  congestion = false

- Increase/Decrease Policy:
  \[ r = \begin{cases} 
    V \left( \beta \frac{\text{MTU}}{\tau} \right) & \text{if no congestion} \\
    \min \left( \beta \frac{\text{MTU}}{\tau} \right) \cdot \frac{1}{\beta} \cdot \text{rtt} & \text{if congestion} \\
    1 > \beta \frac{\text{MTU}}{\tau} >> 0 
  \end{cases} \]

Backoff little \( (\beta_\text{AS}) \) when below assurance \( (a) \),
Backoff \( (\beta_\text{BE}) \) same as best effort when above assurance \( (a) \)

Quasi-Leased Line (cont.)

Single bottleneck queue length analysis:

- \[ q < \frac{b}{b-1} \]
- \( B/\text{w-RTT} \) products

For \( b=5 \), \( q=1 \text{ bw-rtt} \)

Simulated QLL w/ Riviera.

Assured Building Block Vs. Assured Allocation

Wide Range of Assurances
Large Assurances

![Graph showing target throughput vs. actual throughput]

Summary

- Issues:
  - Simplified overlay QoS architecture
  - Intangibles: deployment, configuration advantages
- Edge-based Building Blocks & Overlay services:
  - A closed-loop QoS building block
  - Weighted services, Assured services, Quasi-leased lines

Private Versus Public Networks

- Peering point, International link
- Private Networks:
  - Tens or hundreds of communicating parties
  - Limits per-loop state and configuration at edges.
  - Parties remain in VPN for moderate to long term
  - Amortize configuration over lifespan of VPN.
  - Aggregation
  - Amortize control packet overhead.
- Public Internet:
  - None of these assumptions hold.

Edge-to-Edge Queue Management

- Queue distribution to the edges => can manage more effectively
- w/o Edge-to-Edge Control
- w Edge-to-Edge Control

Distributed Buffer Management (1)

- Implement FRED AQM at edge rather than at bottleneck. Bottleneck remains FIFO.
- Versus FRED at bottleneck and NO edge-to-edge control.

Distributed Buffer Management (2)

- Coefficient of variation vs. Go steam and number of flows.
TCP Rate Control (Near Zero Loss)
- Use Edge-to-edge Control to push bottleneck back to edge.
- Implement TCP rate control at edge rather than at bottleneck.
  Bottleneck remains FIFO.

Remote Bottleneck Bandwidth Management
- Edge redistributes VL’s fair share between end-to-end flows.

TCP Rate Control (2)
Coefficient of Variation in Goodput vs. 10 to 1000 TCP flows
- FRED bottleneck
- FIFO bottleneck
- 2, 5, 10 TCPR Edges

Remote Bandwidth Management (2)
- TCP 0 with weight 3. obtains 3/4 of VL 0
- TCP 1 with weight 1 obtains 1/4 of VL 0

TCP Rate Control (3)
- FRED bottleneck
- FIFO bottleneck
- 2, 5, 10 TCPR Edges
- ZERO LOSS

UDP Congestion Control, Isolate Denial of Service
- UDP source floods networks
- TCP source Ingress 10 Mbps Egress TCP dest
- UDP source 1 1 UDP dest
UDP Congestion Control, Isolate Denial of Service

Trunk 0 carries TCP starting at 0.0s
Trunk 1 carries UDP flood starting at 5.0s

Effects: Bandwidth Assurances

TCP with 4 Mbps assured + 3 Mbps best effort
UDP with 3 Mbps best effort