Improvements to and Performance of Conservative Mode for Single Choke Fairness Mechanism Specified in Draft 2.0

Bob Doverspike, Chuck Kalmanek, Jorge Pastor, K. K. Ramakrishnan, Aleksandra Smiljanic, Dong-Mei Wang, John Wei

AT&T Labs. Research, NJ

(Acknowledgment: thanks to Ed Knightly and P. Yuan of Rice University for sharing ns-2 based RPR simulator)

Introduction

- At the November IEEE 802.17 meeting, we presented our observations on the single choke aggressive mode for TCP.
- In certain scenarios, performance showed short term unfairness and oscillations
- Primary reason for oscillatory behavior was a downstream node’s low add rate
  - Upstream node limited to unfair and unreasonably low rate over short term
- We speculated that computing “fair rate” based on the stations sharing bottleneck link, and communicating this fair rate to upstream nodes would:
  - Allow upstream node’s rate to be determined by the actual link bandwidth available
  - Avoid starvation (complete shut-down) of sources for brief periods of time
  - Mitigate oscillations and potentially improve link utilization
- Conservative mode appeared to have these properties
- We focused on the performance of dual transit buffer, conservative scheme
  - Performance of existing specification, with simplifications - when STQ present
  - Determined several small, but critical, improvements were needed
  - Significant improvement in performance observed with our suggested modifications

Single Choke/Conservative scheme

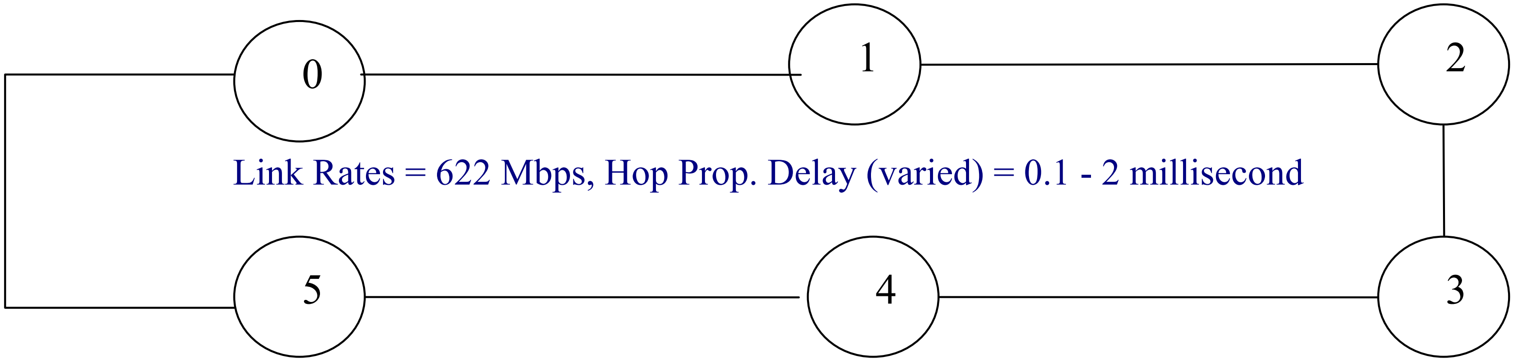
Our high level understanding of existing single choke conservative scheme

- Differences between aggressive and conservative schemes
  - Primarily in congestion detection and local fair rate calculation
- When a span is "congested", backpressure mechanism using Fairness Control Messages (FCM) is triggered
  - STQ buffer occupancy rises above "low threshold" *OR* (filtered) Transmit rate above "low threshold" ⇒ "Congested"
- Fair rate calculation is based on active stations in the "congestion domain"
- Hysteresis in STQ length used to increase/decrease local Fair Rate
  - Congested node modifies local Fair Rate **every round-trip time**:
    - Increase when STQ is below "low threshold", based on ramp coefficient
    - Decrease when STQ is above "high threshold", based on ramp coefficient
  - Maintain local Fair rate as is when STQ is in-between low and high thresholds
- While congested, the node sends local Fair Rate in FCM every "advt. interval"
- When congestion clears, upstream nodes allowed to ramp up to "unreserved rate"
  - STQ buffer sufficient to receive packets in transit, to accommodate feedback delay

Desirable Goals for Scheme

- ☐ Achieve high utilization on the ring – fairness eligible traffic uses available bandwidth opportunistically
  - Allow sources to start up at high rate
- ☐ Converge to fair value as quickly as possible
  - Should achieve fairness on per-source basis with single bottleneck
  - Preferable if unused bandwidth at bottleneck is shared by other sources
- ☐ Operate over a wide dynamic range
  - Even nodes with small STQ buffer still function reasonably over a large ring
- ☐ Minimize oscillations in throughput as much as possible
  - Reflected by the “allowed rate” at the RPR level
  - Reflected at application level by throughput and other specific parameters (e.g., window size for TCP)
- ☐ Avoid even short-term starvation of individual nodes caused by congestion control actions

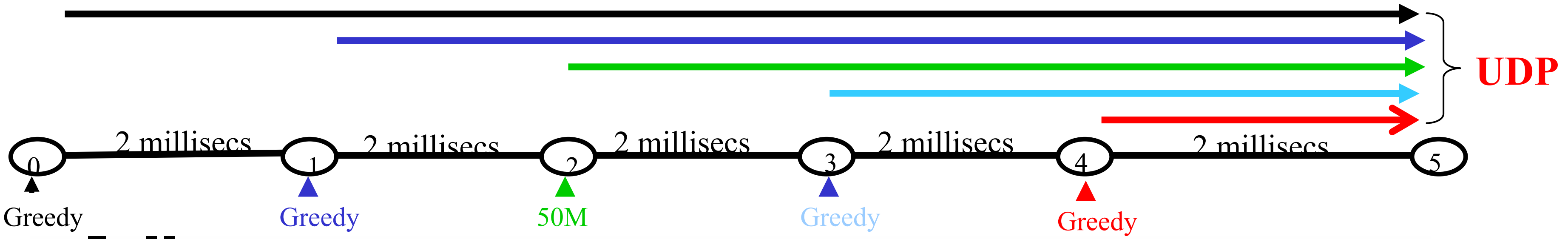
Experimental Framework

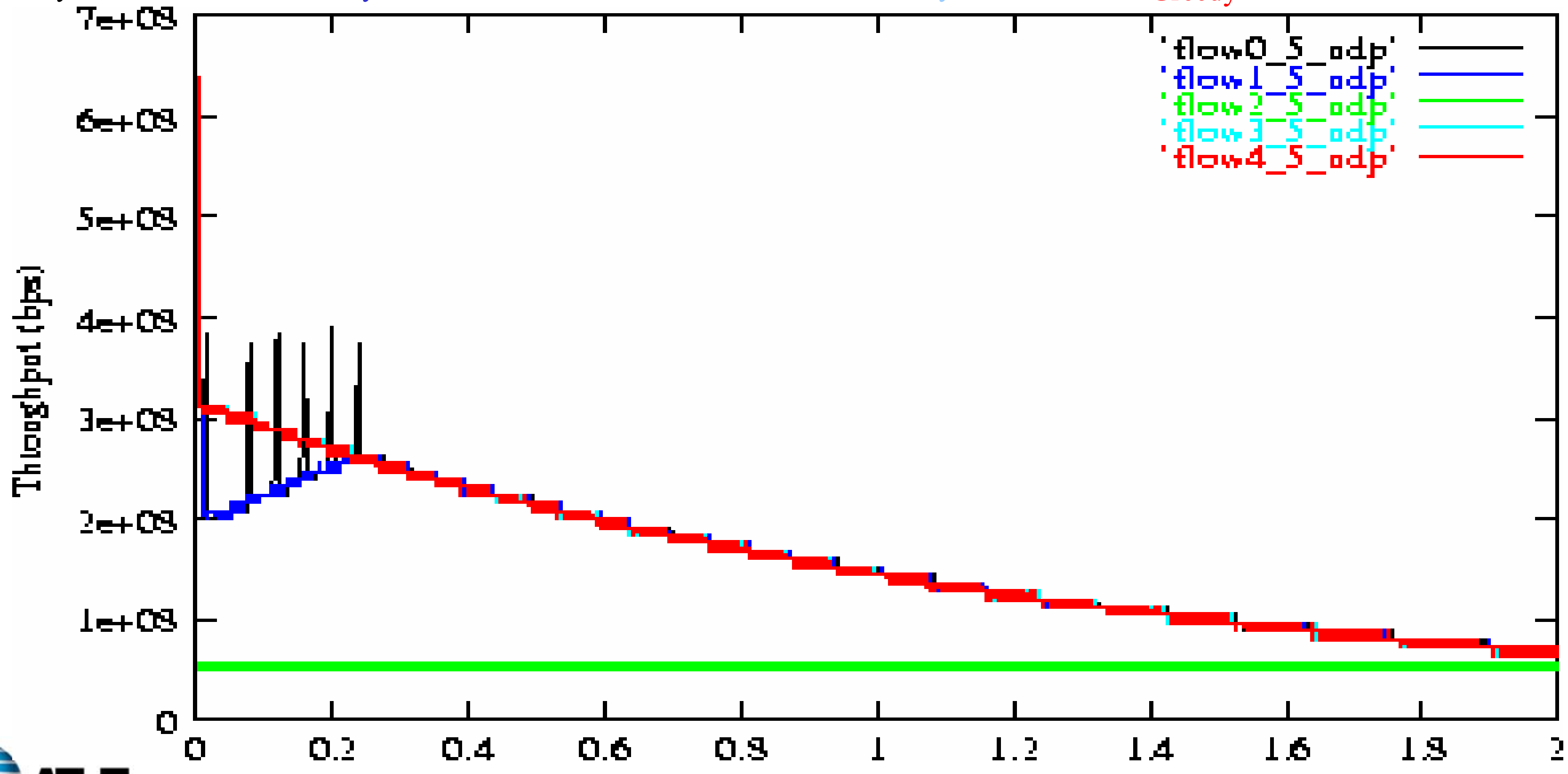

0
1
2
3
4
5
Link Rates = 622 Mbps, Hop Prop. Delay (varied) = 0.1 - 2 millisecond

- ❑ Simulations based on NS-2 simulator (initial version from Rice University)
- ❑ Most of simulation results presented here based on a 6 node ring
  - ➢ Link rates = 622 Mbps; prop. delay = 0.1-2 millisecs; AgingInterval=200 µsecs; Advt. Interval = 100 µsecs.
  - ➢ STQ = 256 Kbytes; Client buffer = 1000 packets (pkt. size = 1040 bytes)
  - ➢ Single congestion domain, with one link being the bottleneck
- ❑ Experiments:
  - ➢ Several experiments with UDP (constant rate) flows
  - ➢ Steady (greedy) TCP flows (FTP); max. window size =64
  - ➢ Mixture of TCP and UDP flows
  - ➢ Start/stop (short-lived) flows: shows scheme's dynamics and responsiveness

Existing (Draft 2.0) Conservative Mode Performance

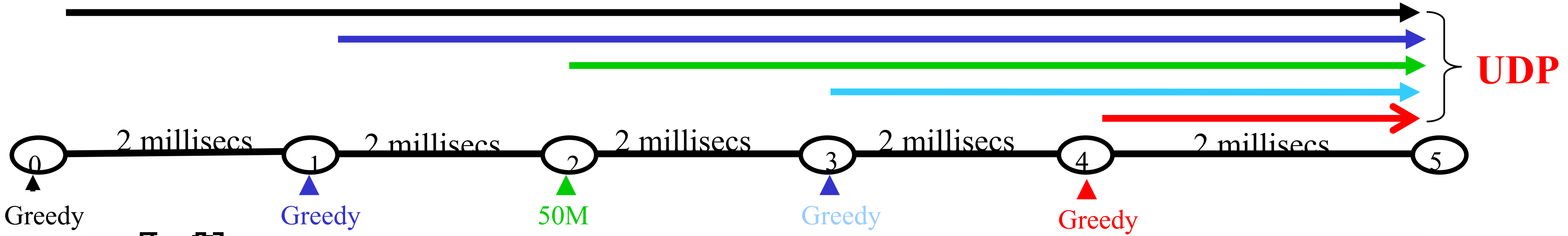
- Simulation: constant rate (UDP) sources; Default parameters; fixed "RTT" = 20msecs.
  - Did not include rate thresholds

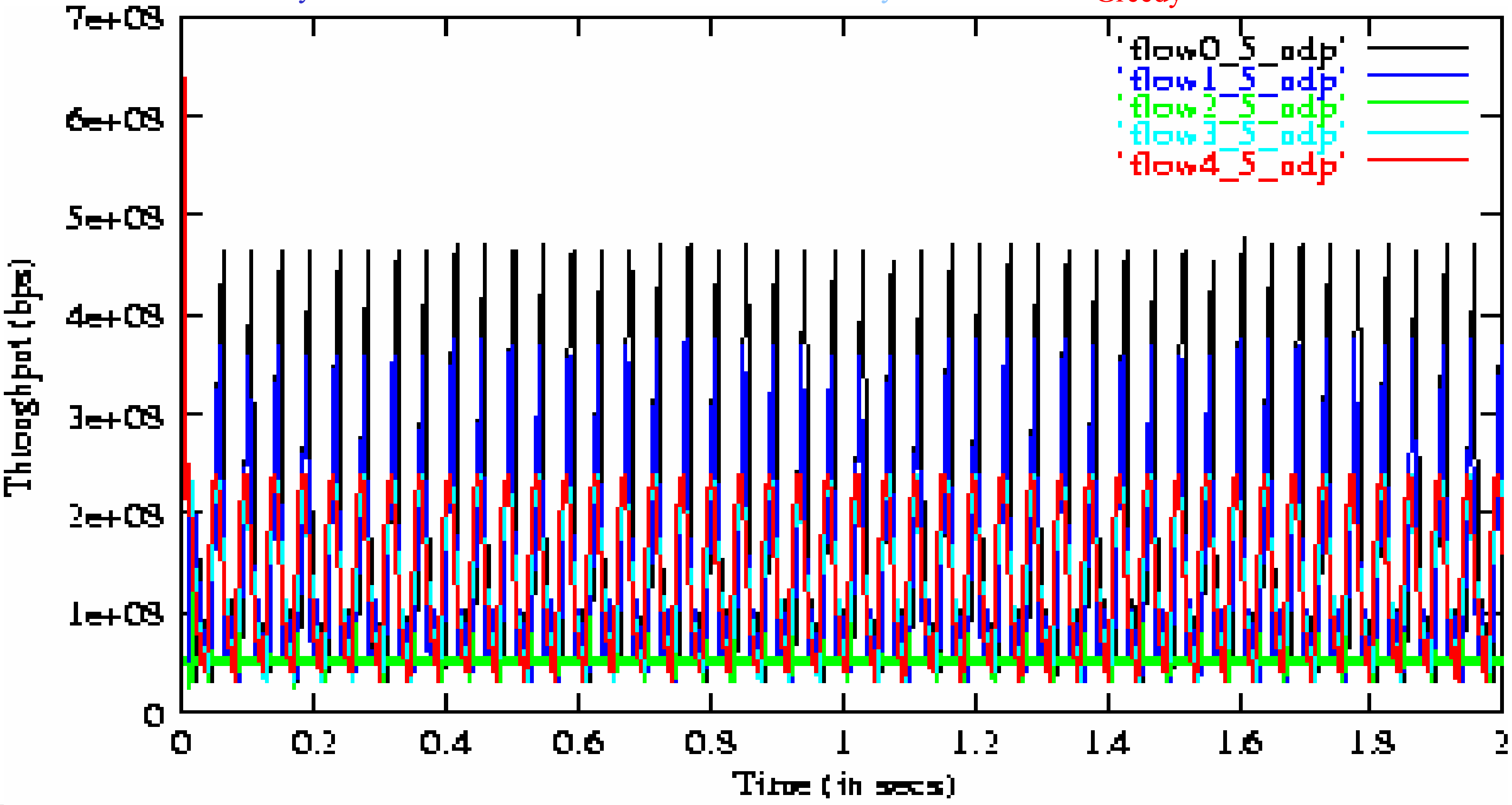




Existing Aggressive Mode Performance

- ☐ Same experiment as previous slide, with 5 UDP flows, 2 millisec. Per-hop propagation delay





Outline of Proposed Improvements

- We have maintained the basic framework of conservative scheme as defined in Draft 2.0
- Our efforts to improve the performance of the existing Draft 2.0 conservative scheme fall into two broad areas
  - Improved estimate of LocalFairRate
    - In existing specification, goal for initial value of LocalFairRate is equal share: "unreserved rate/active stations"
  - Means to estimate Round Trip Time within a Congestion Domain
    - Existing specification uses "RTTWorthofIntervalsHavePassed", but does not specify how this is obtained, and what RTT refers to.
- We have focused on the Dual Transit buffer scheme
  - Simplifications involve looking only at STQ buffer size
    - Do not have targets for utilizations, as reflected by Rate Thresholds
- No new or additional variables are introduced
- Also improved the overall weighted fair allocation mechanism
  - Applicable to both aggressive and conservative scheme

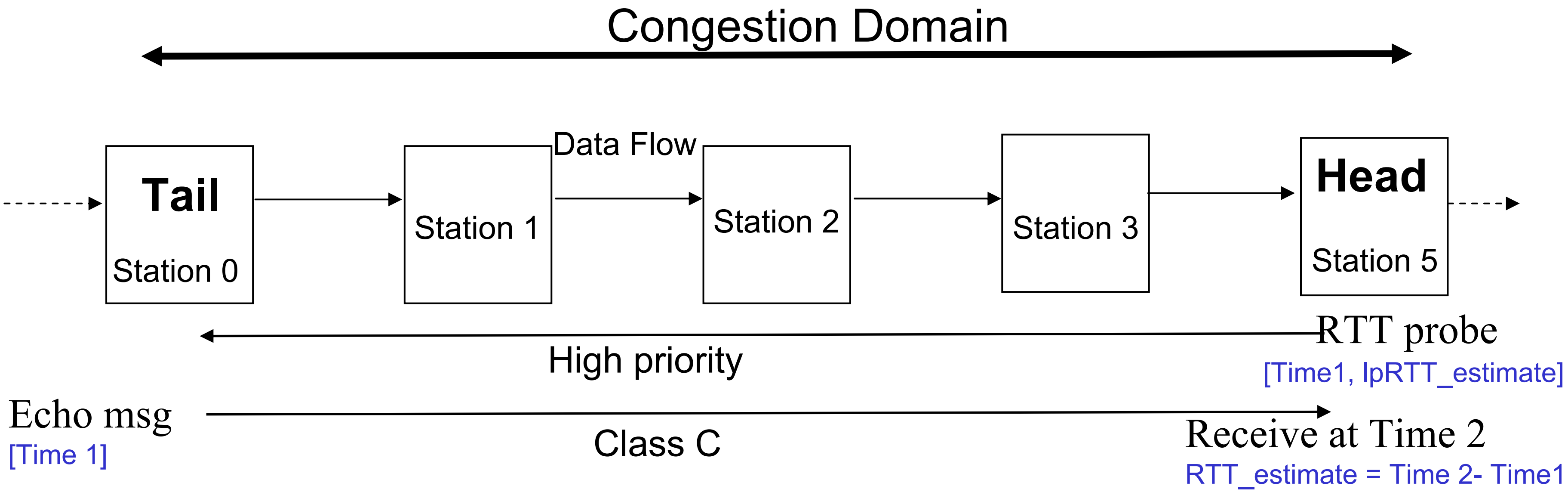
Enhanced LocalFairRate Calculation

- Scheme was designed to work with a reasonable initial value
  - We have improved the initial value of LocalFairRate slightly to factor in the case when the demand of local station is less than "fair share"
    - Helps in reducing oscillations and convergence time
- LocalFairRate is calculated every round trip time, as spec'd before
- During the period a node is congested, we introduce two simple modifications to LocalFairRate computation to reduce oscillations
  - If STQDepth > STQHighThreshold, LocalFairRate reduced by a factor
    - We add a lower bound to LocalFairRate based on what fair share should be
  - If STQDepth < STQLowThreshold, LocalFairRate is increased by a factor
    - We improve the value LocalFairRate is increased by, knowing the amount of traffic added by the local station
- New sources starting up at high rate can starve downstream stations whose STQ buffer is full due to rapid onset of congestion
  - If this happens, station recomputes LocalFairRate when local congested station has a lower add rate than the fair share, and advertises this updated rate

Estimation of Round Trip Time

- ☐ Conservative scheme specified in Draft 2.0 updates LocalFairRate (increase/decrease) every RTTWorthofIntervalsHavePassed
- ☐ Having all the nodes update their localFairRate at the same fundamental frequency of the control loop (RTT) gives a stable and properly damped control
  - When local state is congested or uncongested
  - Recompute rate only after previous change in the rate has taken effect. Control loop RTT includes:
    - the instant at which LocalFairRate is advertised by a congested node in an FCM
    - the FCM is received by all the nodes in the congestion domain
    - these nodes adjust their allowedRateCongested
    - congested node observes results of this change in its received FE traffic
- ☐ RTT is estimated for the congestion domain:
  - "Head" (next to congested span) to "Tail" (last upstream node of domain)
  - Estimated by Fairness Control Unit.
  - Measured RTT is smoothed by a low pass filter

Estimation of RTT (contd.)



Congestion Domain

Data Flow

Tail

Station 0

Station 1

Station 2

Station 3

Head

Station 5

RTT probe

[Time1, lpRTT_estimate]

High priority

Echo msg

[Time 1]

Class C

Receive at Time 2

RTT_estimate = Time 2- Time1

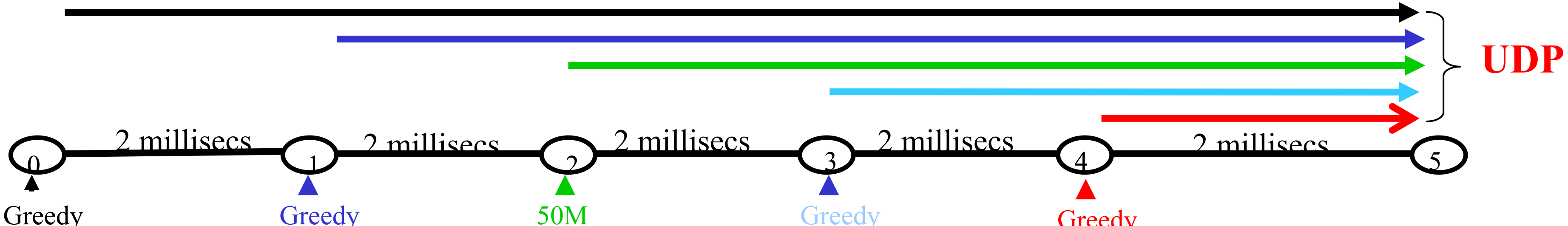
- ❑ Head station sends a message upstream with Time1 = current time, and low pass filtered RTT_estimate (lpRTT_estimate)
  - ➢ A high priority message to emulate transport of FCM
- ❑ Upstream stations in congestion domain copy lpRTT_estimate
- ❑ Tail station echoes back timestamp, Time1 in a message to Head
  - ➢ A low priority message (Class C) to emulate transport of FE traffic
- ❑ Head station computes RTT_estimate and performs low pass filter

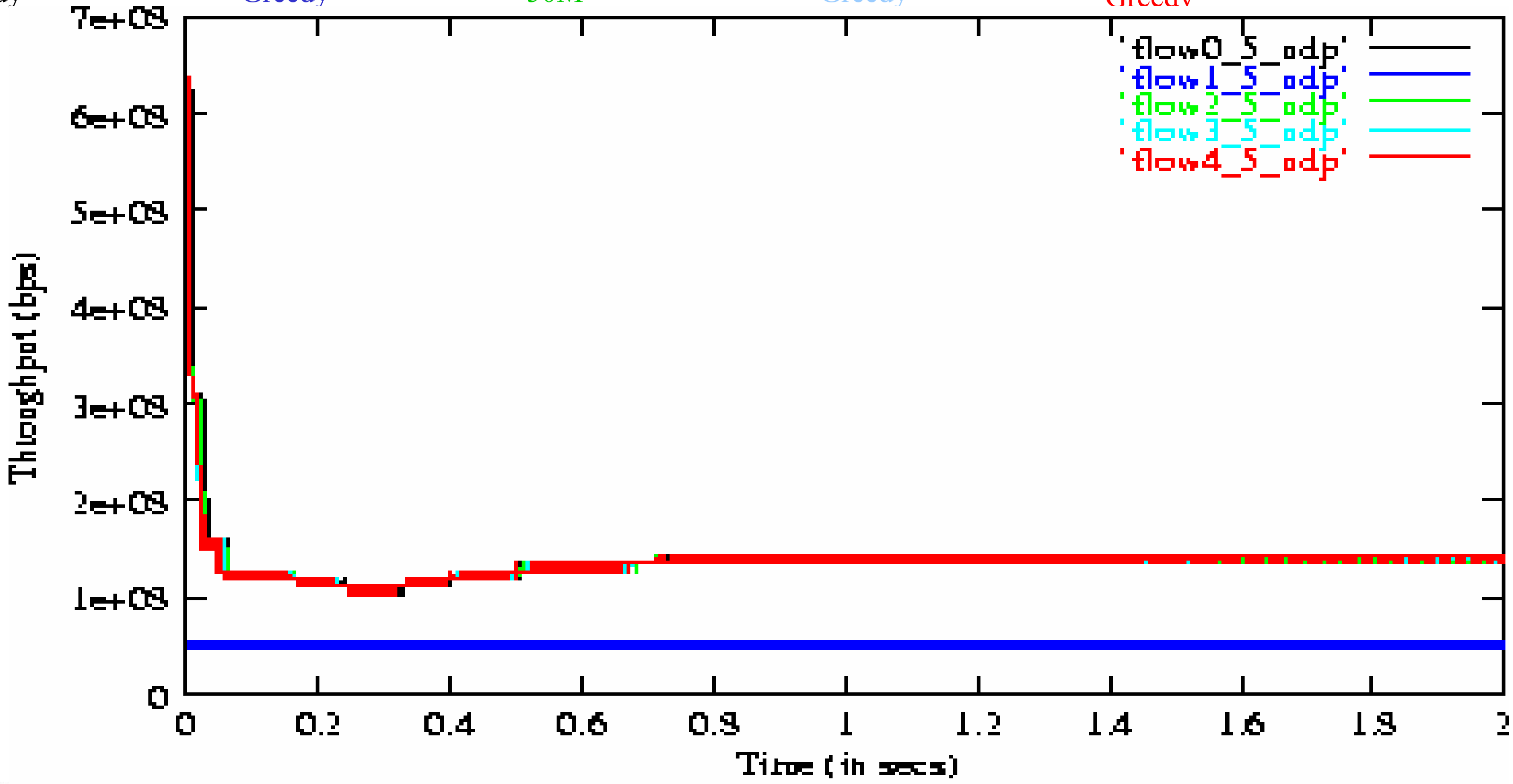
Improvements for Weighted Fair Allocation

- When calculating the LocalFairRate, the estimate of fair share is:
  - unreserved_rate/activeWeights
- Allows for weighted fair allocation, rather than equal share
  - may have been oversight in specification
- Since lpAddRate of local station is known, further improve estimate
  - (unreserved_rate - lpAddRate)/(activeWeights - WEIGHT)
- In Table 9.5, we needed to make sure that the Fairness Control Message was propagated appropriately
  - Simple change (in Row 1) to cover case: normLocalFairRate=rcvdFairRate
  - Ensure that FCM was propagated all the way to the “Tail” node of the congestion domain
- These changes removed some of the problems we had been encountering with weighted fair allocation for both aggressive and conservative modes

Enhanced Conservative Scheme Performance

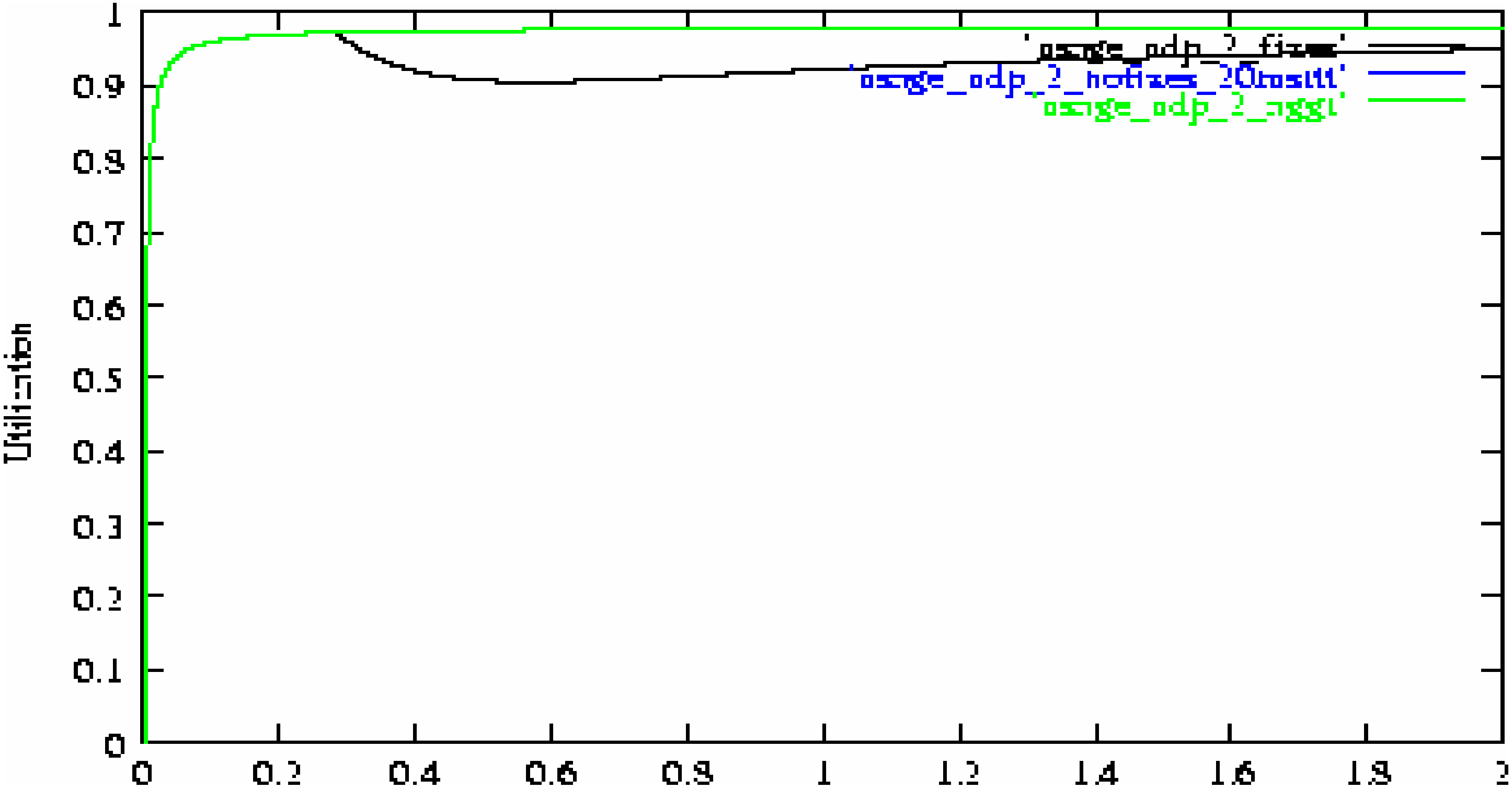
- Simulation: constant rate (UDP) sources; Default thresholds; agingInterval = 200µsecs.)
- Convergence time to “fair share” is approximately 4 round trips



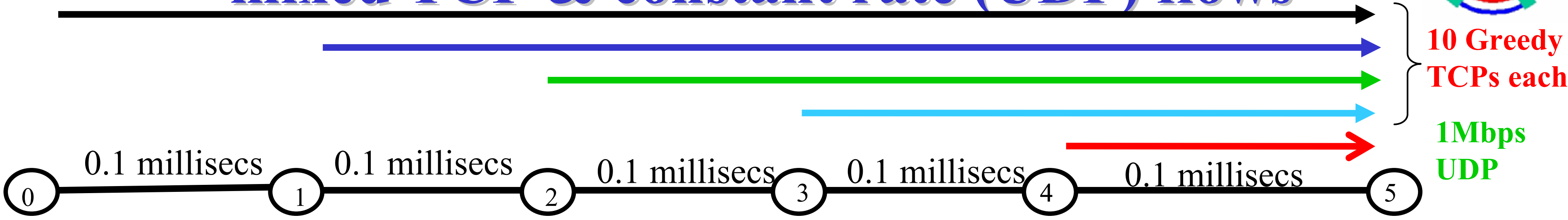


Link utilization of 3 schemes with Constant Rate traffic

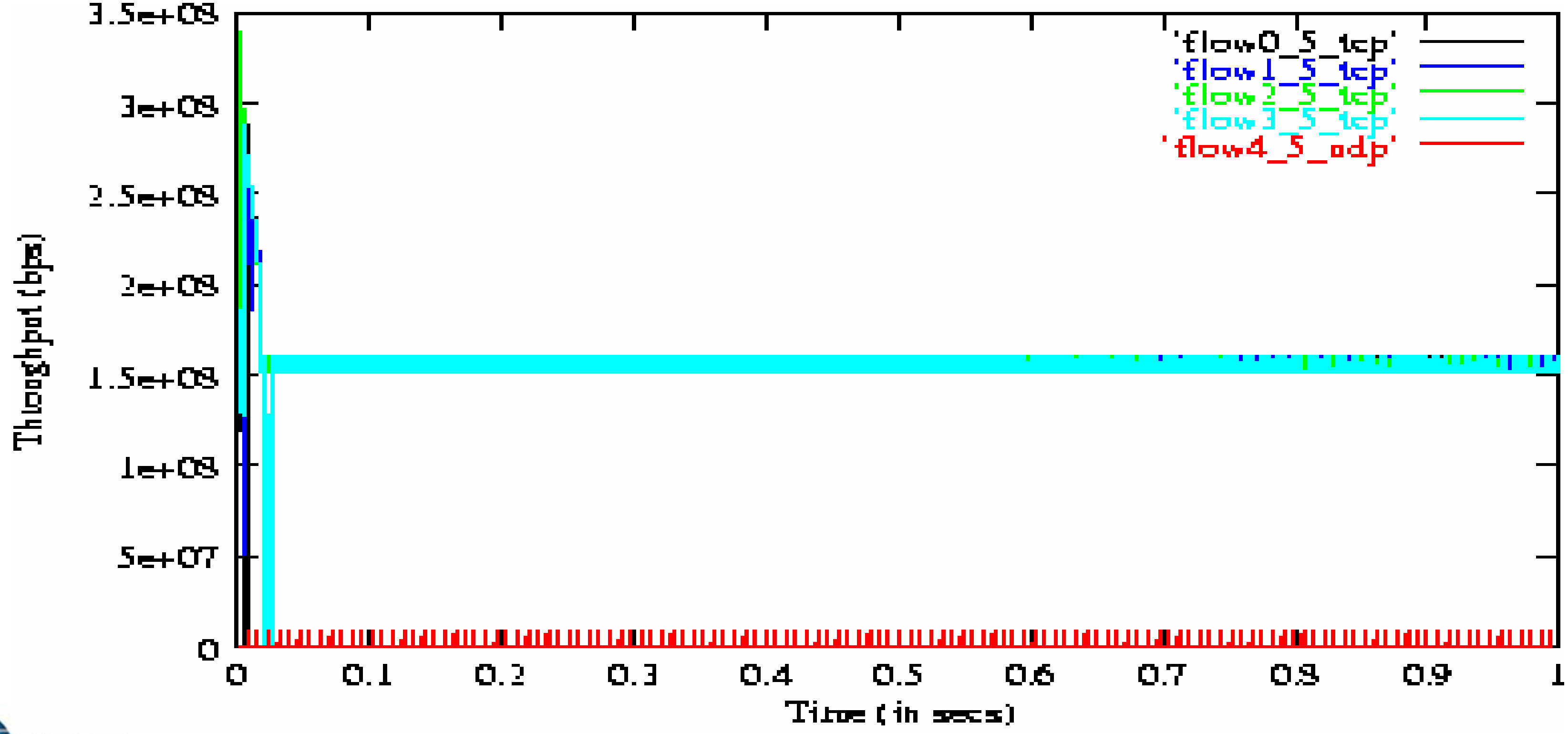
- Oscillations in individual station’s throughput considerably reduced with new scheme
  - Initially, converges to “equal share” (120 Mbps). Algorithm actually accounts for lower demand from station 2, and gradually approaches max-min fair share = (622-50)/4 = 144 Mbps
    - Max-min fairness (i.e., per-flow fairness) will not be achieved in all cases though
- Bottleneck link utilization for enhanced conservative scheme is below the aggressive and original conservative schemes, when averaged over time



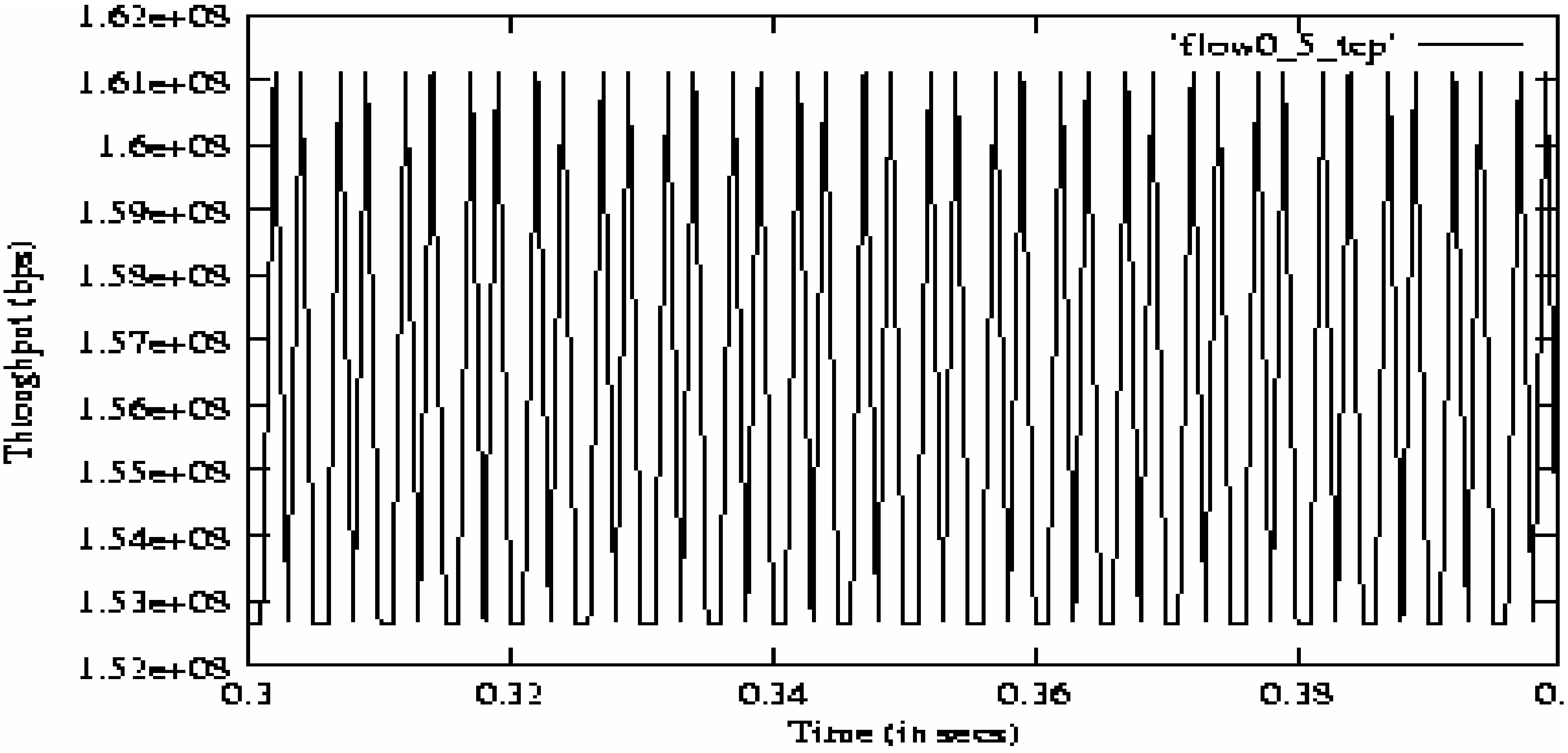
Performance of Enhanced Conservative with mixed TCP & constant rate (UDP) flows



- UDP flow is constant rate (=1Mbps) on last hop
- 10 greedy TCP flows from each RPR node; TCP co-resident w/MAC client
- Span 4 → 5 is bottleneck; Fair rate per source = (622-1)/4 ≈155 Mbps



Short-term Throughput for one node

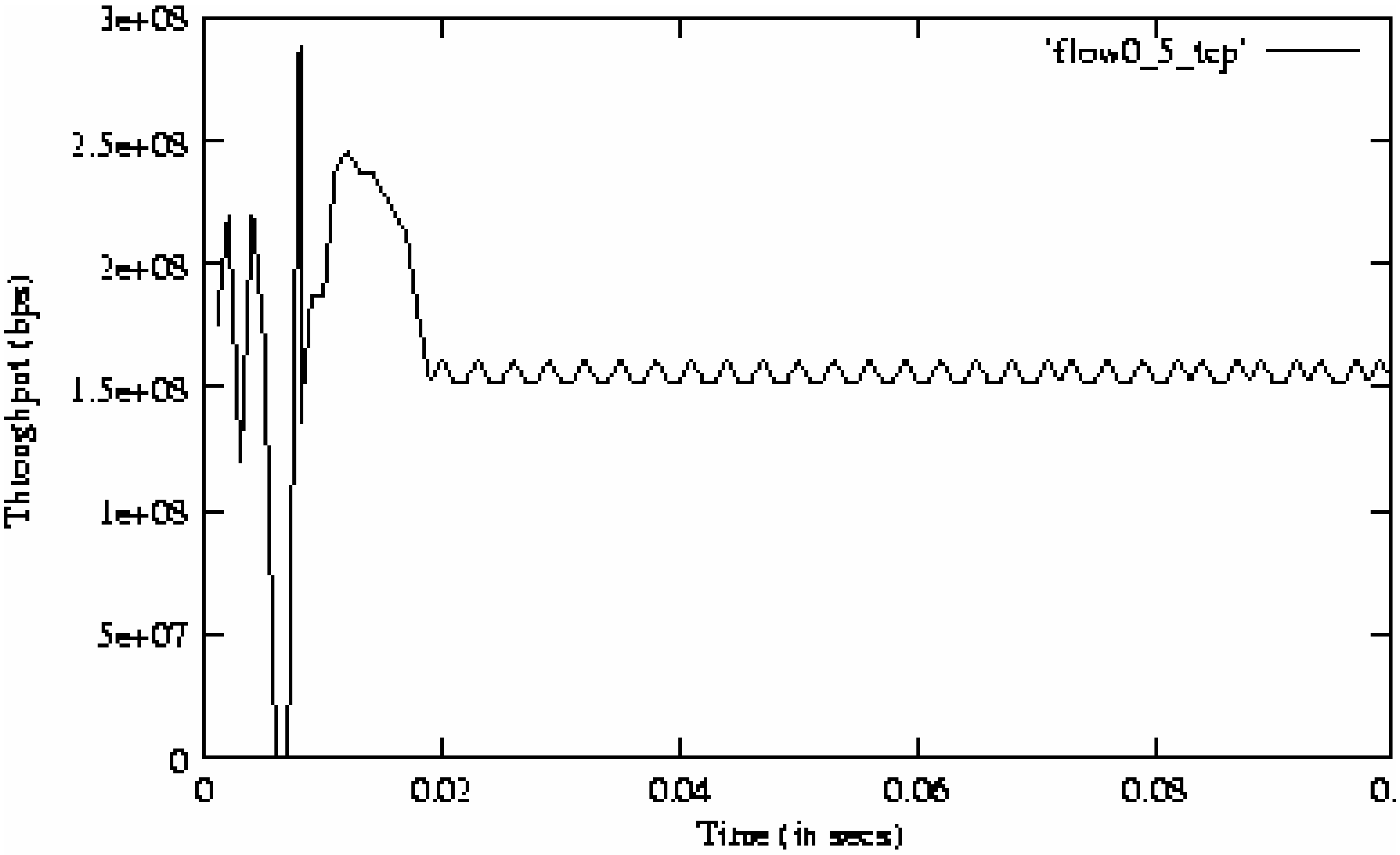


Short term behavior of aggregate of 10 TCPs from node 0

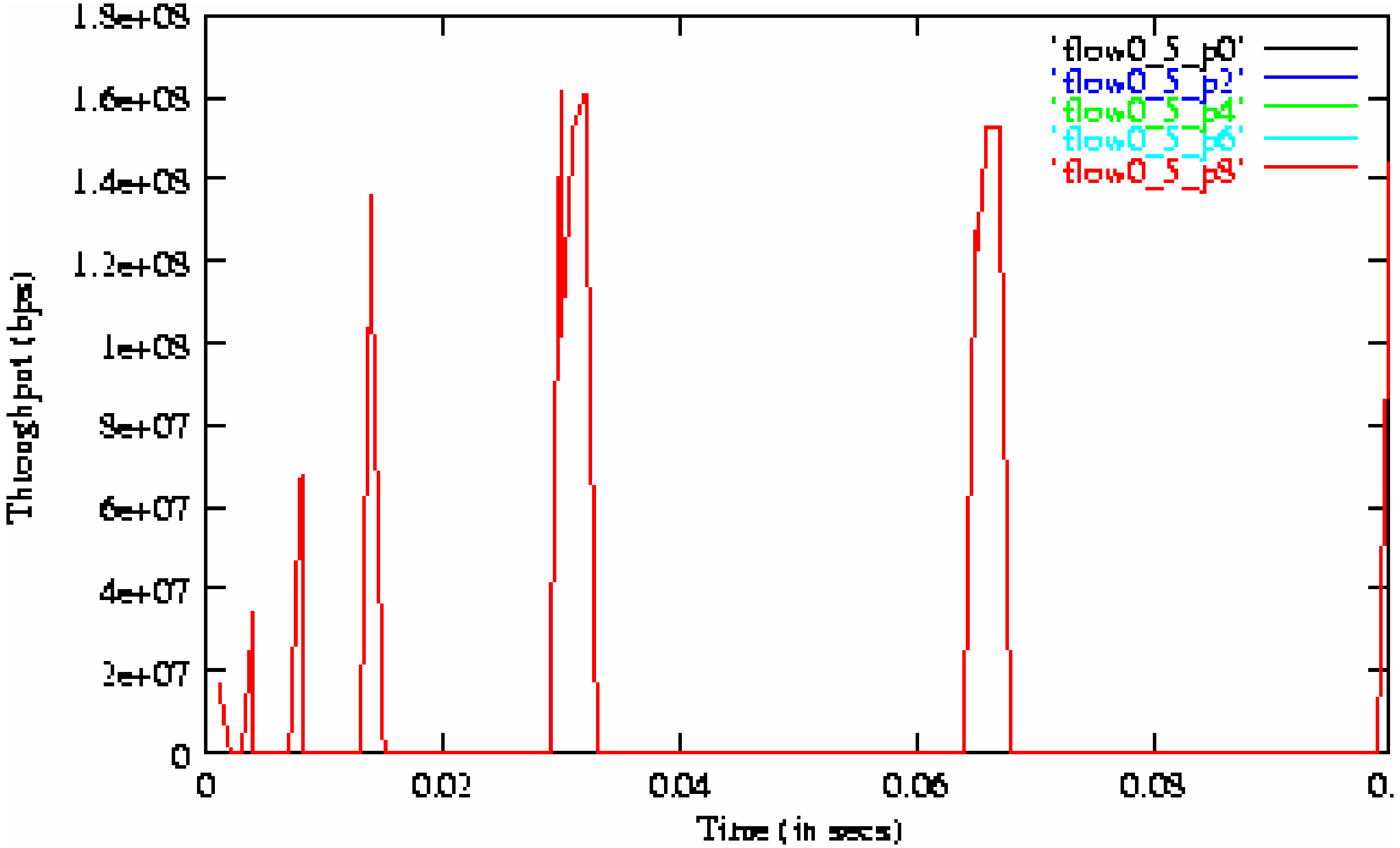
- Individual node throughput oscillates over a tight range

Initial short-term Throughput for one node

- ❑ Initial startup behavior to show convergence times
- ❑ Initial convergence to the fair, stable rate occurs within 20 milliseconds
  - ➢ Primarily driven by TCPs being in slow start and waiting for acks. to grow the window

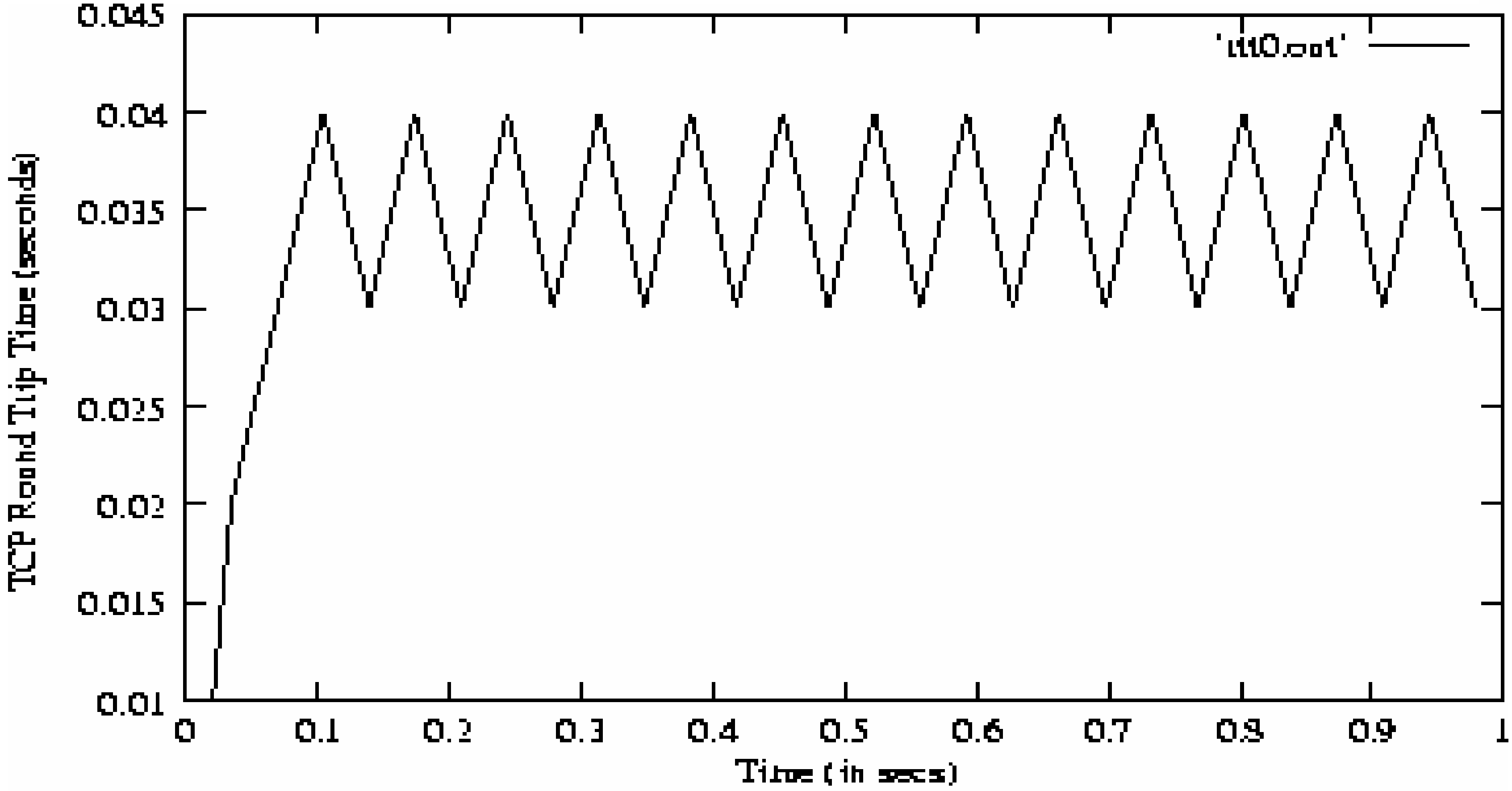


Behavior of individual TCPs at one node



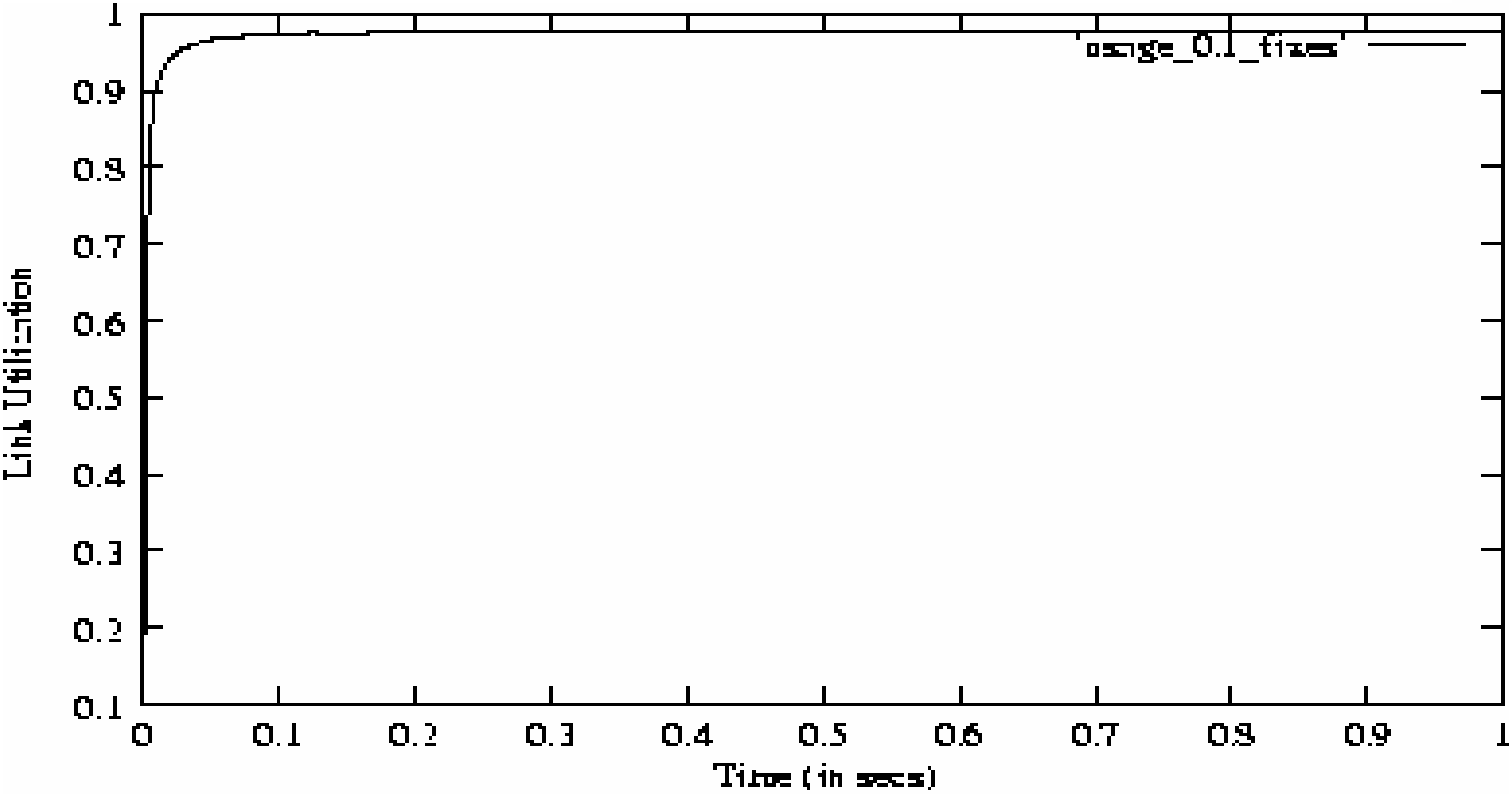
- ❑ Each individual TCP generates packets when acks are received
  - ➢ Overlaps with the other TCPs at source in generating packets (buffered in client)
  - ➢ Startup behavior shows TCP slow start action, exponentially increasing window

RTT Behavior of TCP flows



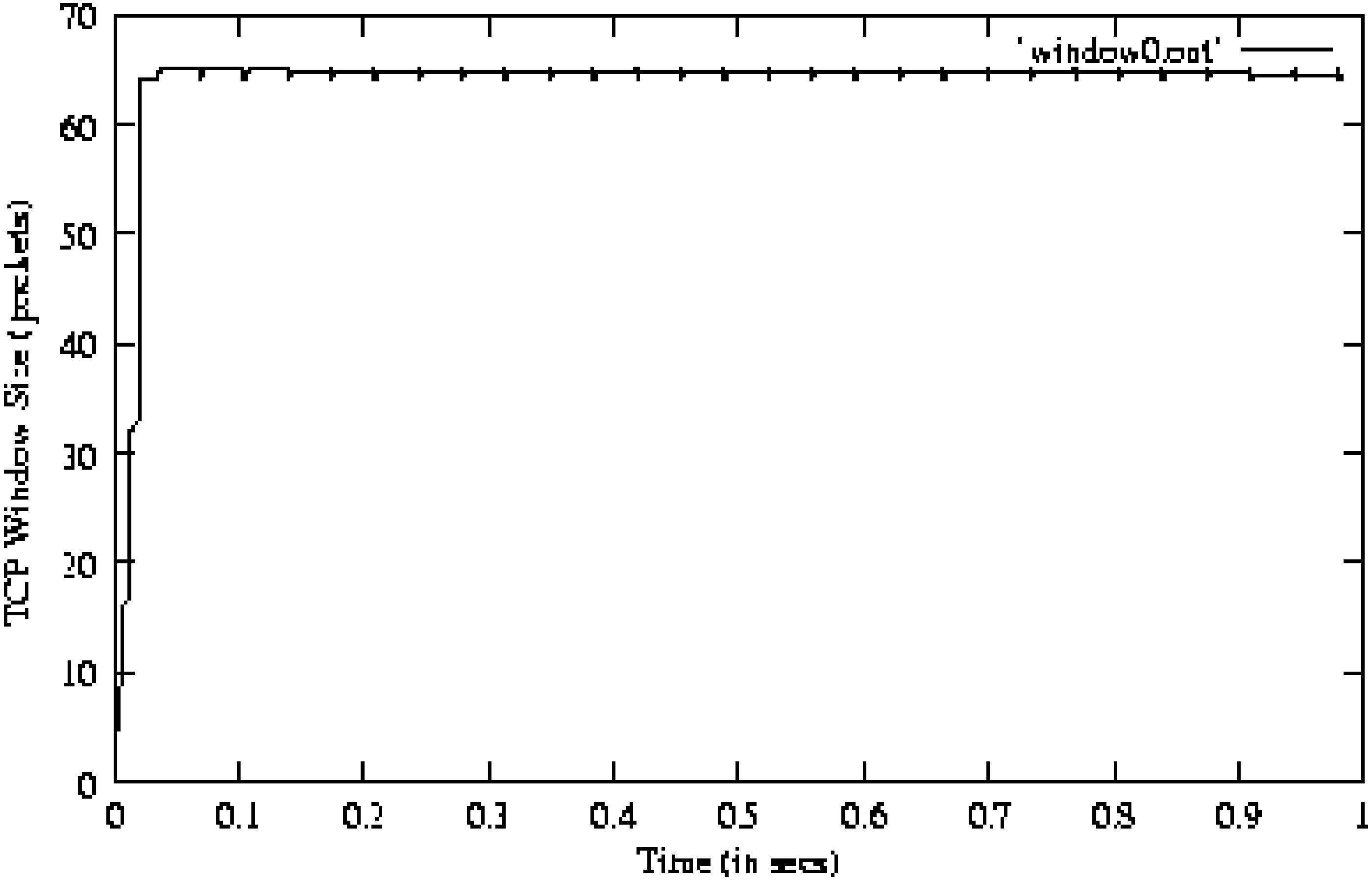
- Large delay even though total RPR link propagation delay = 1 milliseconds
  - Reflects considerable queueing in the client buffers (∴↑convergence time)
- Oscillations in RTT follow the pattern of aggregate throughput
  - TCP round trip time (Y-axis is Time in seconds) ≈ 35 milliseconds

Bottleneck Utilization with enhanced conservative mode



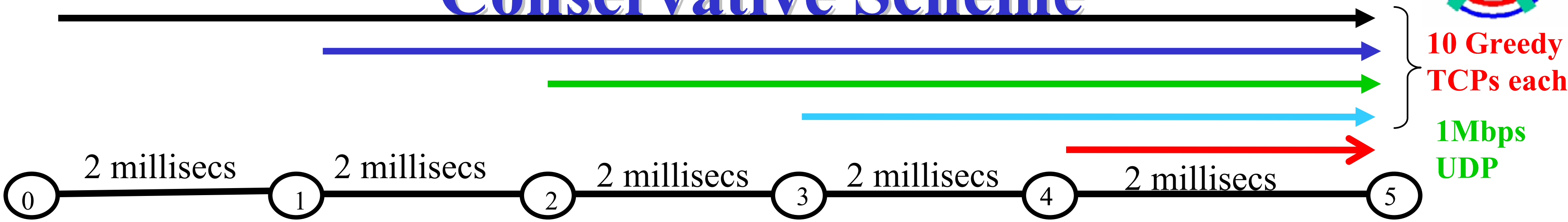
- ❑ Link utilization measured and averaged from time T=0
  - ➢ Instantaneous utilization may (will likely) be higher than utilization shown
- ❑ Utilization reached after 1 second = 97.87%

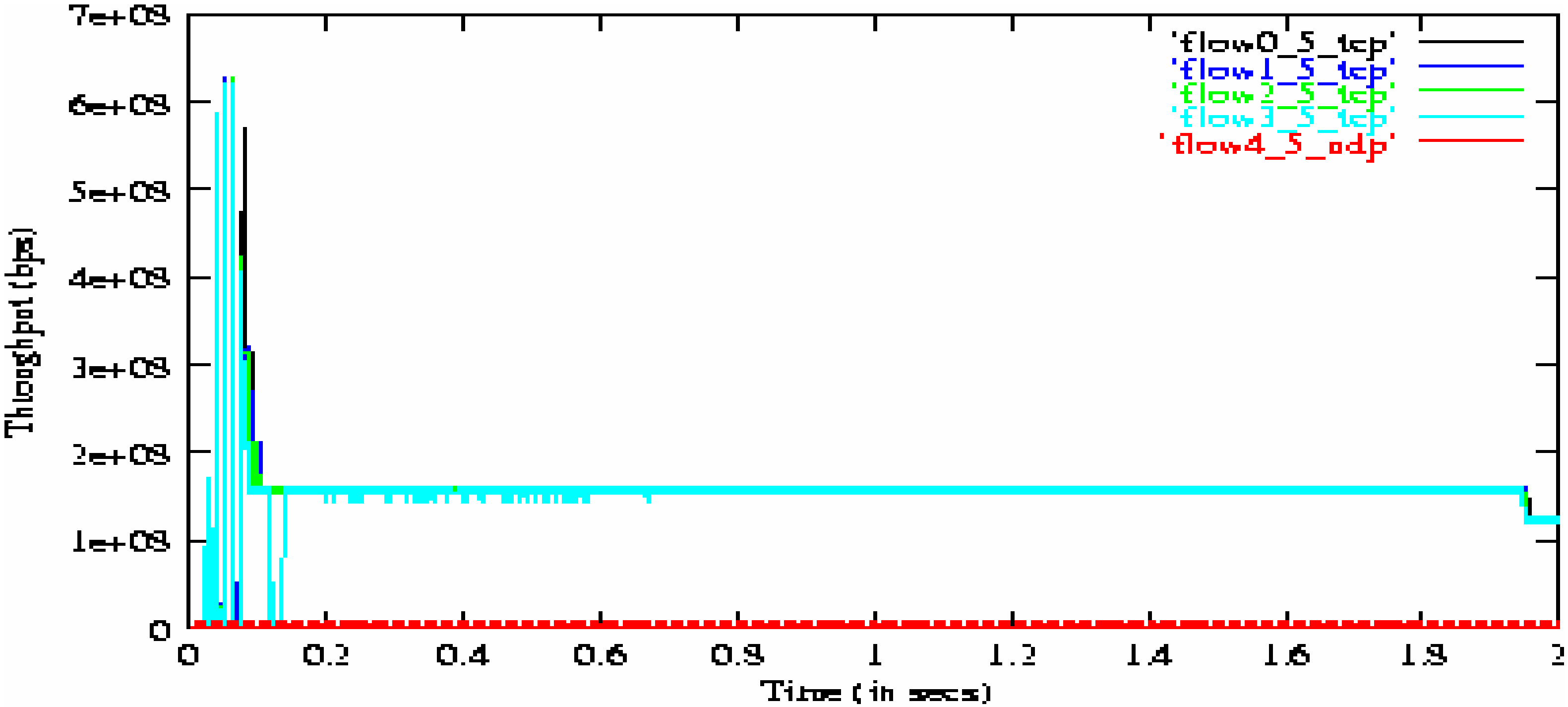
One flow's TCP Window



- ❑ TCP window grows reasonably quickly to max. window size of 64

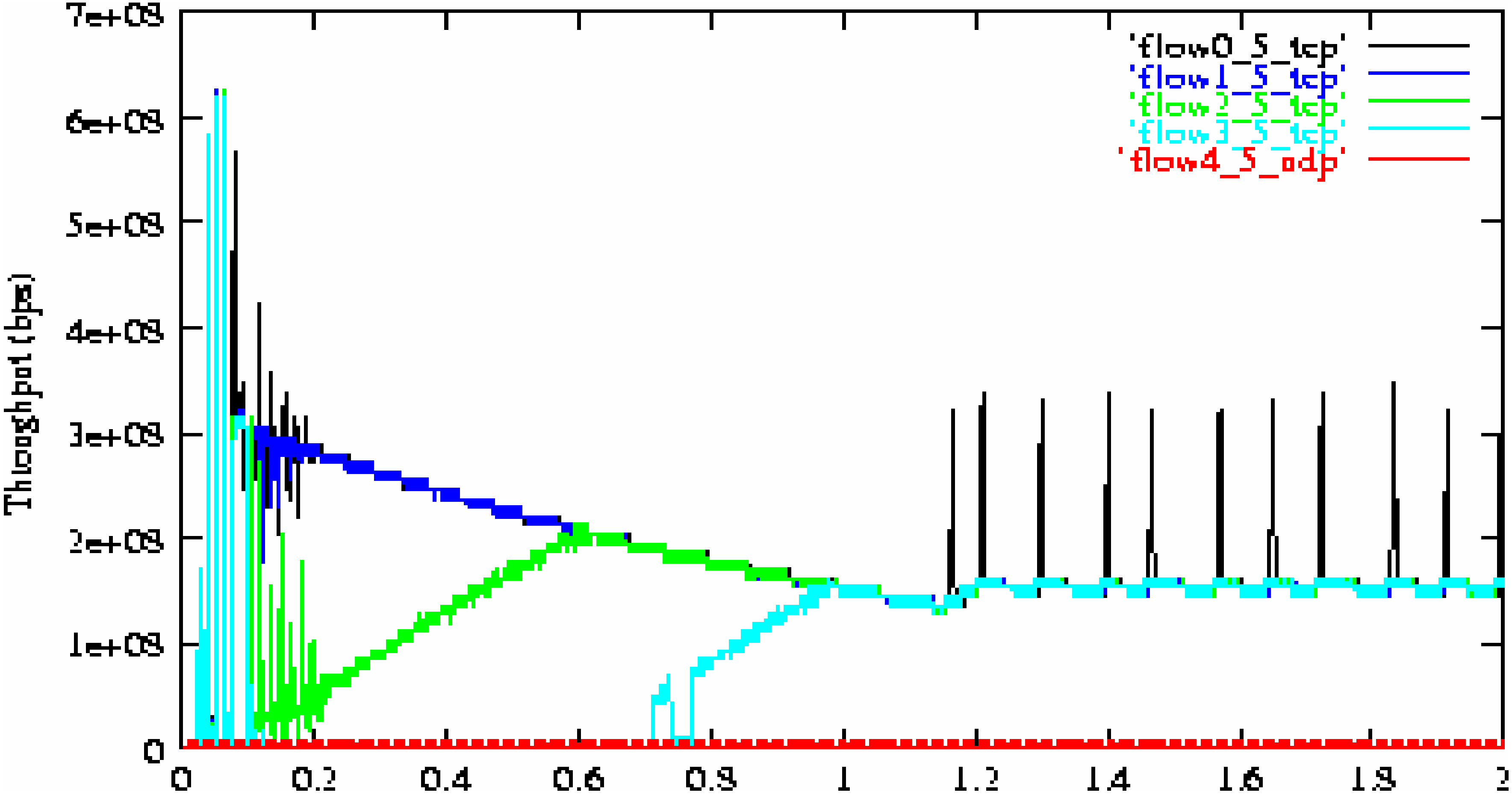
TCP with Large RTT: Enhanced Conservative Scheme





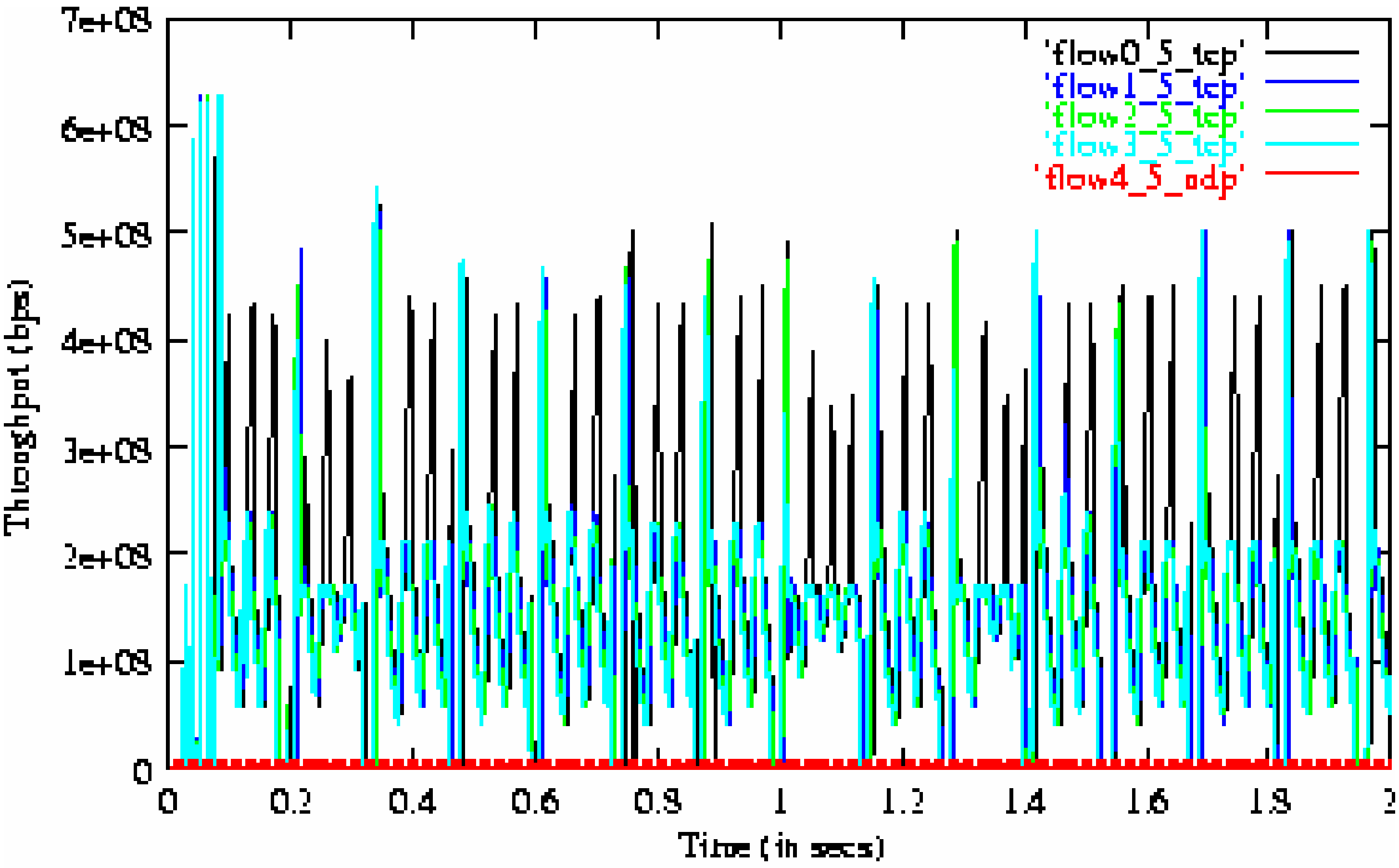
- ❑ What if we increase the ring size (RTT) considerably?
  - ➢ Per hop link propagation delay set to 2 milliseconds (400 Km)
- ❑ STQ buffer still maintained at 256 Kbytes
  - ➢ Meet difficult challenge to operate with same buffer size over wide dynamic range

TCP with Large RTT: Existing (Draft 2.0) Conservative Scheme



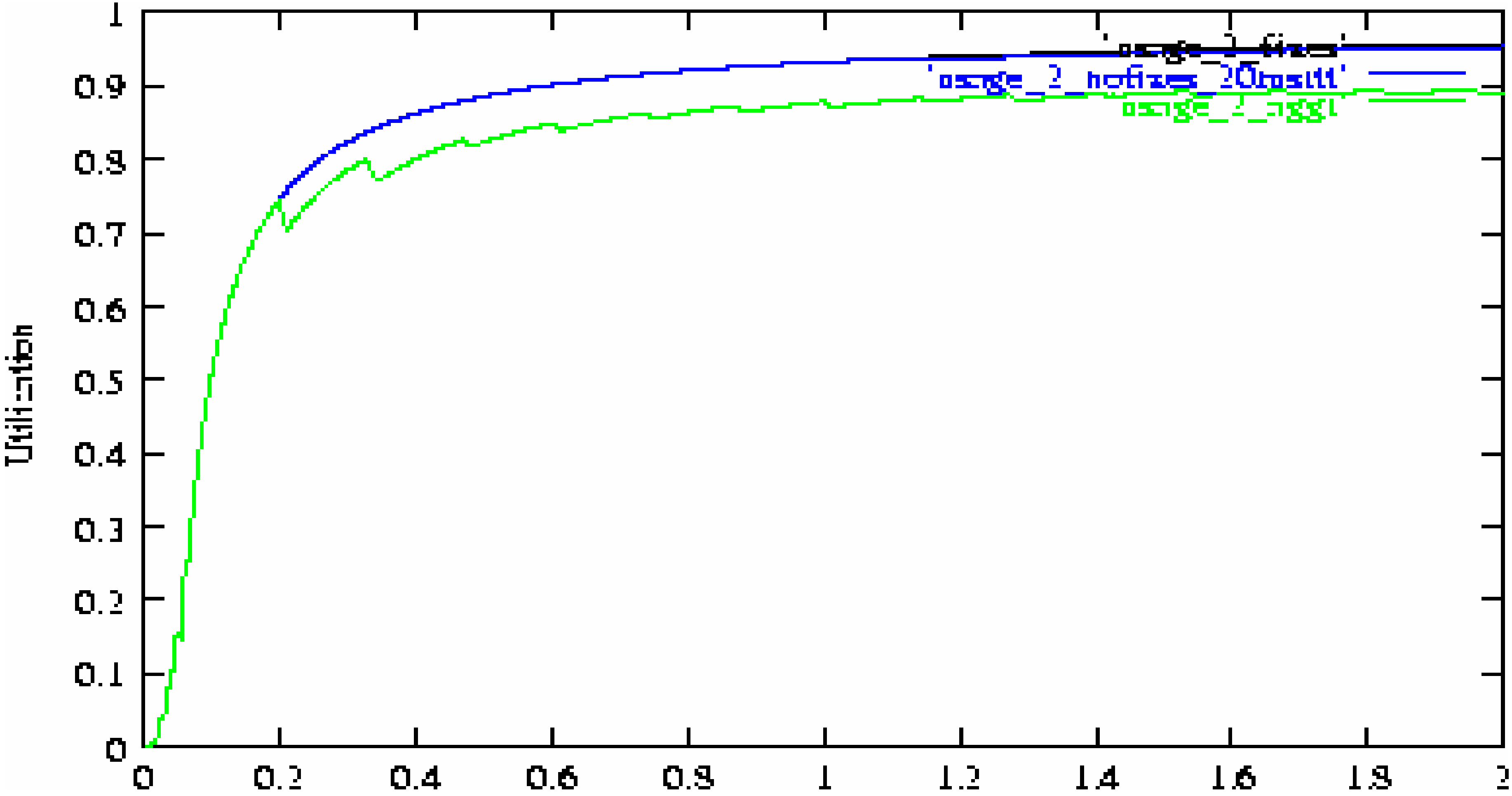
- ❑ Original conservative scheme with fixed RTT estimate = 20 milliseconds
- ❑ Node 0 (farthest upstream) appears to get an unfair advantage
  - ➢ Starves downstream nodes (e.g., node 3) when they experience congestion

TCP with Large RTT: Aggressive scheme



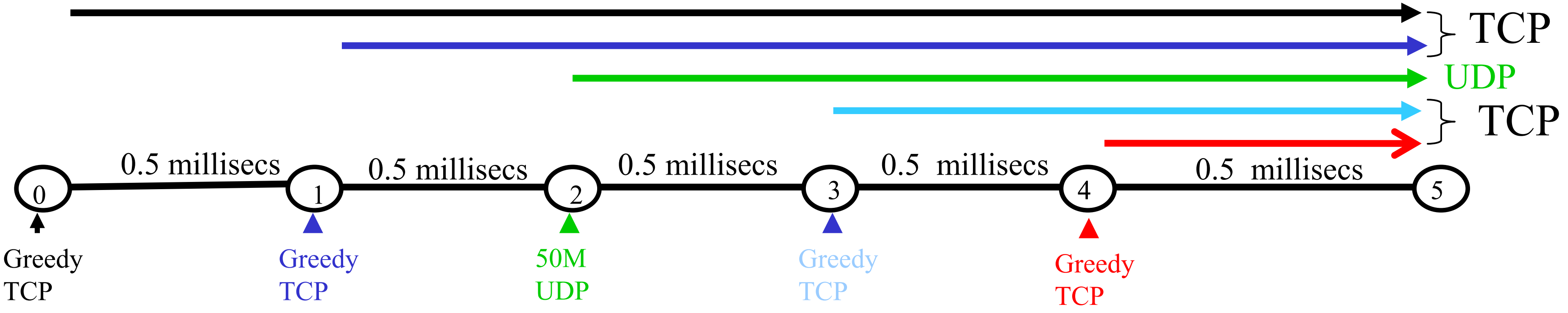
- ☐ Aggressive scheme behaves similar to previously explained (Nov. 2002) behavior for large RTTs

Comparison of Bottleneck Link Utilizations with TCP, Large RTT



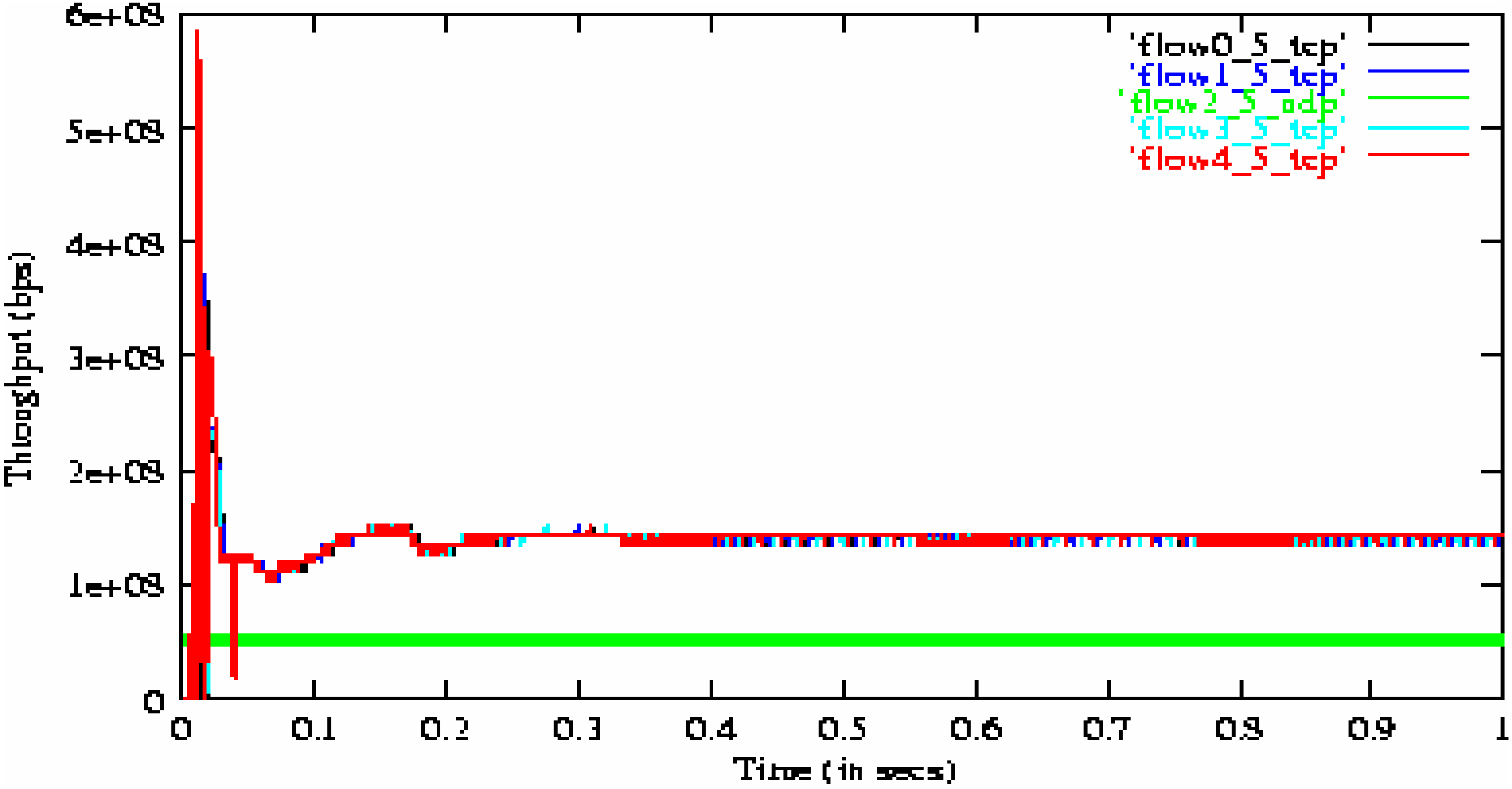
- ☐ Link utilization with enhanced conservative scheme is slightly higher than the existing conservative or aggressive schemes

Benefits of Estimating RTT accurately



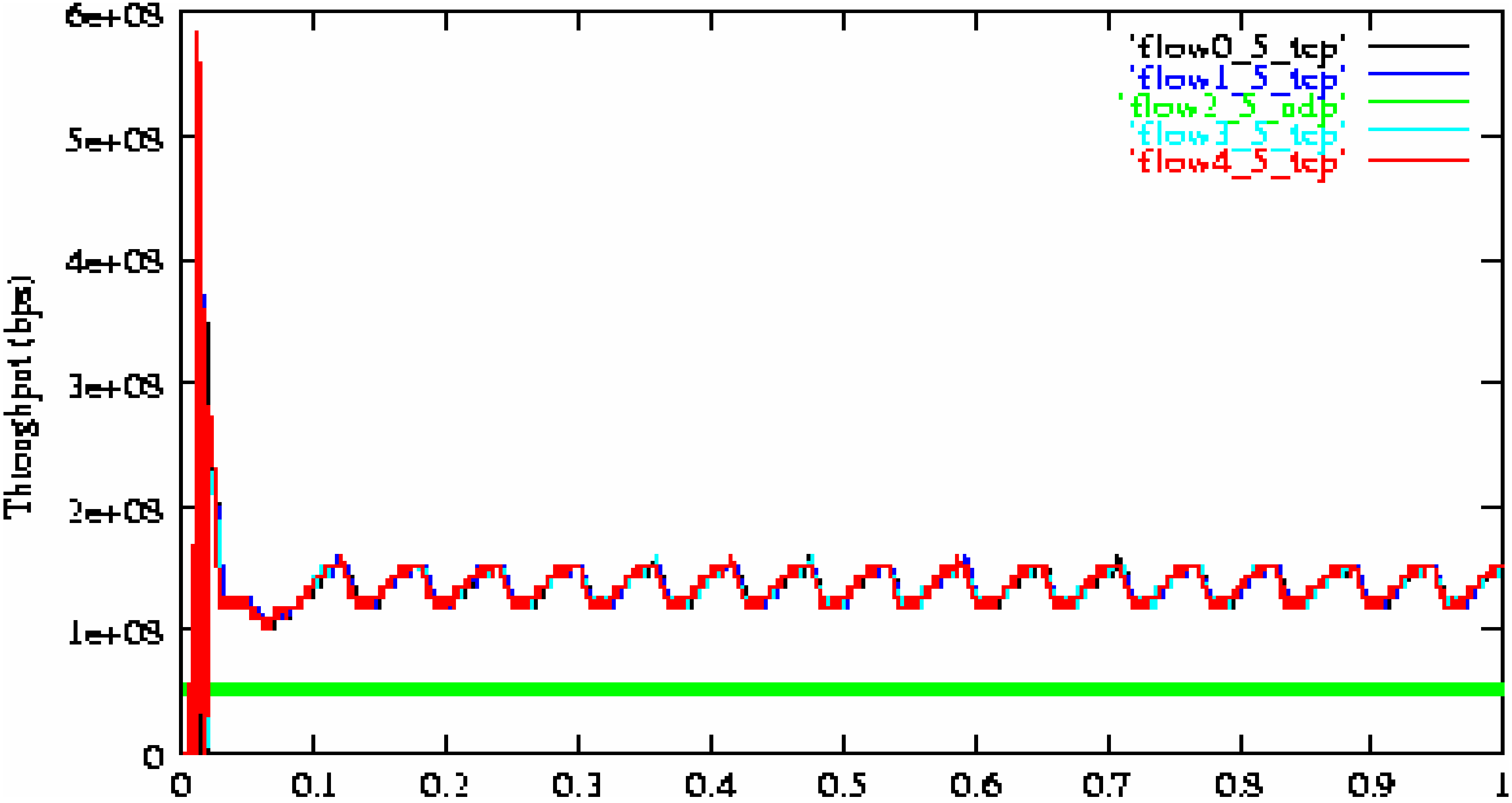
- ☐ What is the benefit of estimating the RTT accurately?
- ☐ We compared the performance of the Enhanced Conservative scheme with 3 experiments.
  - ➢ Estimating RTT accurately as suggested here
    - ❖ (5 msec. + STQ delays)
  - ➢ Having a fixed RTT = 10% of the round trip propagation delay of domain
    - ❖ (0.5*10) * 0.1 = 0.5 milliseconds
  - ➢ Having a fixed RTT = 20* round trip propagation delay of domain
    - ❖ (0.5*10) * 20 = 100 milliseconds
- ☐ Simulation configuration uses 4 TCP sources with 10 Greedy TCPs each and 1 fixed rate (50 M) UDP source

Enhanced Conservative Mode with dynamic RTT estimation with TCP&UDP flows



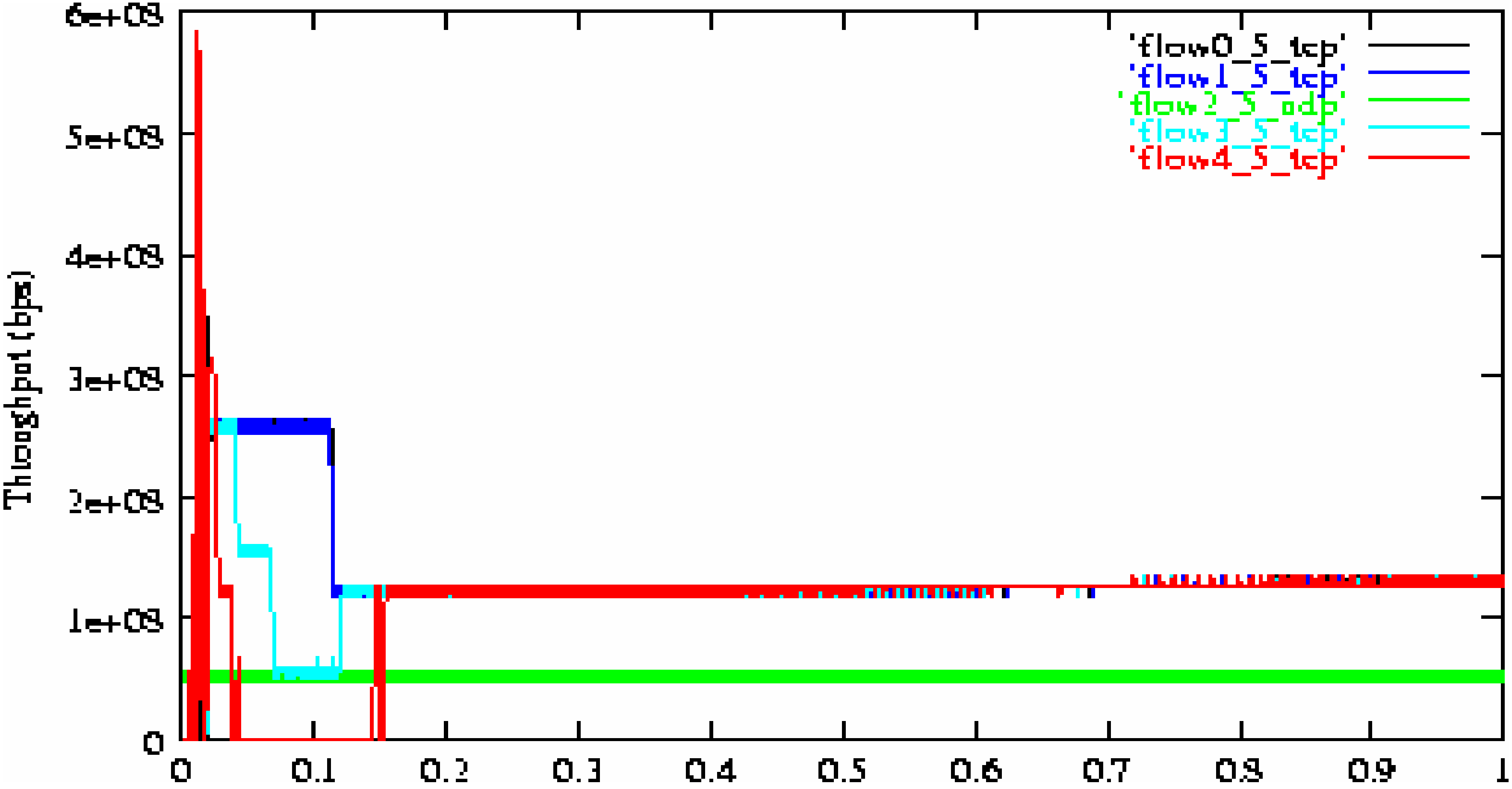
- Estimation of RTT using the protocol specified enables the flows to operate with small oscillations
  - congested node (node 4) does not starve, even during the initial transient
  - approaches fair share (**also using node 2's left over bandwidth**) over long term

Enhanced Conservative Mode, Fixed RTT=0.5 msec



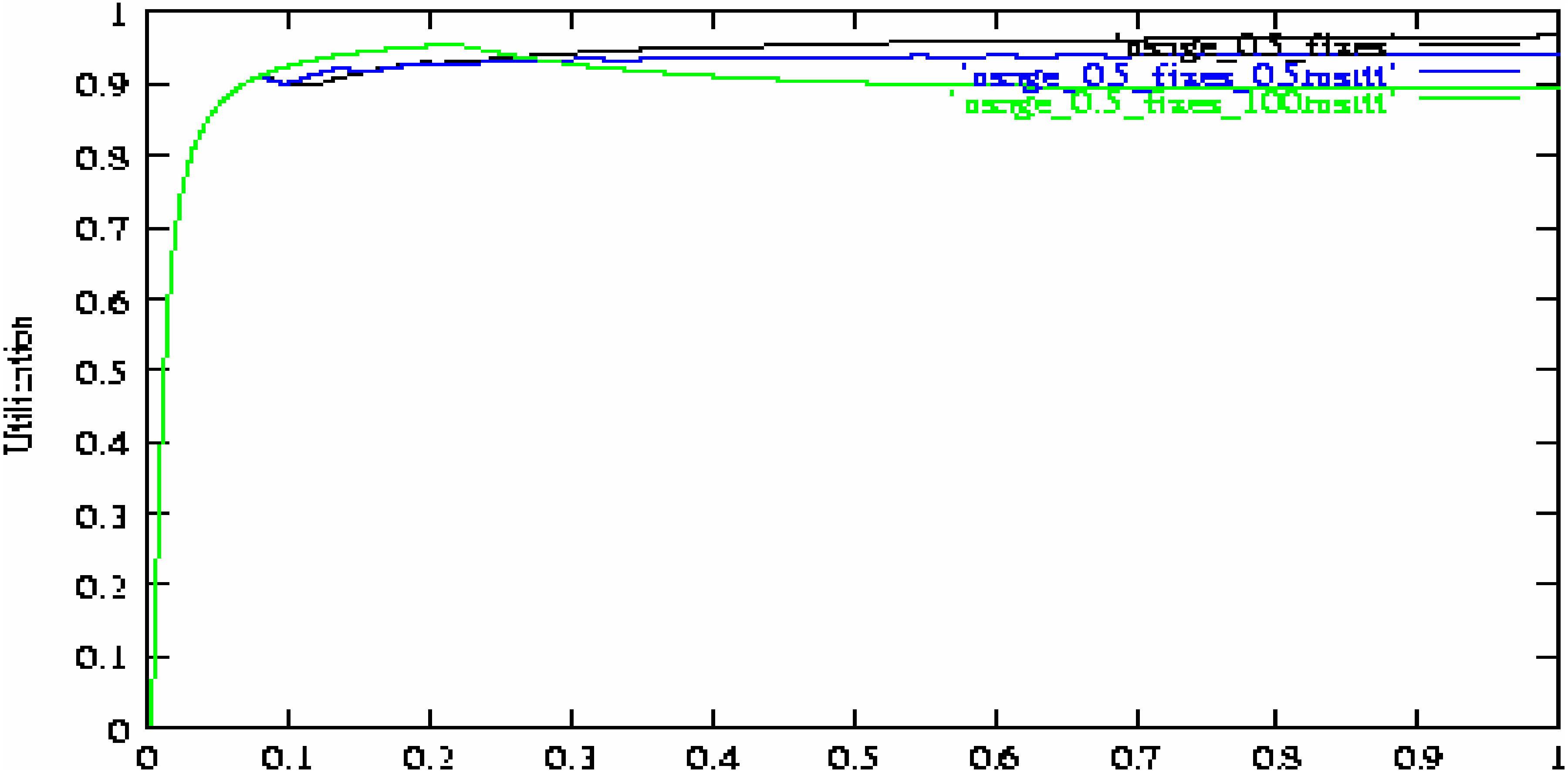
- ☐ Using fixed RTT =10% round-trip link propagation delays ⇒ some oscillation.
- ☐ Average behavior achieves “fair share” (but doesn’t recover unused bandwidth)
  - ➢ Reasonable performance for this example

Enhanced Conservative Mode, Fixed RTT=100msec



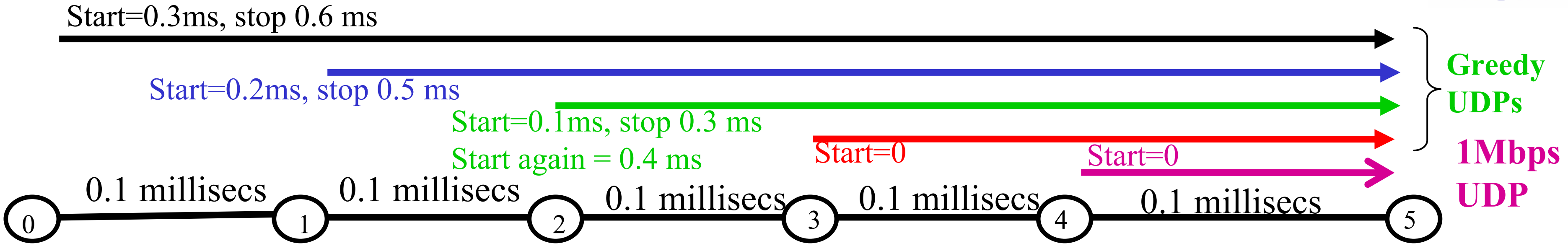
- ❑ Using fixed RTT =20* round-trip link propagation delays ⇒ some oscillation.
- ❑ Node 4 starved for a brief period – as a result of "overdamping"
- ❑ Average behavior achieves "fair share"

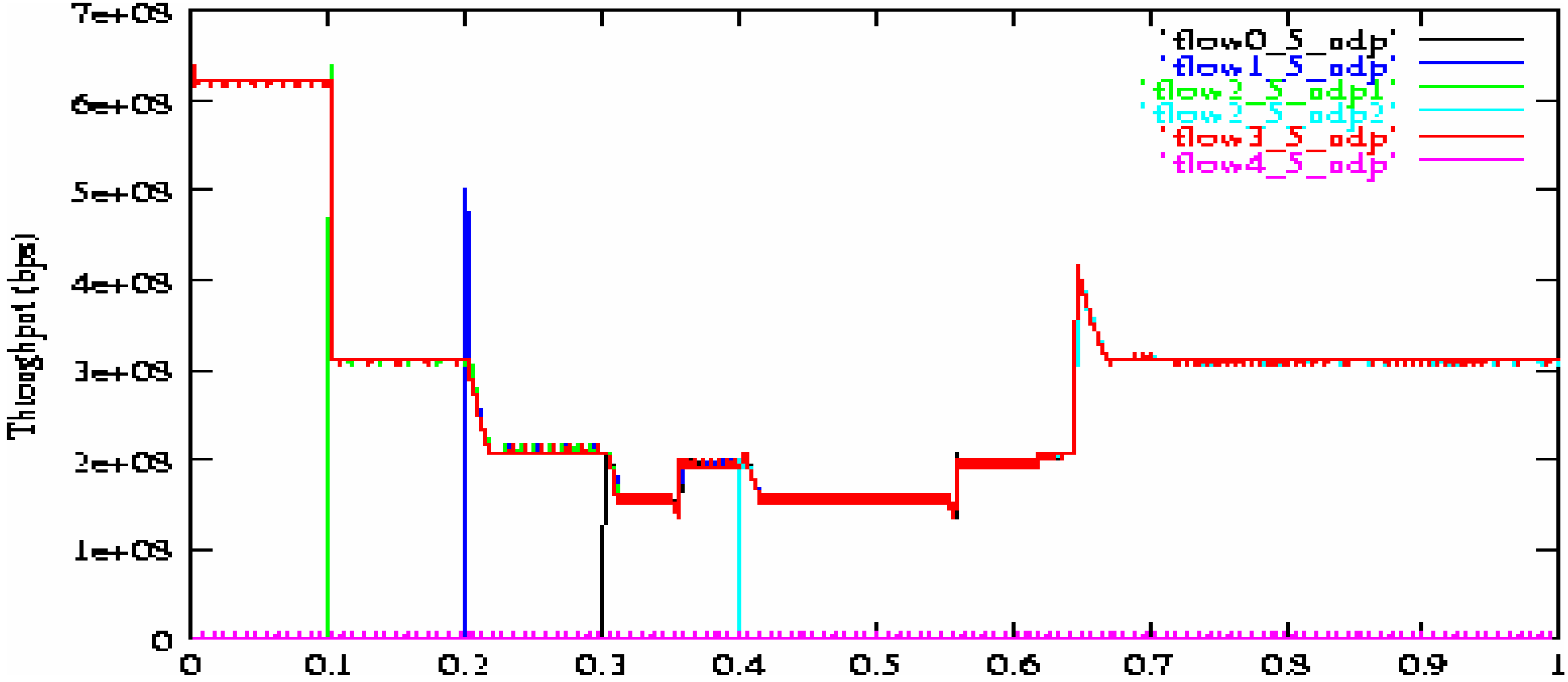
Bottleneck Link Utilization



- Enhanced conservative, dynamic RTT estimate, utilization = 96.73%
- Enhanced, fixed RTT = 10% of the round trip propagation delay, util= 94.11%
- Enhanced, fixed RTT = 20* round trip propagation delay, util = 89.45%

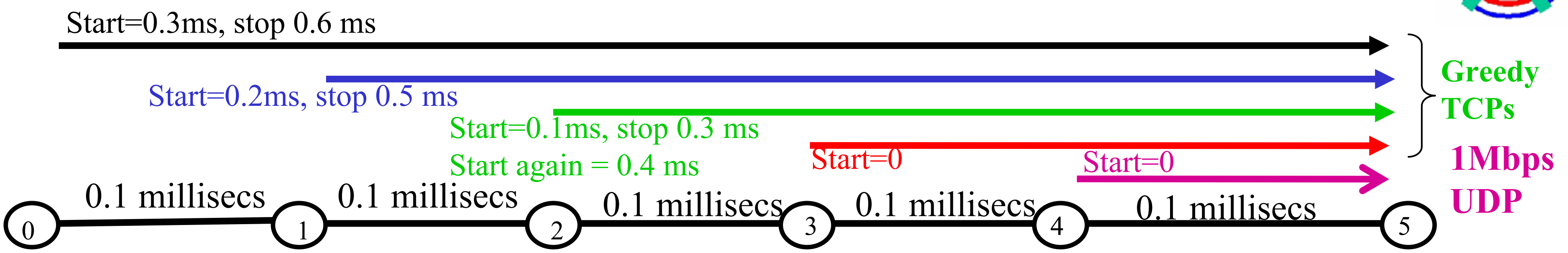
Enhanced Conservative Mode, Short Lived Flows: UDP





- ☐ What happens when we have short lived flows; interaction with long lived flows?
  - ➢ All flows are constant rate (UDP) flows. System is quite responsive to changes

Enhanced Conservative Mode: Short Lived Flows: TCP



Start=0.3ms, stop 0.6 ms

Start=0.2ms, stop 0.5 ms

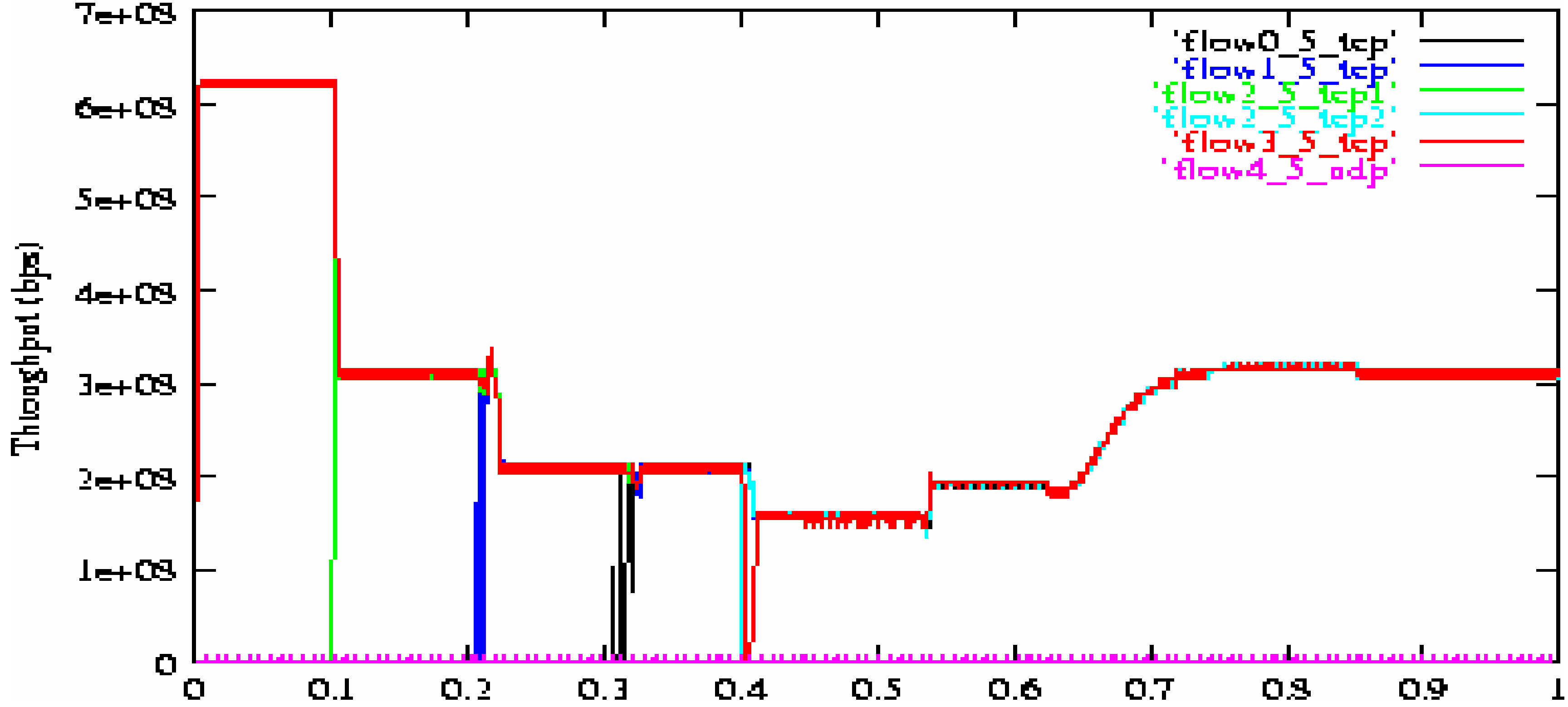
Start=0.1ms, stop 0.3 ms
Start again = 0.4 ms

Start=0

Start=0

Greedy TCPs

1Mbps UDP



- TCP flows also respond reasonably
  - convergence to fair rate when recovering after a flow stops is longer with TCP

Summary

- The conservative fairness allocation scheme for RPR can be made to work reasonably when simple improvements are made
- We have suggested two broad improvements
  - Improved estimate of LocalFairRate
    - improved estimate of fair share, account for unused bandwidth, set a lower bound for LocalFairRate when it is reduced during congestion.
  - Estimate Round Trip Time within a Congestion Domain
- The enhanced conservative mode works quite well
  - improved bottleneck link utilization: very high 90s (%) range
  - operates with a small STQ (256Kbyte) over wide range of ring RTT (2msec)
  - minimizes oscillation; reduces likelihood of starvation of a congested node
    - A station's transmit rate stabilizes quickly to fair share
  - reasonable convergence times: 3-4 round trips for UDP
- We encourage/request others to further examine scheme
  - Rate based schemes notorious for complexity and counter-intuitive behavior