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

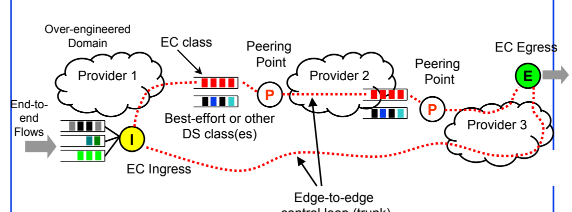
Single Vs. Multi-Provider Solutions

- 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.*

Congestion Avoidance Goals

- 1. Avoid of congestion collapse or persistent loss.
  - Behave like TCP Reno in response to loss.
- **2. Avoid starvation and gross unfairness.**
  - Isolate from best effort traffic.
  - Solve Vegas RTPD estimation errors.
- 3. High utilization when demand.
- **4. Bounded queue.**
  - ***Zero*** loss with sufficient buffer.
  - Accumulation.
- **5. Proportional fairness.**
- …
- **Attack goals 2,4, and 5 in reverse order.**

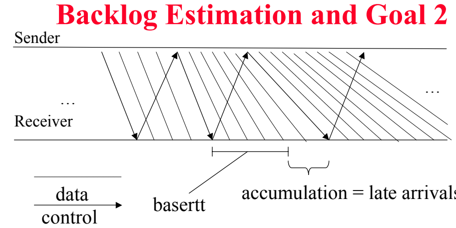
Mechanisms for Fairness and Bounded Queue

- Estimate this control loop's backlog in path.

  If backlog > max_thresh
      Congestion = true
  Else if backlog <= min_thresh
      Congestion = false

- All control loops try to maintain between min_thresh and max_thresh 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.

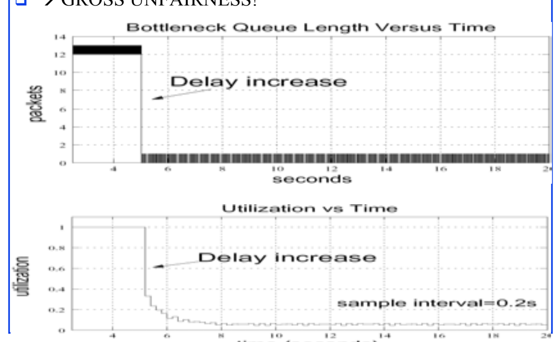
Backlog Estimation and Goal 2



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

Vegas & Delay Increase (Goal 2)

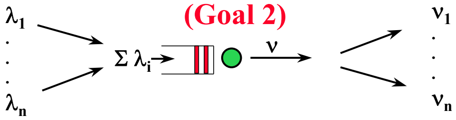
- Vegas sets basertt to the minimum RTT seen so far.
- → GROSS UNFAIRNESS!



Riviera Round-trip Propagation Delay (RTPD) Estimation (Goal 2)

- Reduce gross unfairness w/ good RTPD estimation.
- Minimum of last k=30 control packet RTTs.
- Drain queues in path so RTT in last k RTTs likely reflects RTPD.
  - Set max_thresh high enough to avoid excessive false positives.
  - Set min_thresh low enough to ensure queue drain.
- Provision drain capacity with each decrease step

Increase/Decrease Policy to Drain Queue (Goal 2)



r_i = rate limit on leaky bucket (σ,ρ) shaper. $\lambda_i \le r_i$

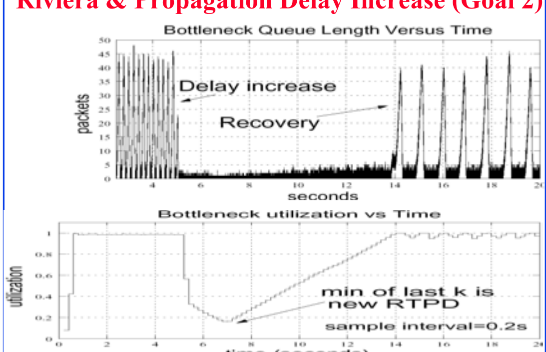
Increase/decrease Policy

$$r_i = \begin{cases} \nu_i + MTU/RTT & \text{if no congestion} \\ \beta \nu_i & \text{if congestion} \end{cases}$$

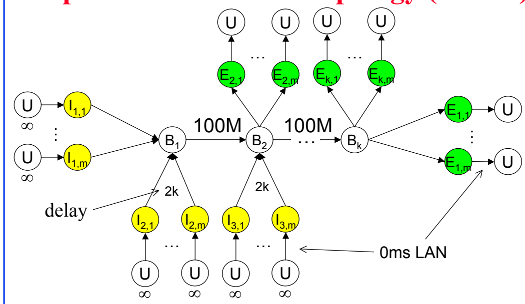
$$1 > \beta >> 0$$

Lower β improves probability queues drain at cost to utilization.

Riviera & Propagation Delay Increase (Goal 2)

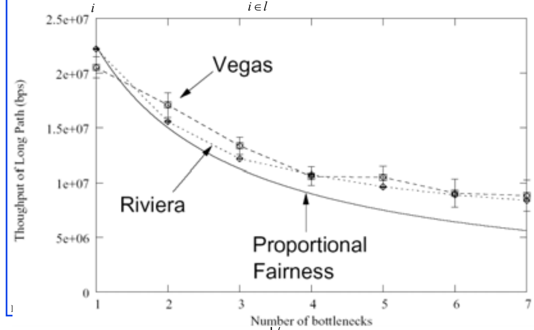


Proportional Fairness Topology (Goal 5)



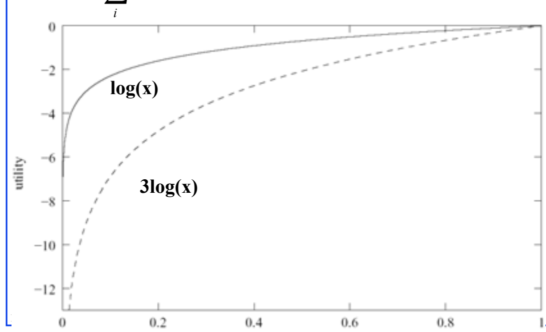
Riviera Achieves Proportional Fairness? (Goal 5)

$$\max \sum_i \log \lambda_i \quad \textbf{with} \quad \sum_{i \in l} \lambda_i \le C_l, \; l \in L, \text{ and } \lambda_i \ge 0$$



Weighted Proportional Fairness

$$\max \sum_i w_i \log \lambda_i$$

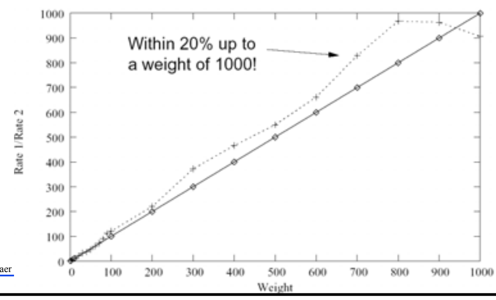


Weighted Service Building Block

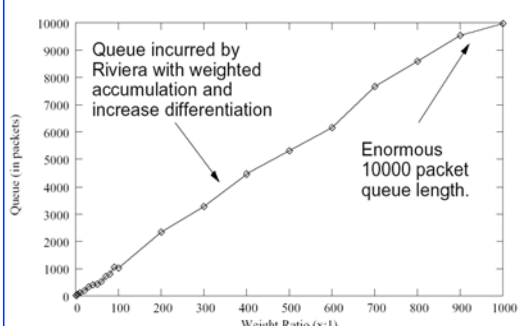
- Modify accumulation thresholds:

  $max\_thresh_i = w_i * max\_thresh_i$

  $min\_thresh_i = w_i * min\_thresh_i$

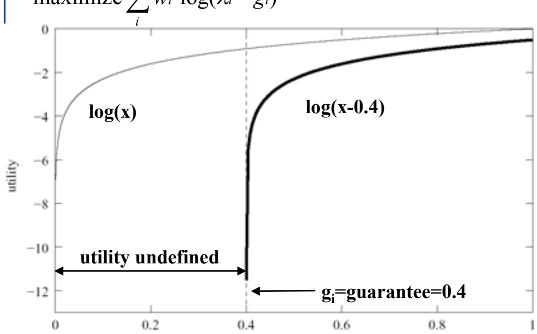


Weighted Service Building Block (2)



Guaranteed Bandwidth Allocation

$$\text{maximize} \sum_i w_i \log(\lambda_i - g_i)$$



Quasi-Leased Line (QLL)

- Converges on guaranteed bandwidth allocation.
- **Accumulation Modification:**

  Apply Little's Law

  $\lambda_i \rightarrow \lambda_{ig}, \lambda_{ib} \rightarrow$ [t] (queue) $\rightarrow \nu_{ig} = g_i, \nu_{ib} \rightarrow \nu_i$

  $q_i = \lambda_i t = \nu_i t$

  $q_{ig} = \lambda_{ig} t = \nu_{ig} t$

  $\frac{q_i}{q_{ig}} = \frac{\nu_i}{g_i}$, $q_i = q_{ig} + q_{ib}$ $\rightarrow$ $q_{ib} \approx q_i \left(1 - \frac{g_i}{\nu_i}\right)$ ← **All these variables known**

  if (q_{ib} > max_thresh_i) congestion = true

  if (q_{ib} <= min_thresh_i) congestion = false

QLL Increase/Decrease Policy

- **Increase/decrease policy:**

$$r_i = \begin{cases} \max(g_i, \nu_i + MTU/RTT) & \textbf{if no congestion} \\ \max(g_i, \beta(\nu_i - g_i) + g_i) & \textbf{if congestion} \end{cases}$$

$1 > \beta >> 0$

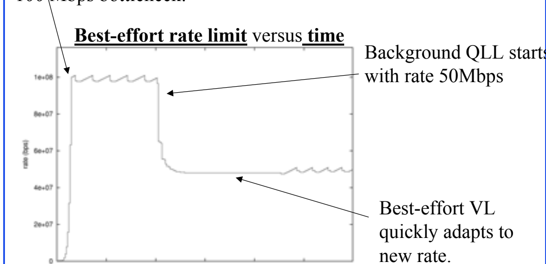
Go immediately to guarantee and refuse to go below.

Decrease based only on the rate that is above the guarantee

- No admission control → unbounded queue.

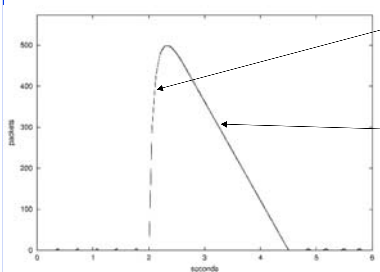
Quasi-Leased Line Example

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



Quasi-Leased Line Example (cont.)

Bottleneck queue versus **time**



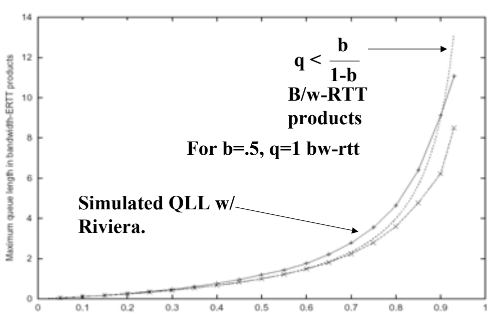
Starting QLL incurs backlog.

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

Requires more buffers: larger max queue

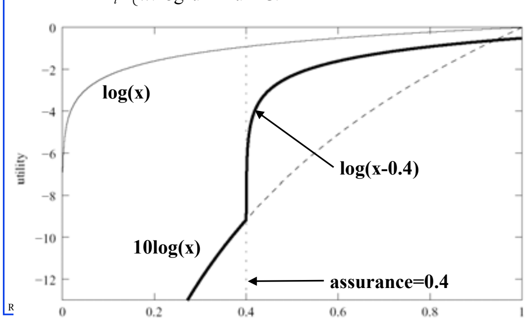
Quasi-Leased Line (cont.)

Single bottleneck queue length analysis:



Assured Bandwidth Allocation

$$\text{maximize} \sum_i \begin{cases} \log(\lambda_i - a_i) & \text{if } \lambda_i > C_i \\ w_i \log \lambda_i & \text{if } \lambda_i \le C_i \end{cases} \quad \textbf{with} \quad \log(C_i - a_i) = w_i \log C_i$$



Assured Building Block

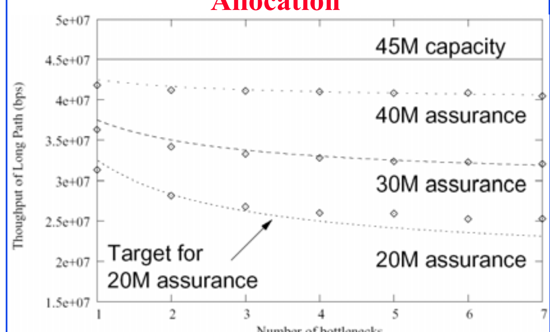
- **Accumulation:** $q_{ib} \approx q_i\left(1 - \frac{a_i}{v_i}\right)$
  - if (q_{ib} > max_thresh || q_i > w_i * max_thresh)
    congestion = true
    else if (q_{ib} <= min_thresh && q_i <= w_i * max_thresh)
    congestion = false
- **Increase/Decrease Policy**:

$$r_i = \begin{cases} v_i + MTU/RTT & \textbf{if no congestion} \\ \min(\beta_{AS}\, v_i,\ \beta_{BE}(v_i - a_i) + a_i) & \textbf{if congestion} \end{cases}$$

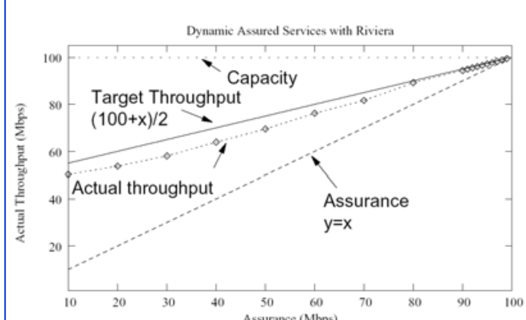
$$1 > \beta_{AS} > \beta_{BE} >> 0$$

- Backoff little (β_{as}) when below assurance (a),
- Backoff (β_{be}) same as best effort when above assurance (a)

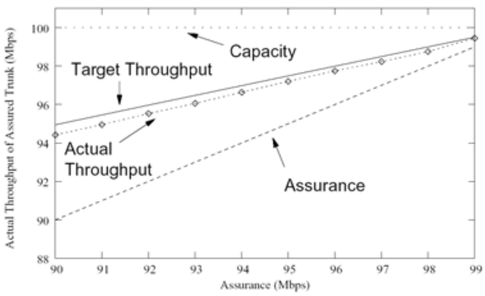
Assured Building Block Vs. Assured Allocation



Wide Range of Assurances



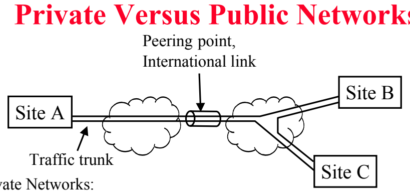
Large Assurances



Summary

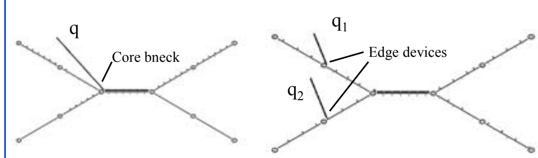
- ***Issues:***
  - Simplified overlay QoS architecture
    - <u>Intangibles:</u> deployment, configuration advantages
- ***Edge-based Building Blocks & Overlay services***:
  - A ***<u>closed-loop</u>*** QoS building block
  - Weighted services, ***Assured*** services, ***Quasi***-leased lines

Private Versus Public Networks



- 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.

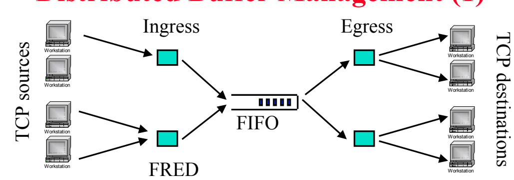
Edge-to-Edge Queue Management



w/o Edge-to-Edge Control w Edge-to-Edge Control

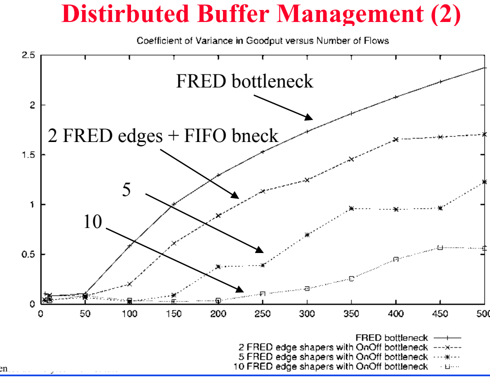
Queue distribution to the edges => can manage more effectively

Distributed Buffer Management (1)

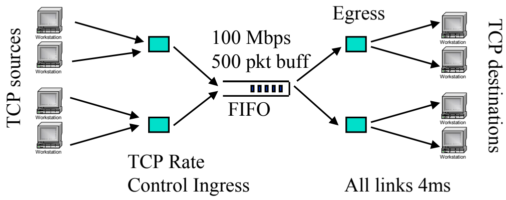


- Implement FRED AQM at edge rather than at bottleneck. Bottleneck remains FIFO.
- Versus FRED at bottleneck and NO edge-to-edge control.

Distirbuted Buffer Management (2)



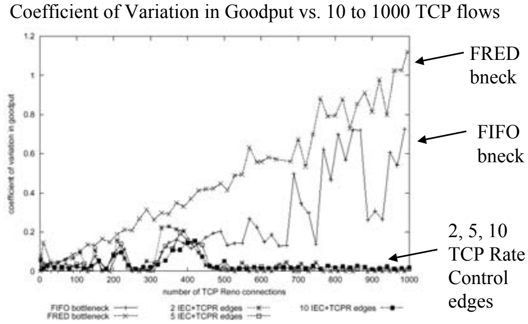
TCP Rate Control (Near Zero Loss)



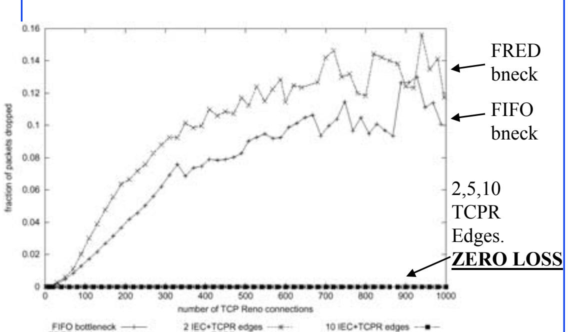
- Use Edge-to-edge Control to push bottleneck back to edge.
- Implement TCP rate control at edge rather than at bottleneck. Bottleneck remains FIFO.

TCP Rate Control (2)

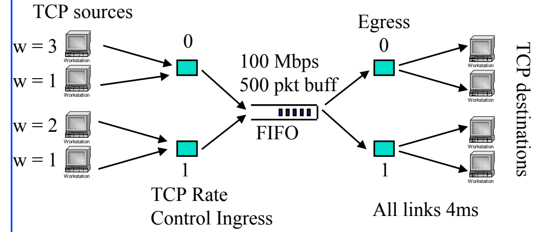
Coefficient of Variation in Goodput vs. 10 to 1000 TCP flows



TCP Rate Control (3)

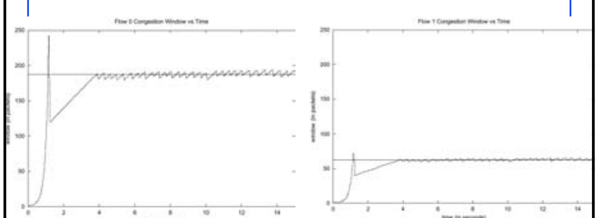


Remote Bottleneck Bandwidth Management



- Edge redistributes VL's fair share between end-to-end flows.

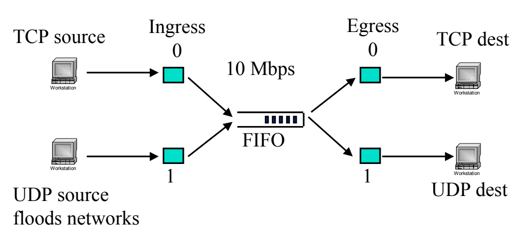
Remote Bandwidth Management (2)



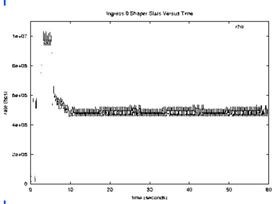
TCP 0 with weight 3. obtains 3/4 of VL 0

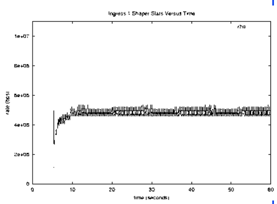
TCP 1 with weight 1 obtains 1/4 of VL 0

UDP Congestion Control, Isolate Denial of Service



UDP Congestion Control, Isolate Denial of Service

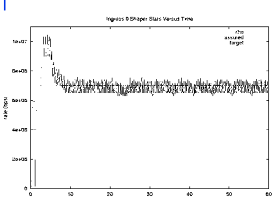


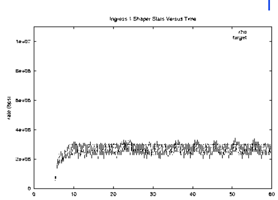


Trunk 0 carries TCP starting at 0.0s

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