Forward Explicit Congestion Notification (FECN) for Datacenter Ethernet Networks

Jinjing Jiang, Raj Jain, Chakchai So-In
Washington University In Saint Louis
Saint Louis, MO 63131
Jain@wustl.edu

IEEE 802.1au Congestion Notification Group Meeting,
Orlando, FL, March 12-15, 2007

These slides are also available on-line at
http://www.cse.wustl.edu/~jain/ieee/fecn703.htm

- Top 10 Requirements for a Good Scheme
- FECN Overview
- Switch Algorithm and Enhancements
- Simulation Results
  - FECN with TCP flows
  - Symmetric Topology
  - Large Topology
  - Bursty Traffic

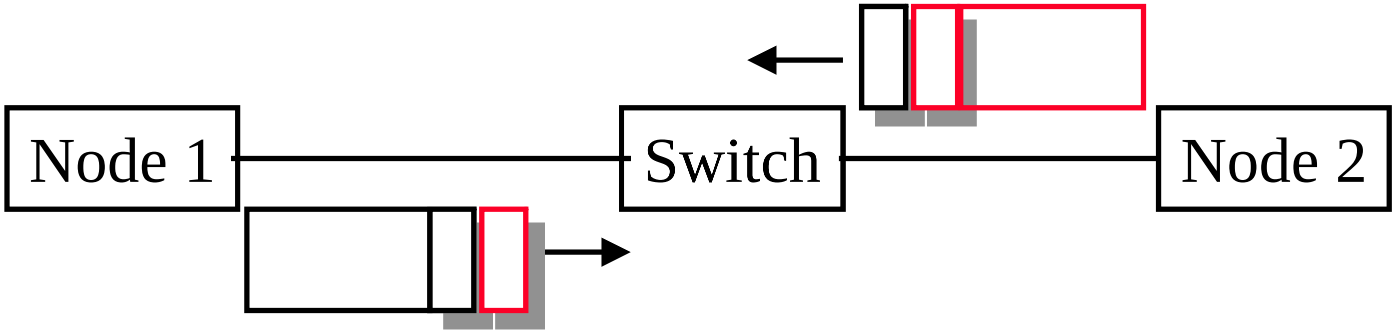
Datacenter Networks

- Bounded delay-bandwidth product
  - High-speed: 10 Gbps
  - Short round-trip delays
  - 1 Mb to 5 Mb delay-bandwidth product
- Storage Traffic ⇒ short access times ⇒ Low delay
- Packet loss ⇒ Long timeouts ⇒ Not desirable

Goals of FECN

1. Fast convergence to stability in rates
   Stable rates ⇒ TCP Friendly (IETF feedback)
2. Fast convergence to fairness
3. Good for bursty traffic ⇒ Fast convergence
4. Efficient operation: minimize unused capacity. Minimize chances of switch Q=0 when sources have traffic to send
5. Extremely low (or zero) loss
6. Predictable performance: No local minima
7. Easy to deploy ⇒ Small number of parameters
8. Easy to set parameters
9. Parameters applicable to a wide range of network configurations link speeds, traffic types, number of sources.
10. Applicable to a variety of switch architectures and queueing/scheduling disciplines

FECN Overview

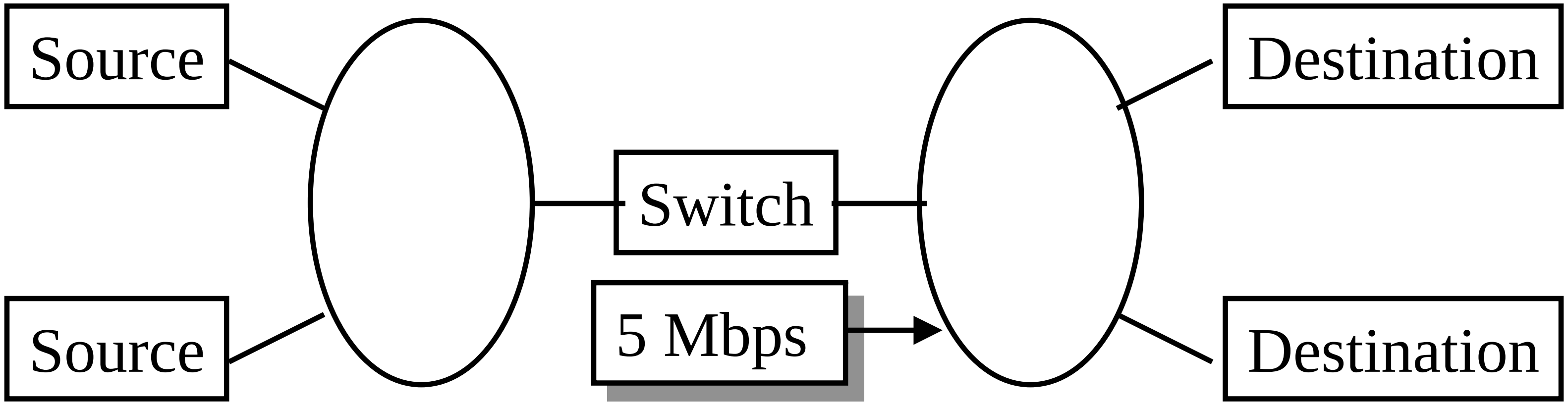


- Periodically, the sources piggyback a "Rate Discovery Tag" (RD tag) on the outgoing packet.
- The tag contain only rate, Rate limiting Q ID, and direction. (Direction = Forward (discovery) tag or Returning tag)
- The sender initializes the RD tag with rate=-1 ($\Rightarrow\infty$)
- The switches adjust the rate down if necessary
- The receiver copies the forward RD tag in a control packets in the reverse direction
- Source adjusts to the rate received

FECN: Observations

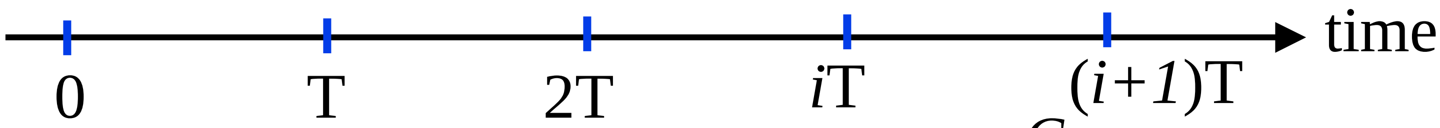
- This is similar to what is done in TCP/IP, Frame Relay, ATM with 1 bit in every packet (n=1).
- ATM ABR had a similar explicit rate indication that was selected after 1 year of intense debate and scrutiny.
- Only the feedback format has to be standardized
- No need to standardize switch algorithm.
- Vendor differentiation: Different switch algorithms will "inter-operate" although some algorithms will be more efficient, more fair, and achieve efficiency/fairness faster than others.
- We present a sample switch algorithm and show that it achieves excellent performance.

Switch Algorithm



- The switch use the same "**Advertised Rate**" in all RD tags
- All sources passing through the switch get the same feedback.
- The sources send at the rate received.

The Basic Switch Algorithm



0. Start with an Advertised Rate of r. $r_0 = \frac{C}{N_0}$
   Here C is the link capacity.
1. Measure input rate every T interval
2. Compute overload factor z in the last T interval
3. Change the advertised rate to r/z
4. In every RD tag: set rate to min{rate in tag, advertised rate}
5. Go back to step 1

Although this simple algorithm will work but:

- It will oscillate even if the rate is close to optimal.
- Queues will not be constant ⇒ ***Need a Q Control Fn***

Enhancement 1: Queue-Control

1. **Measurement**: Let A_i be the measured arrival rate in bits/s then the load factor is $z = A_i/C$. We update this load factor based on the queue length so that the ***effective load factor*** is:

$$\rho_i = \frac{z}{f(q_i)} = \frac{A_i}{f(q_i) \times C}$$

2. **Bandwidth Allocation**:

$$r_{i+1} = \frac{r_i}{\rho_i}$$

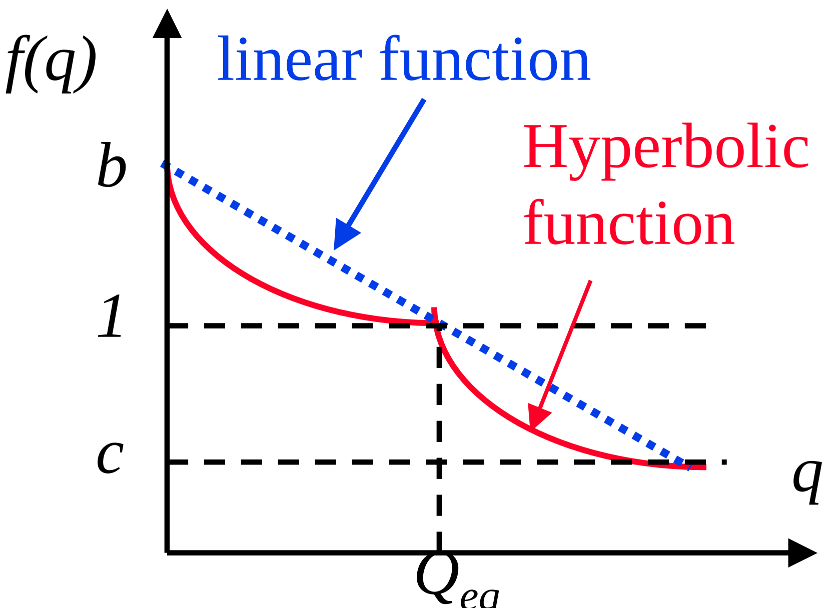
Note: We also tried additive queue control. It has similar performance.

Queue Control Function: *f(q)*

Idea: Give less rate if queue length is large and more if queue length is small compared to desired queue length of Q_{eq} and $f(Q_{eq})=1$

$$f(q) = \begin{cases} \geq 1 & q \leq Q_{eq} \\ = 1 & q = Q_{eq} \\ \leq 1 & q \geq Q_{eq} \end{cases}$$

Reserves some capacity for draining the queue.



We analyzed many different functions and recommend the hyperbolic function because it gives smaller oscillations. [See reference]

Queue Control Function (Cont)

- **Linear Function**: k is some constant

$$f(q) = 1 - k\frac{q - Q_{eq}}{Q_{eq}}$$

- **Hyperbolic function**: a, b, c are constants. Pre-computed in a table.

$$f(q) = \begin{cases} \dfrac{bQ_{eq}}{(b-1)q + Q_{eq}}, & if\ \ q \le Q_{eq}; \\ \max\left(c, \dfrac{aQ_{eq}}{(a-1)q + Q_{eq}}\right), & otherwise. \end{cases}$$

q	f(q)

In all simulations, a = 1.1, b = 1.002, c = 0.1

Enhancement 2: Exponential Averaging

Exponentially weighted average in the Switch:

$$r_{i+1} = \alpha \frac{r_i}{\rho_i} + (1-\alpha) r_{i-1}$$

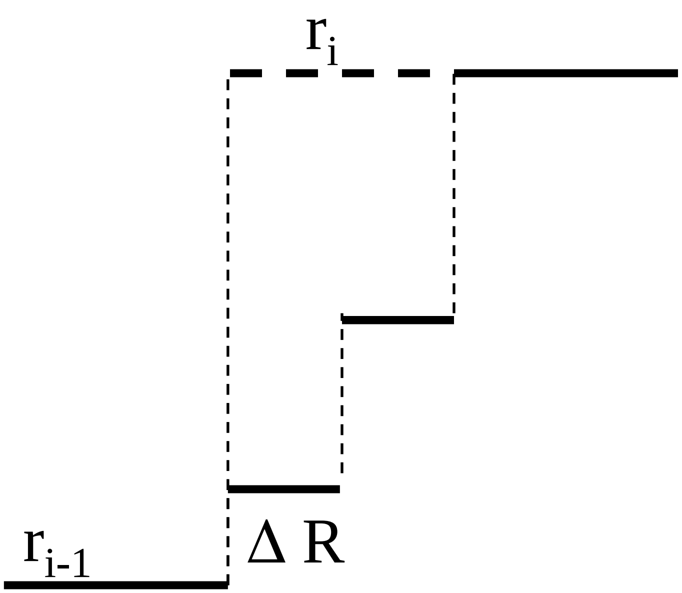
$$\alpha \in (0,1)$$

Remembers recent history. In all simulations α = 0.5

Enhancement 3: Limited Rate Increase

Limit rate increase in the switch

$$
\begin{aligned}
&\text{IF } (q < Q_{eq}) \text{ THEN} \\
&\quad \Delta R = 1.414 \Delta R \\
&\text{ELSE IF } (q > Q_{sc}) \text{ THEN} \\
&\quad \Delta R = 0.707 \Delta R \\
&\text{END IF} \\
&\text{IF } (\ r_i - r_{i-1} > \Delta R\ ) \text{ THEN} \\
&\quad r_i = r_{i-1} + \Delta R \\
&\text{END IF}
\end{aligned}
$$

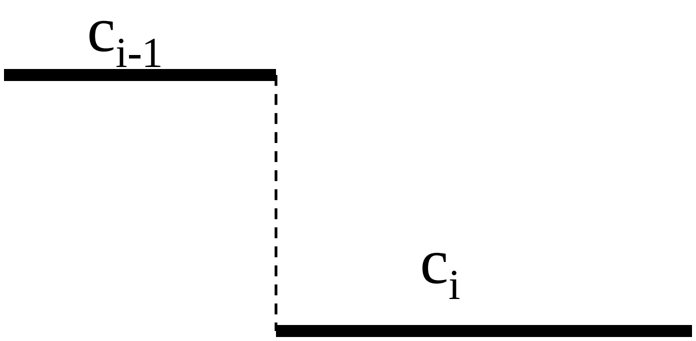


- **Strategy**: Take small jumps. Jump size increases with every step. Results in fast rise time but avoids sudden queue increase if false signal.

Enhancement 4: Variable Capacity Adjustment

- If capacity of the link reduces due to failure of a component link in an aggregated link or other reasons, the allocated rate is reduced accordingly.

$$
\begin{aligned}
&\text{IF } (c_i < c_{i-1}) \text{ THEN} \\
&\quad r_i = (c_i / c_{i-1}) r_i \\
&\quad r_{i-1} = (c_i / c_{i-1}) r_{i-1} \\
&\text{END IF} \\
&r_{i-2} = r_{i-1} \\
&r_{i-1} = r_i
\end{aligned}
$$



Enhancement 5: Time Based Sampling

- **Time-based sampling at the source**: Packet tagged if time since the last time tag was sent is more than τ
- In all simulations $\tau = T$

General Simulation Parameters

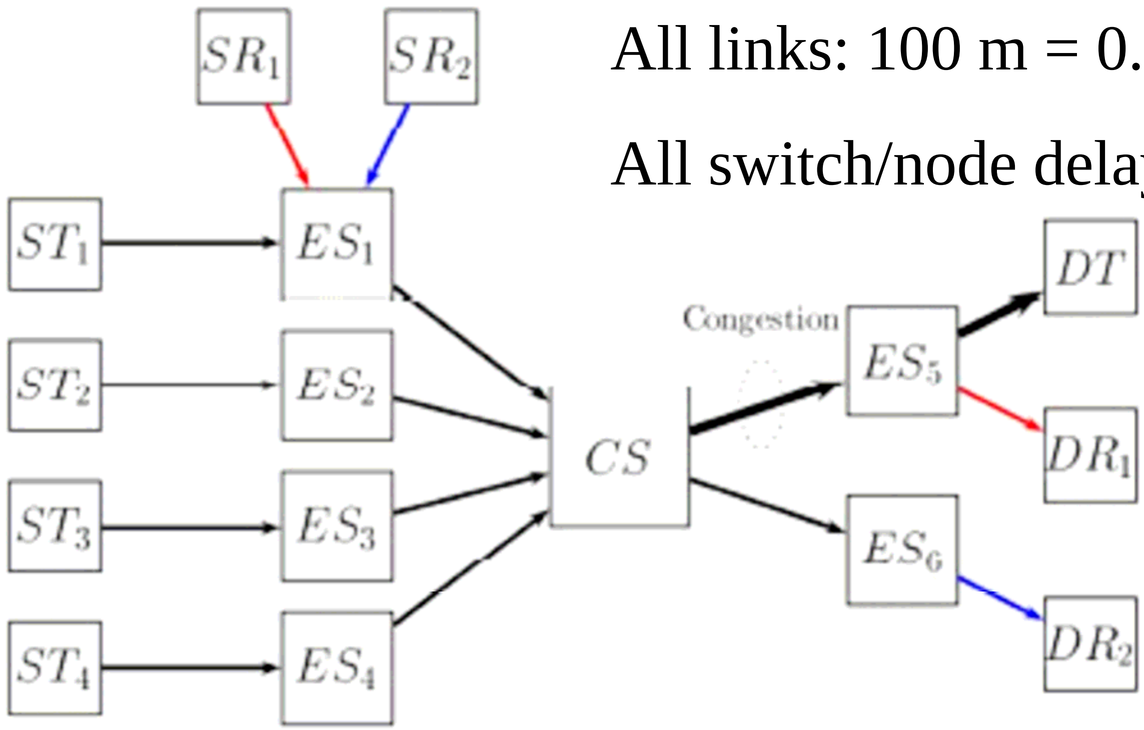
- Queue control function: Hyperbolic
- Packet size = 1500 B
- Measurement interval T = 1 ms

Baseline Simulation Results

1. FECN with TCP flows
2. Symmetric Topology
3. **Large** Topology with 100 flows
4. **Bursty Traffic**: Pareto-distributed burst time
5. Output-Generated **Hot-Spot** Scenario
6. Output-Generated Hot-Spot Scenario with **long delay**

All simulations use the same parameter values!

FECN with TCP flows



All links: 100 m = 0.5 us

All switch/node delays = 1 us

- 6-source topology
- SR1-to-DR1 and SR2-to-DR2 are *reference flows*
- SR_i-to-DT are four flows that share the bottleneck link

FECN with TCP flows (Cont)

- *T* = 1 ms
- Total simulation time = 1 sec
- Workload
  - ST1-ST4: 10 parallel TCP connections transferring 1 MB each continuously
    1 Transaction = 1 MB transfer
  - Reference flows: 1 TCP connection transferring 10kB each with average idle time of 16 us for SR1 and 1 us for SR2

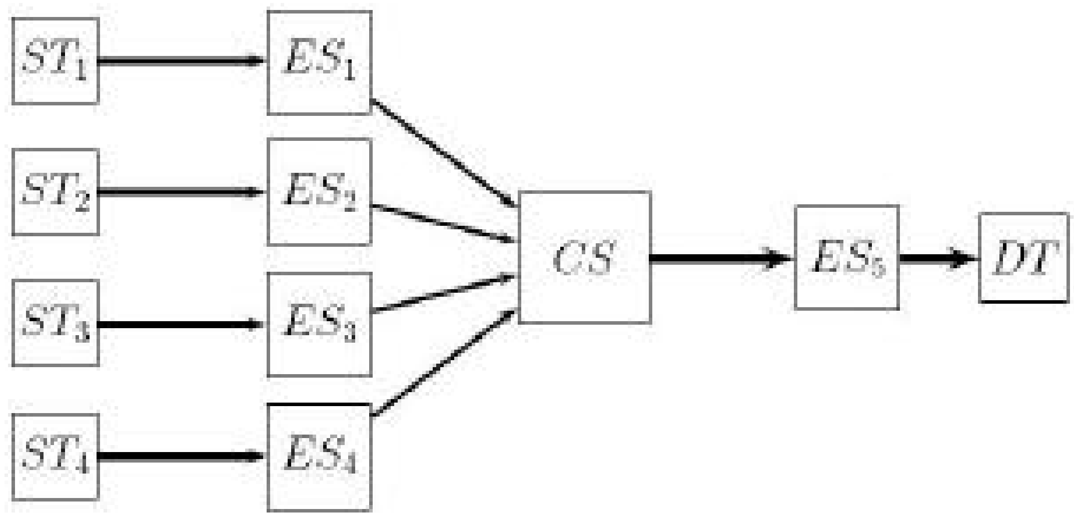
Simulation Results

	Reference Flow 1			Reference Flow 2		
Congestion Mechanism	Throughput (Transactions/s)	Throughput (Gbps)	Transaction Completion Time (us)	Throughput (Transactions/s)	Throughput (Gbps)	Transaction Completion Time (us)
None	556	0.06	1780.78	16634	1.44	59.11
FECN	6970	0.604	127.63	16630	1.44	59.16

Congestion Mechanism	Average Throughput (Gbps)	Jain Fairness Index	Link Utilization (%)
None	2.49	2%	99.9
FECN	2.35	99%	99.9

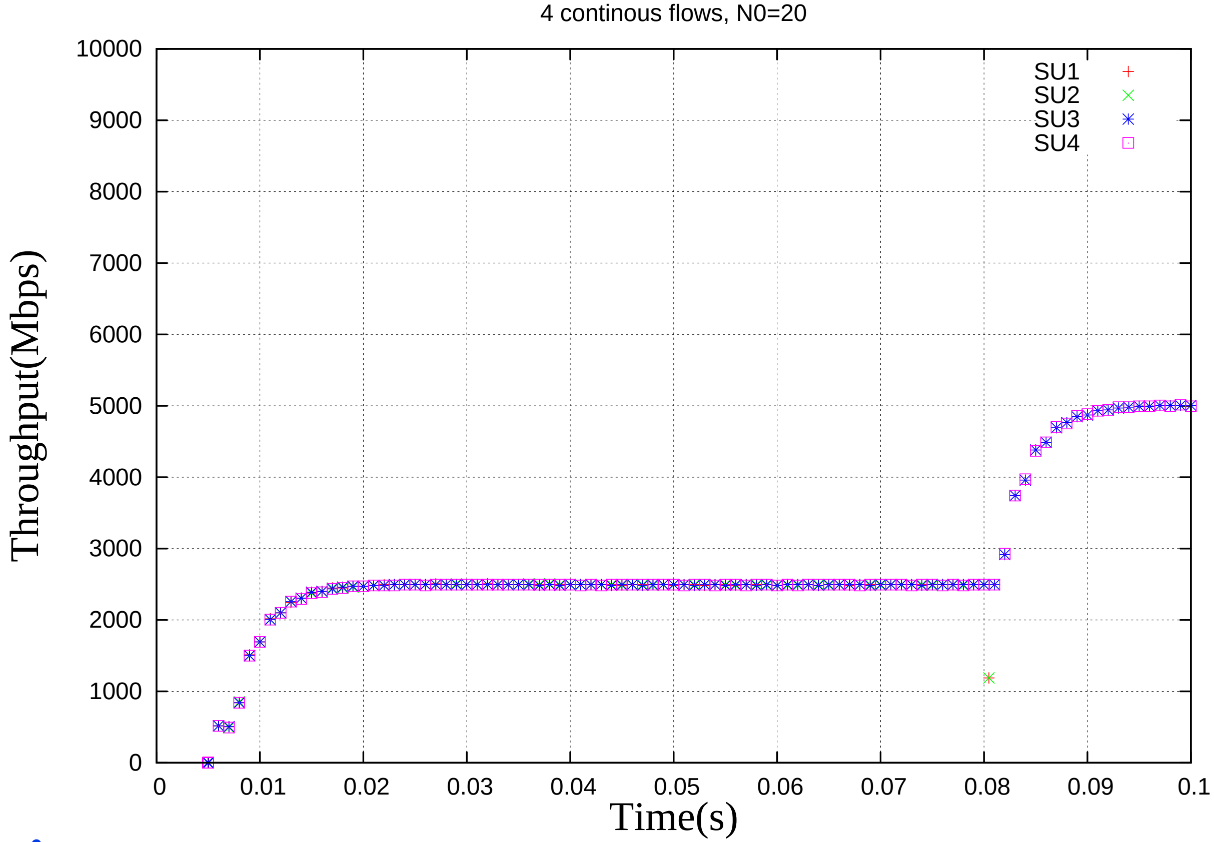
Conclusions: FECN can protect fragile TCP flows and improve its goodput and fairness significantly. FECN reduced packet loss from 50,008 packets to 0 packets.

Symmetric Topology: Configuration



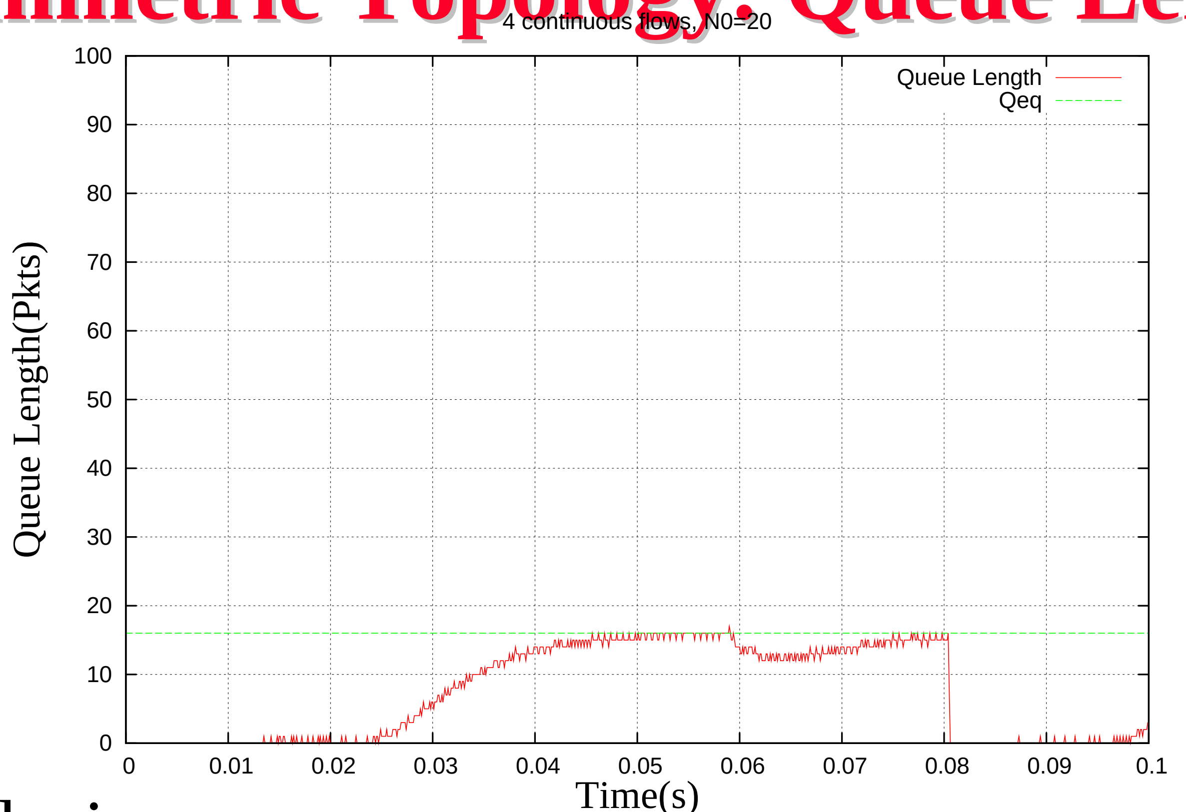
- UDP *Bernoulli Traffic* with average *5* Gbps rate
- Measurement Interval T is *1* ms, N0 = 20
- Simulation Time is *100* ms, all sources starts at *5* ms
- At *80* ms, 2 sources stop
- Per-hop-delay=0.5 us, switch/node delay is 1 us

Symmetric Topology: Source Throughput



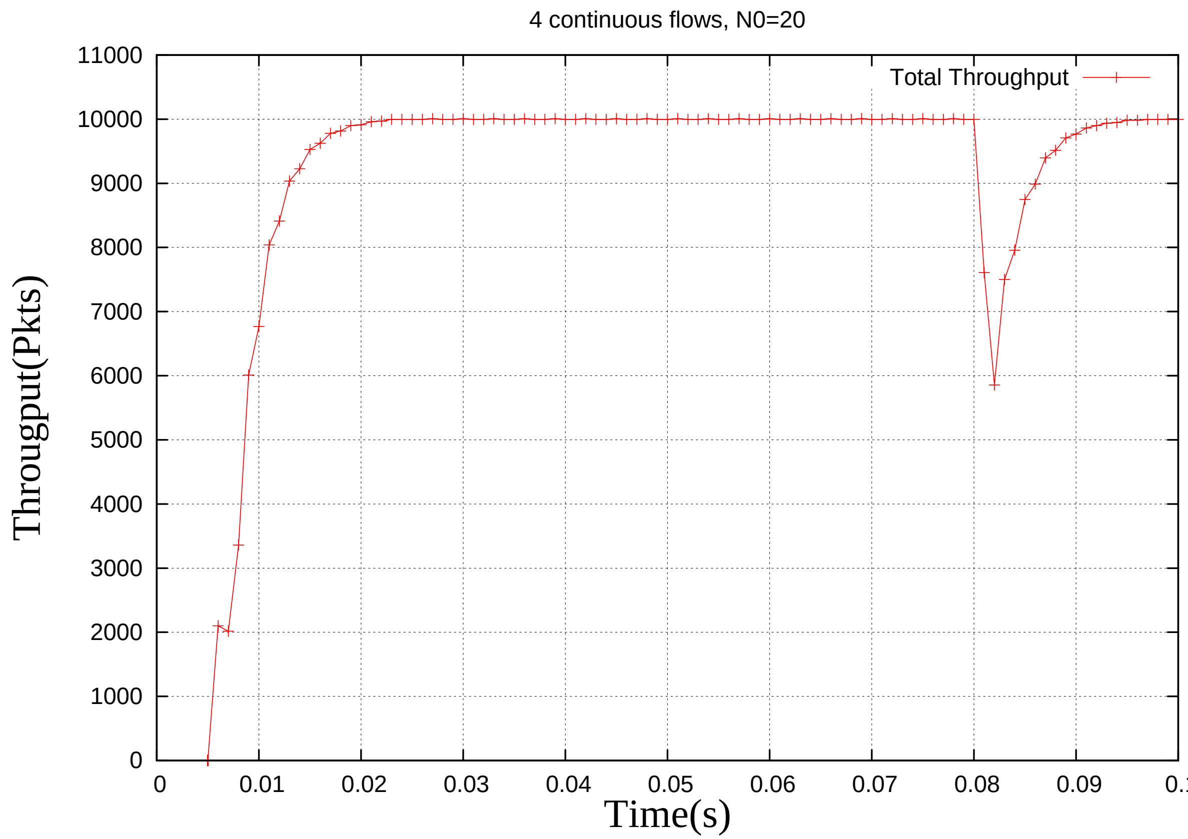
- **Conclusions**:
  - Four sources overlap ⇒ Perfect Fairness!
  - Fast Convergence: around 10 ms

Symmetric Topology: Queue Length



- Conclusions:
  - Queue builds up to Qeq and can stays there.
  - Queue never overflows

Symmetric Topology: Link Utilization

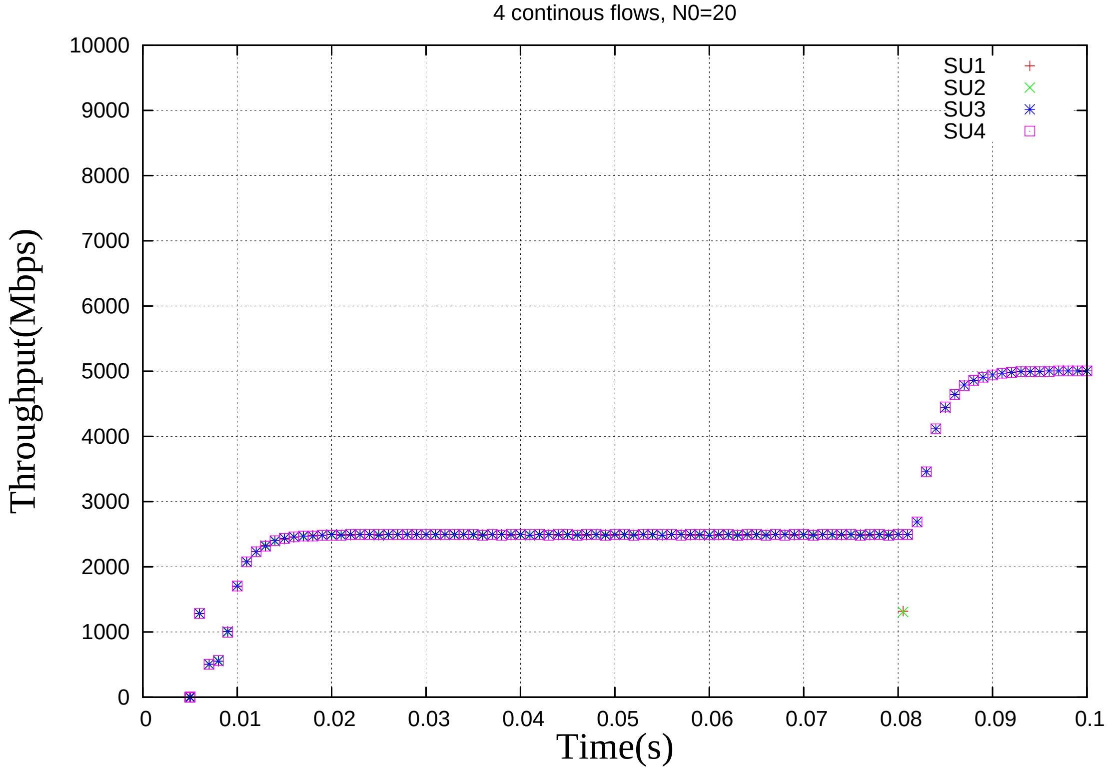


- **Conclusions**: Link is highly utilized when the rate achieves the fair share in around 10 ms.

Simple Topology, 400us Delay

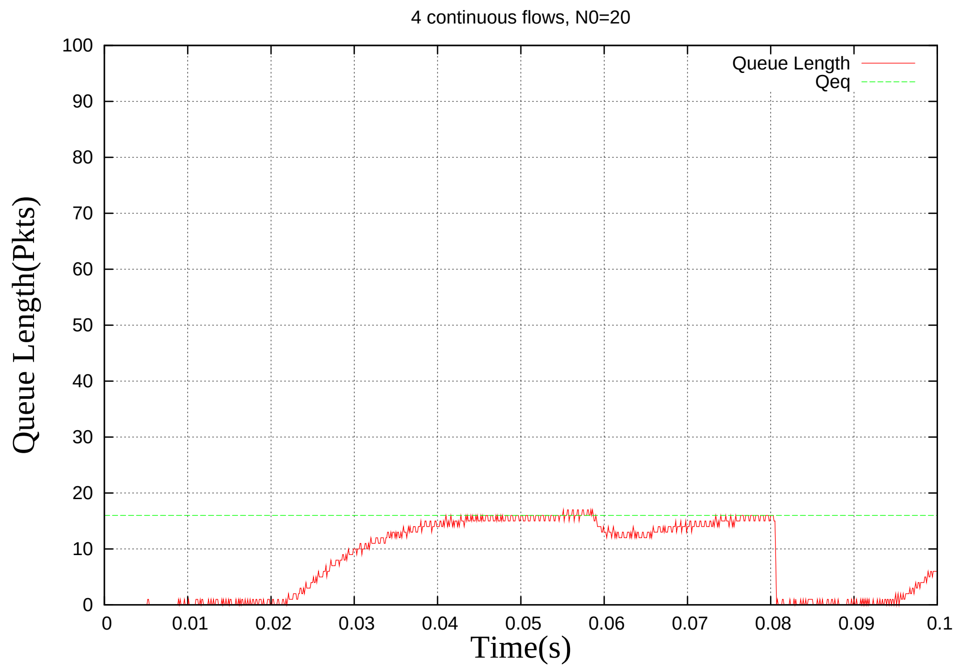
- Control loop delay is 400 us
- Each link and station delay is 50 us
- No very clear effect due to long delay in this simple case

Symmetric Topology LD: Source Throughput



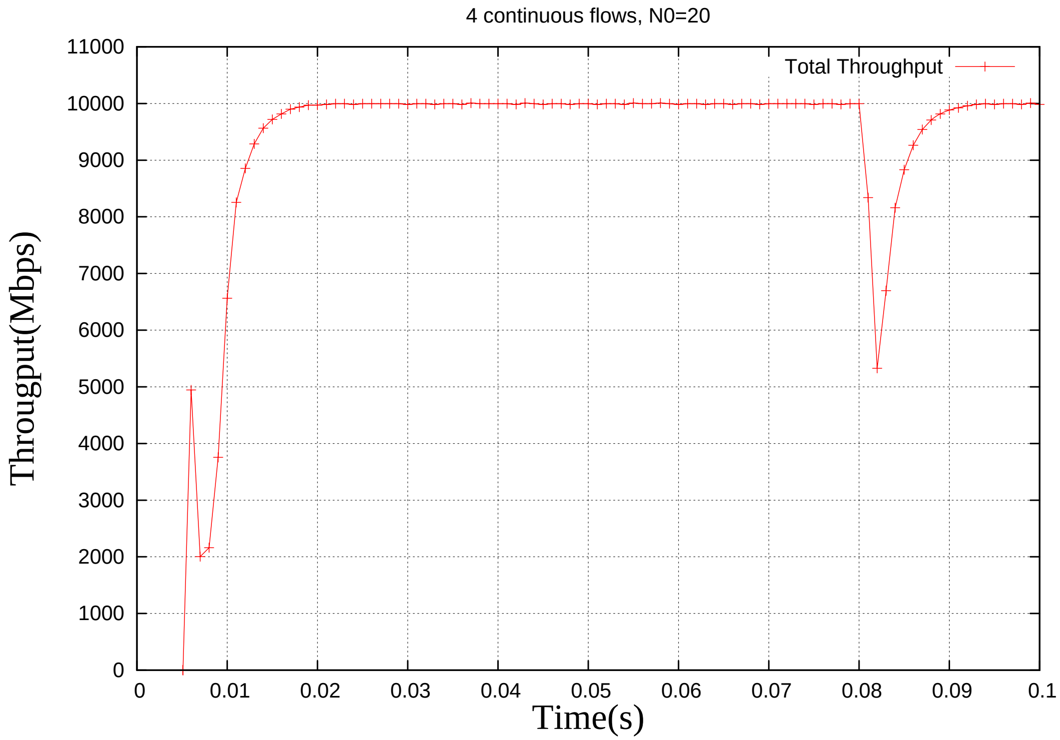
- **Conclusions**:
  - Four sources overlap ⇒ Perfect Fairness!
  - Fast Convergence: around 10 ms

Symmetric Topology LD: Queue Length



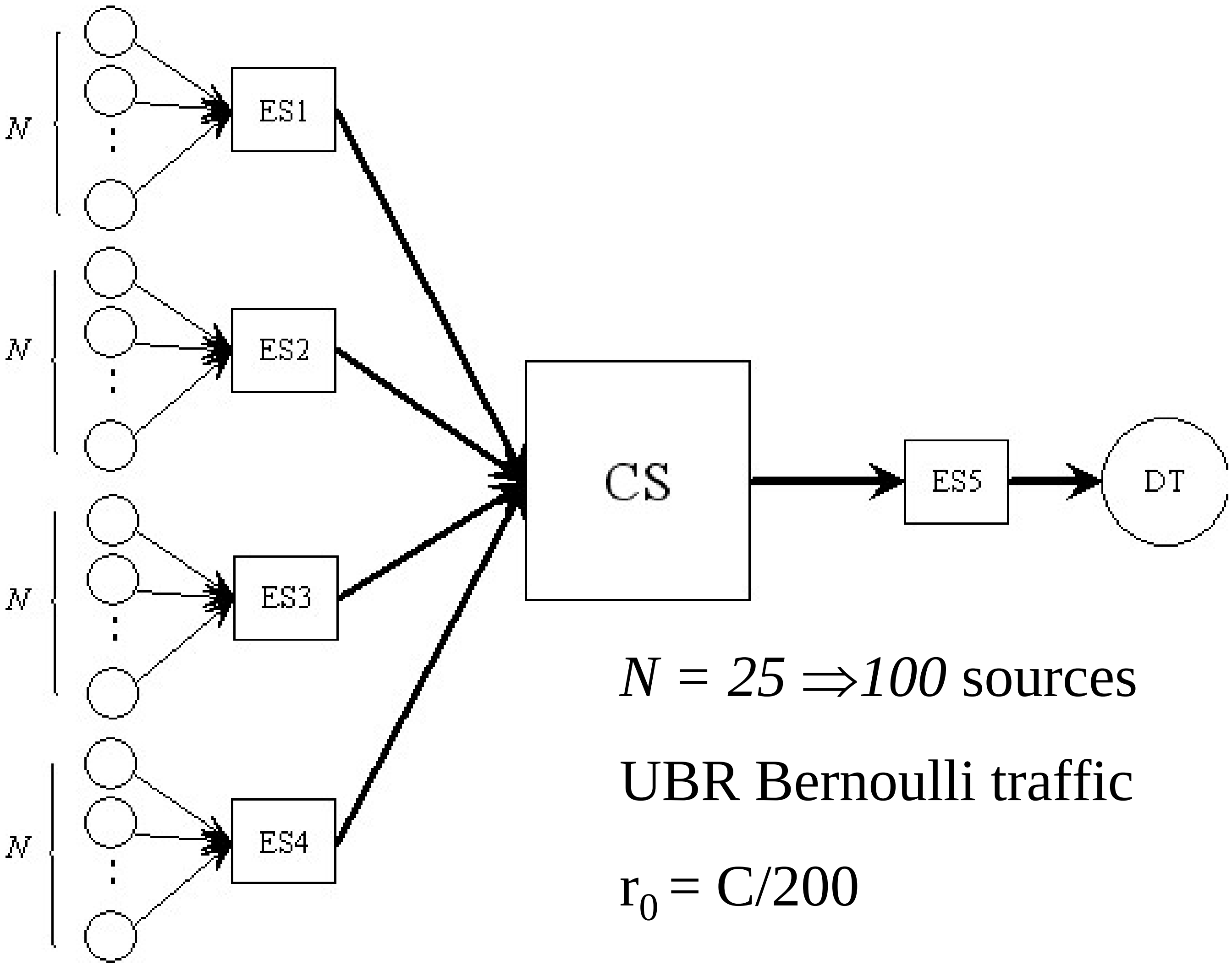
- Conclusions:
  - Queue builds up to Qeq and can stays there.
  - Queue never overflows

Sym. Topology LD: Link Utilization



- **Conclusions**: Link is highly utilized when the rate achieves the fair share in around 10 ms.

Large Topology: Configuration

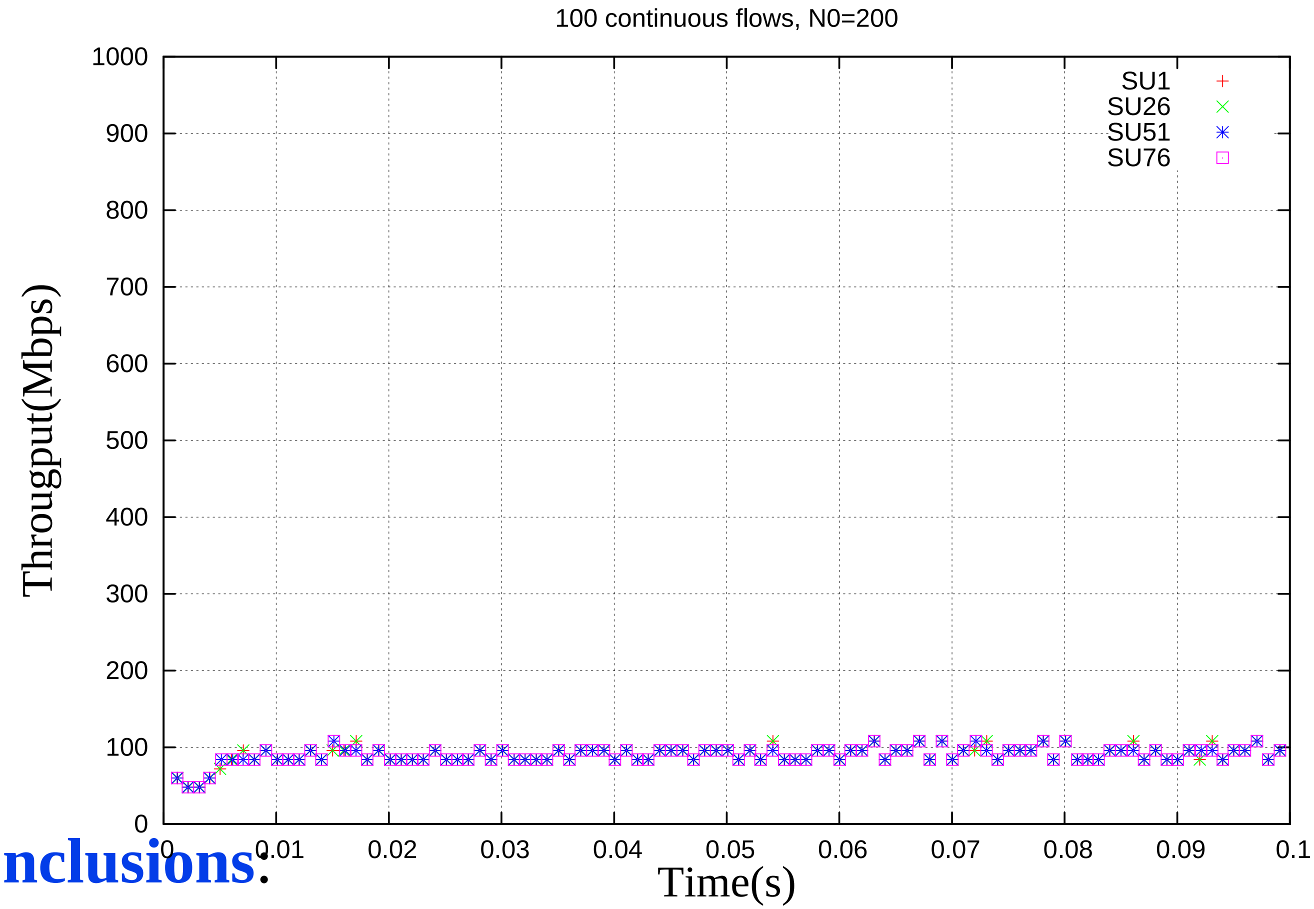


N = *25* ⇒ *100* sources

UBR Bernoulli traffic

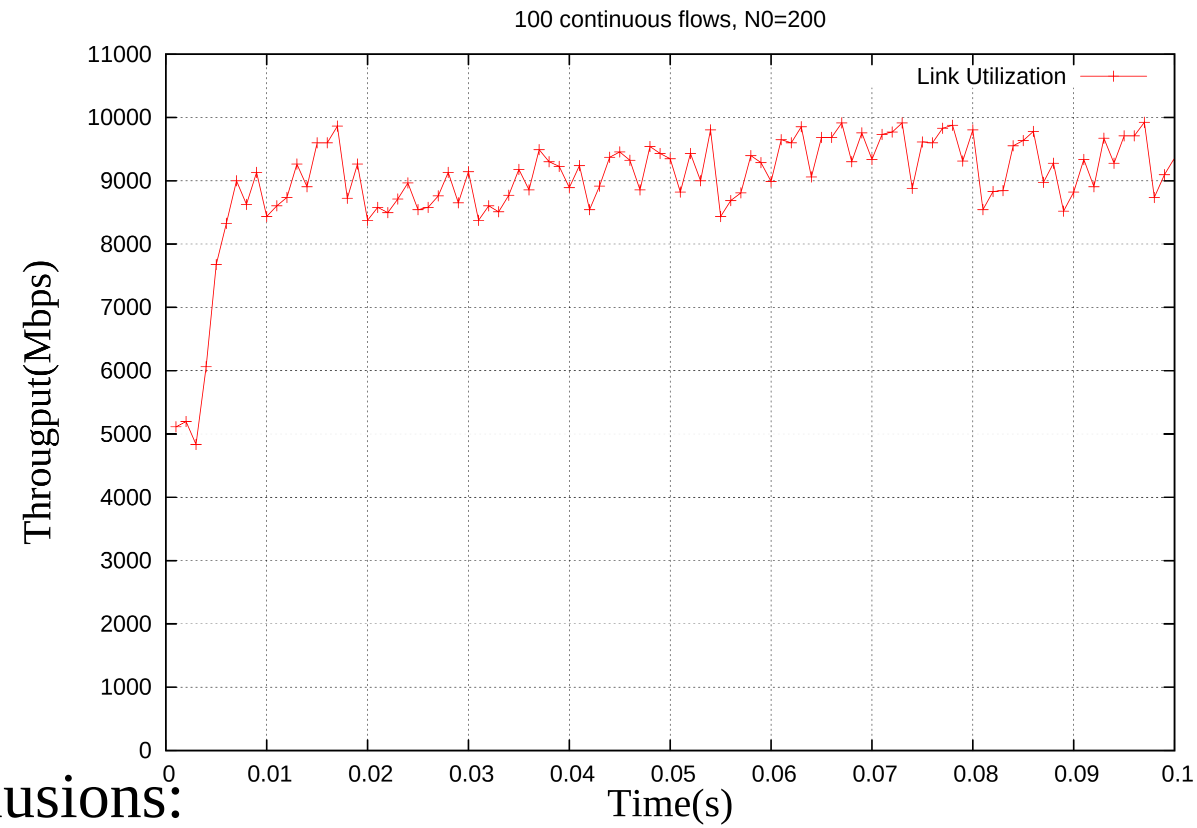
$r_0 = C/200$

Large Topology: Source Rates



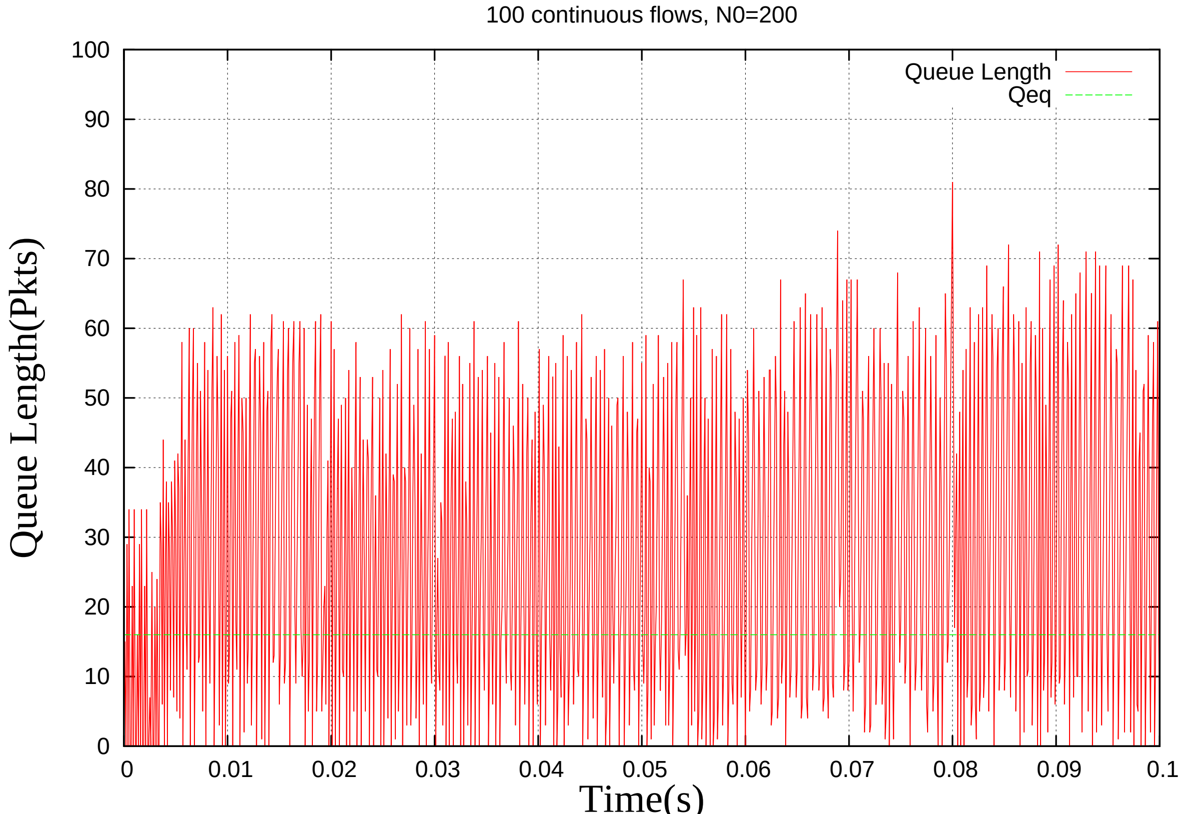
- **Conclusions**:
  - Perfect Fairness!
  - Fast Convergence: less than 10 ms

Large Topology: Link Utilization



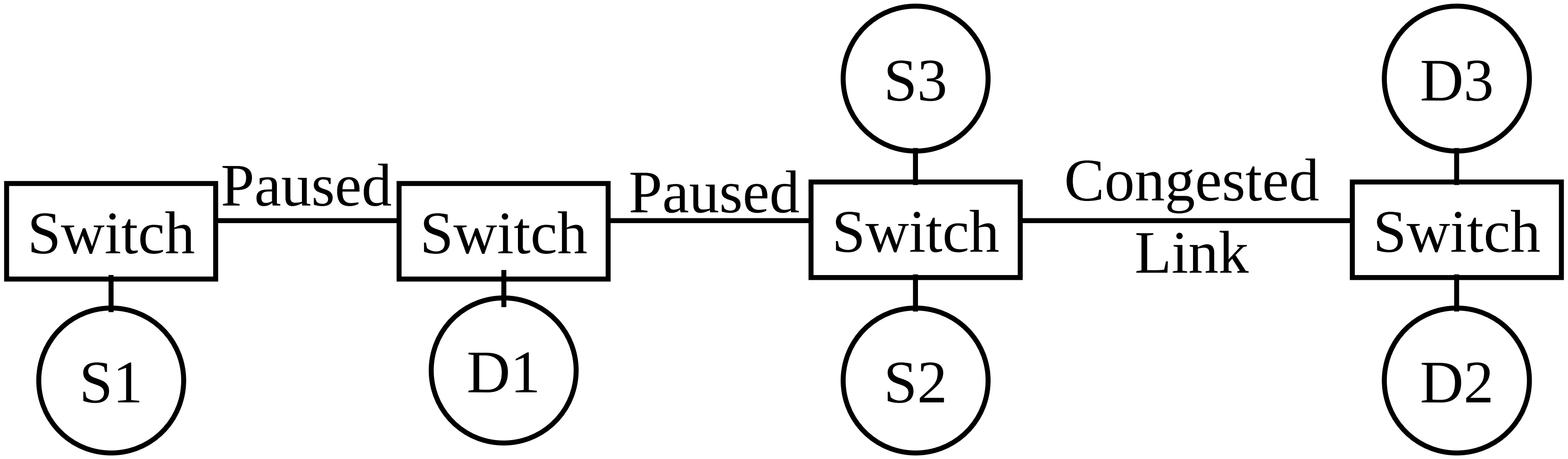
- Conclusions:
  - The link is still 90+% utilized on average

Large Topology: Queue Length



- **Conclusions**:
  - Queue does not overflow!
  - No PAUSE required or issued

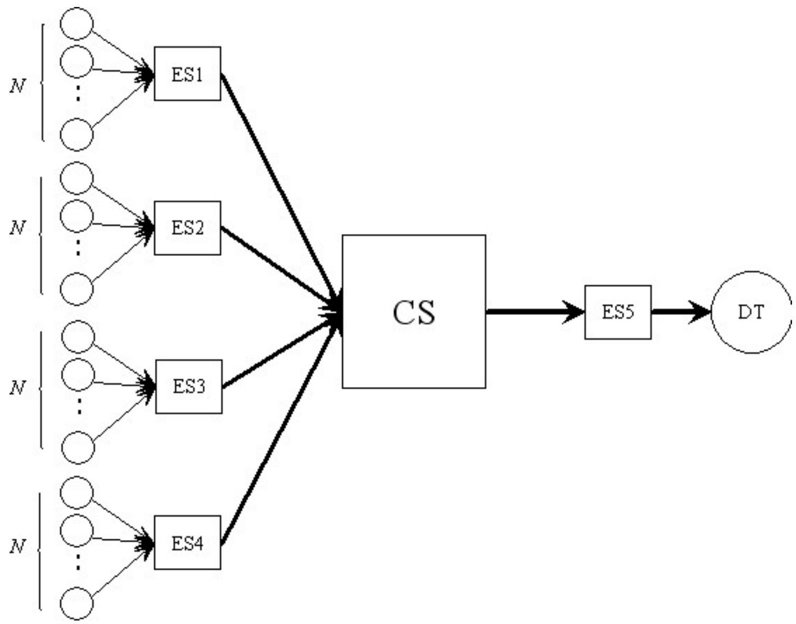
PAUSE



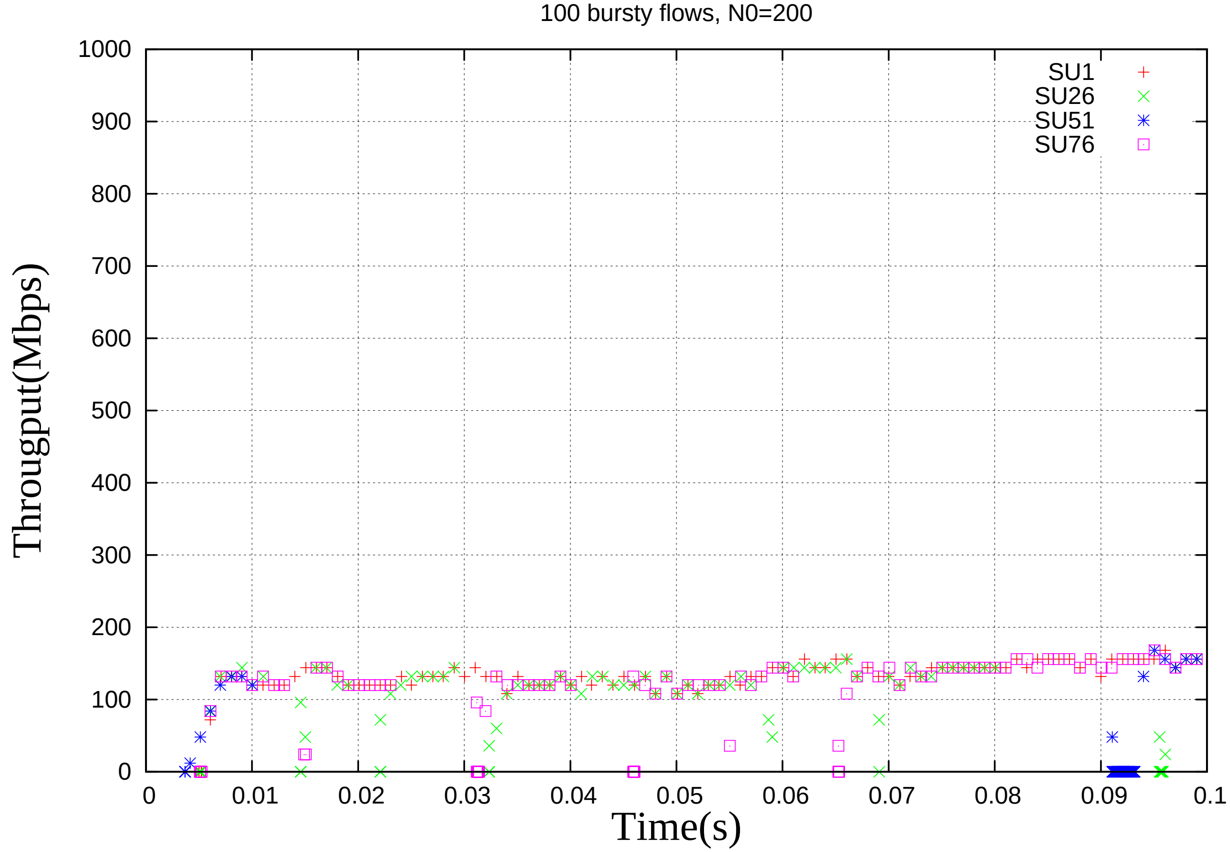
- S1-to-D1 flow is not using congested resources but is stopped by congestion caused by S2-to-D2 and S3-to-D3
- **Conclusion**:
  - Pause unfairly affects non-congestion causing flows
  - Pause should not be used as a primary or frequent mechanism
  - Pause can reduce loss but increase delays in the network
  - Pause is an emergency mechanism for rare use

Bursty Traffic: Configuration

- Large Topology (100 Sources)
- The sources come on and go off after transmitting a burst.
- The ON/OFF period is Pareto distributed
- Average ON/OFF period is *10* ms

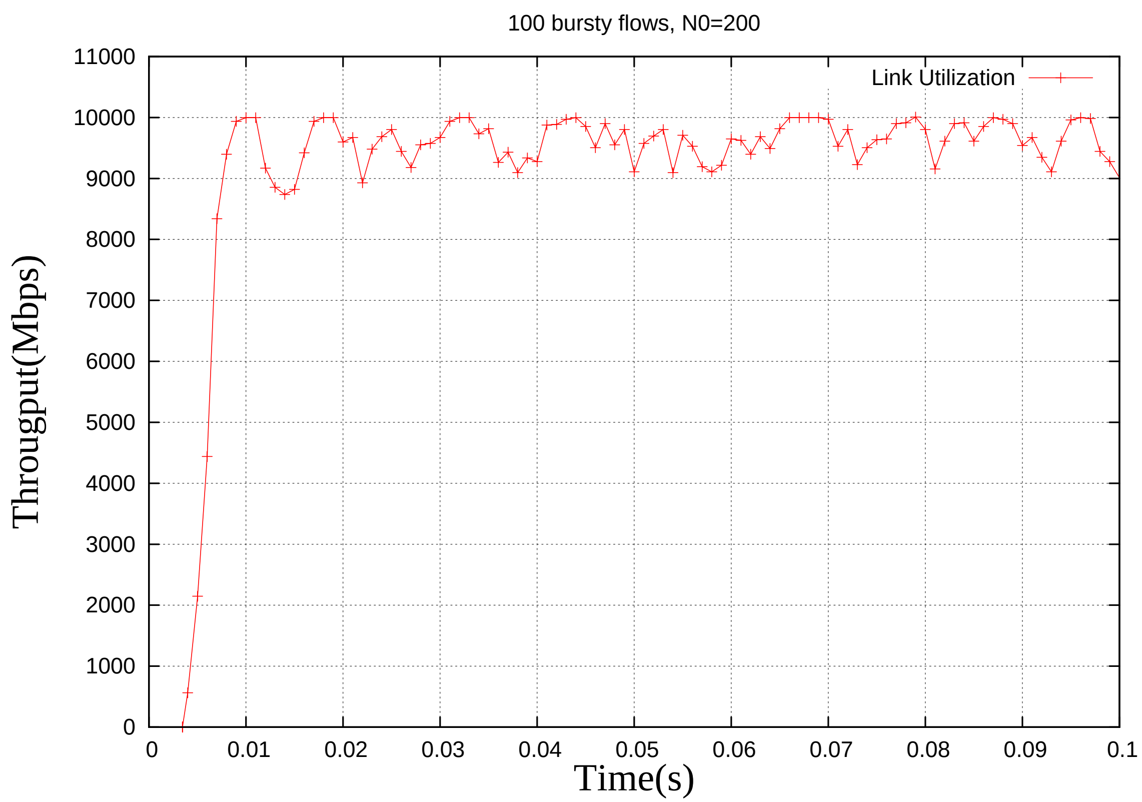


Large Topology - Bursty Traffic: Througput



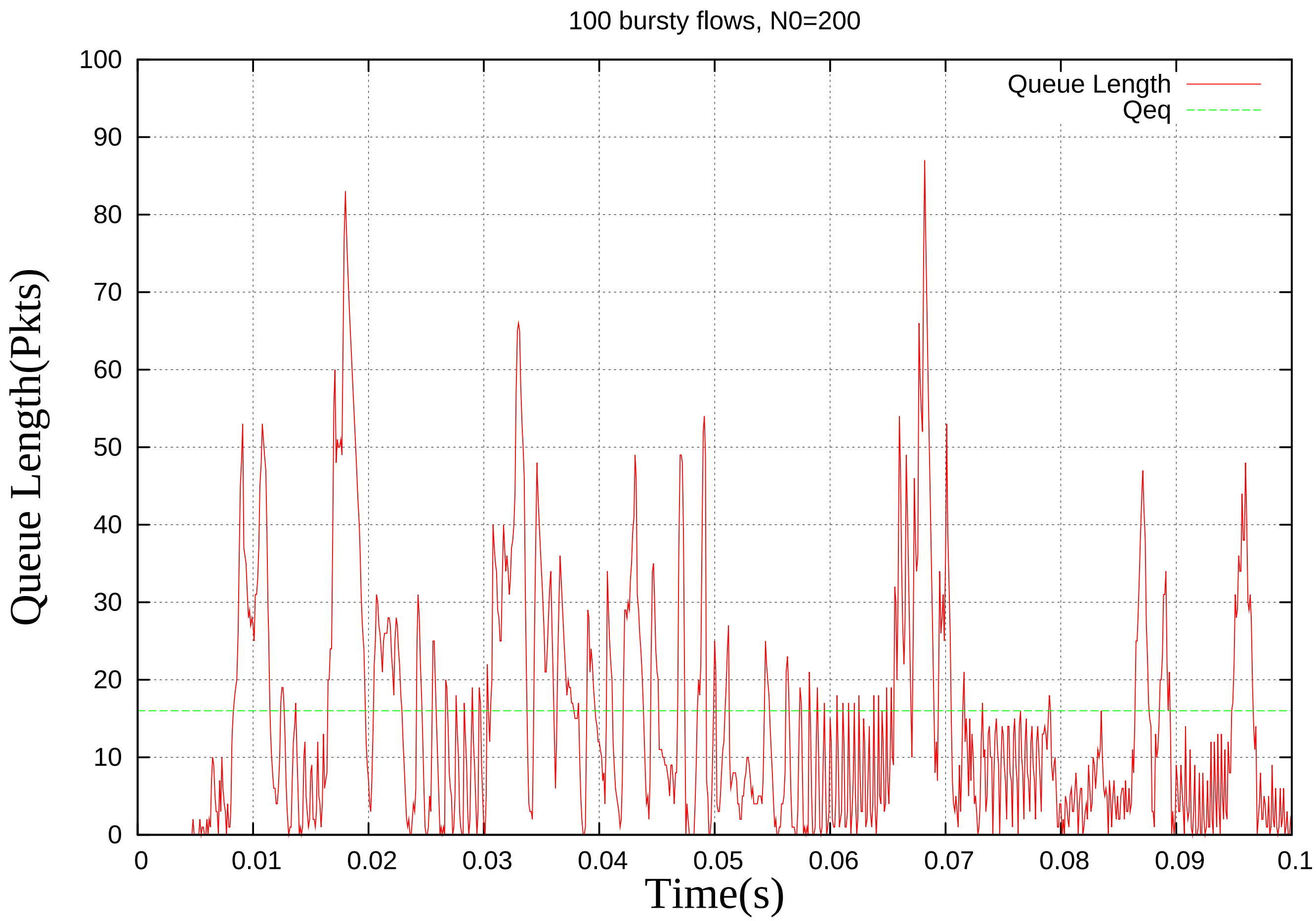
- **Conclusion**: Perfect Fairness!

Large Topology - Bursty Traffic: Link Utilization



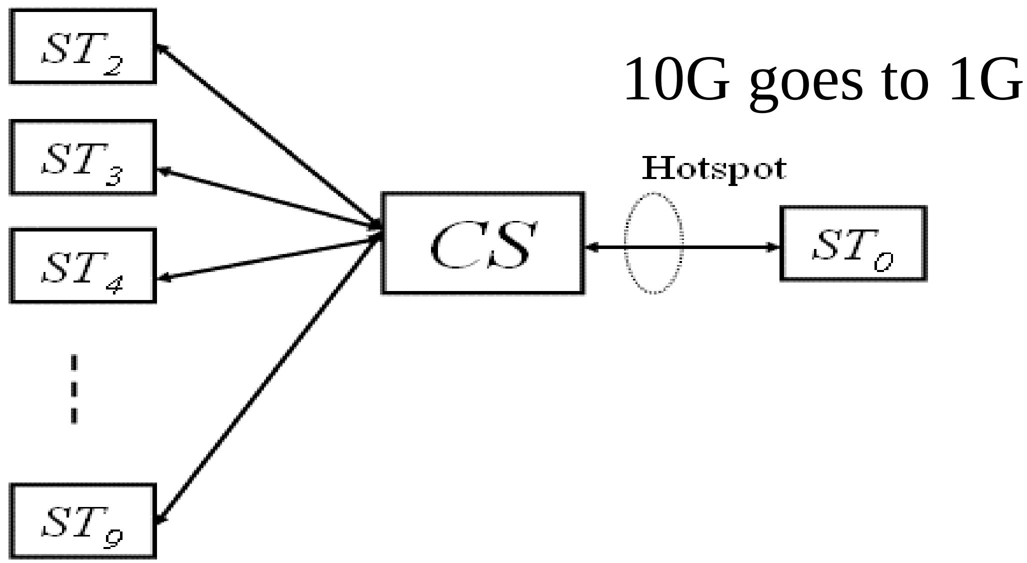
- **Conclusion**: On average, the link is 95+% utilized

Large Topology - Bursty Traffic: Queue



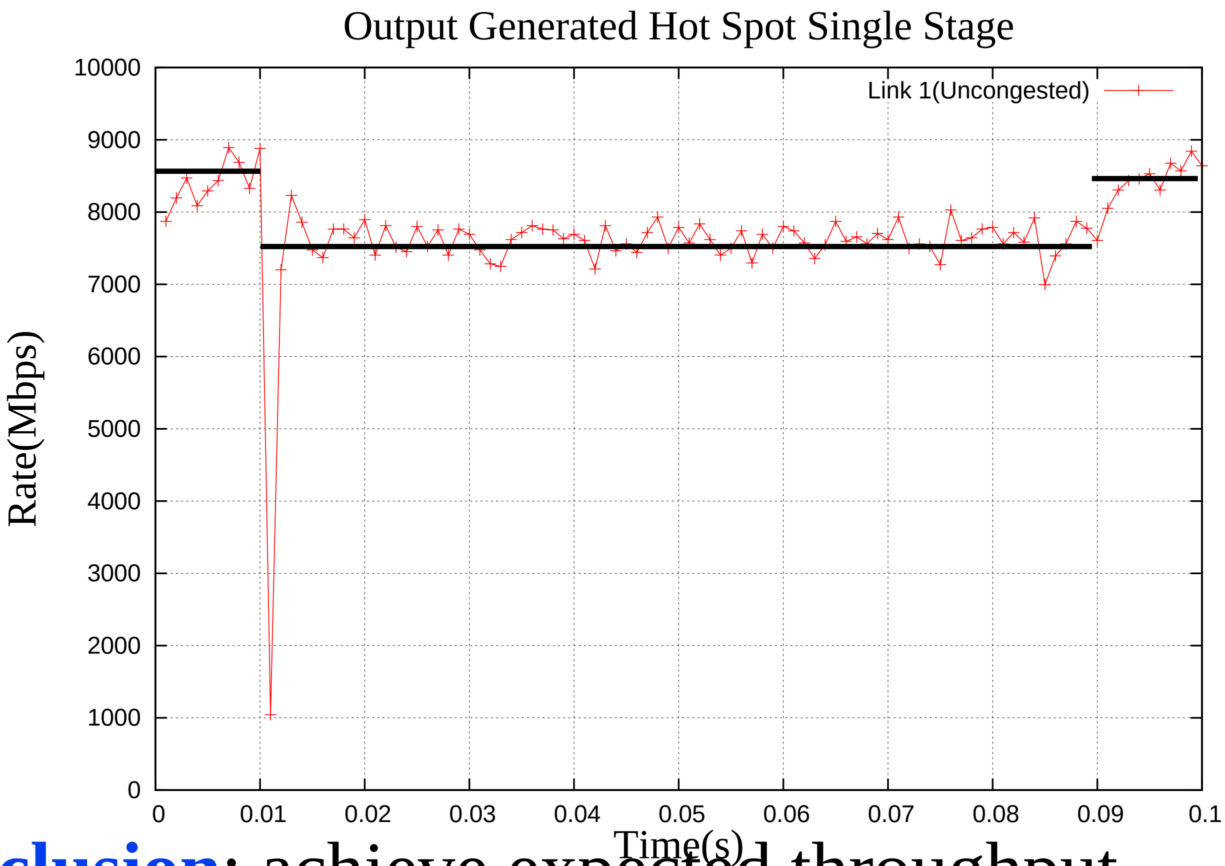
- **Conclusion**: Queue length is always under the buffer size. No pause required for this case.

Output Generated Hotspot Scenario



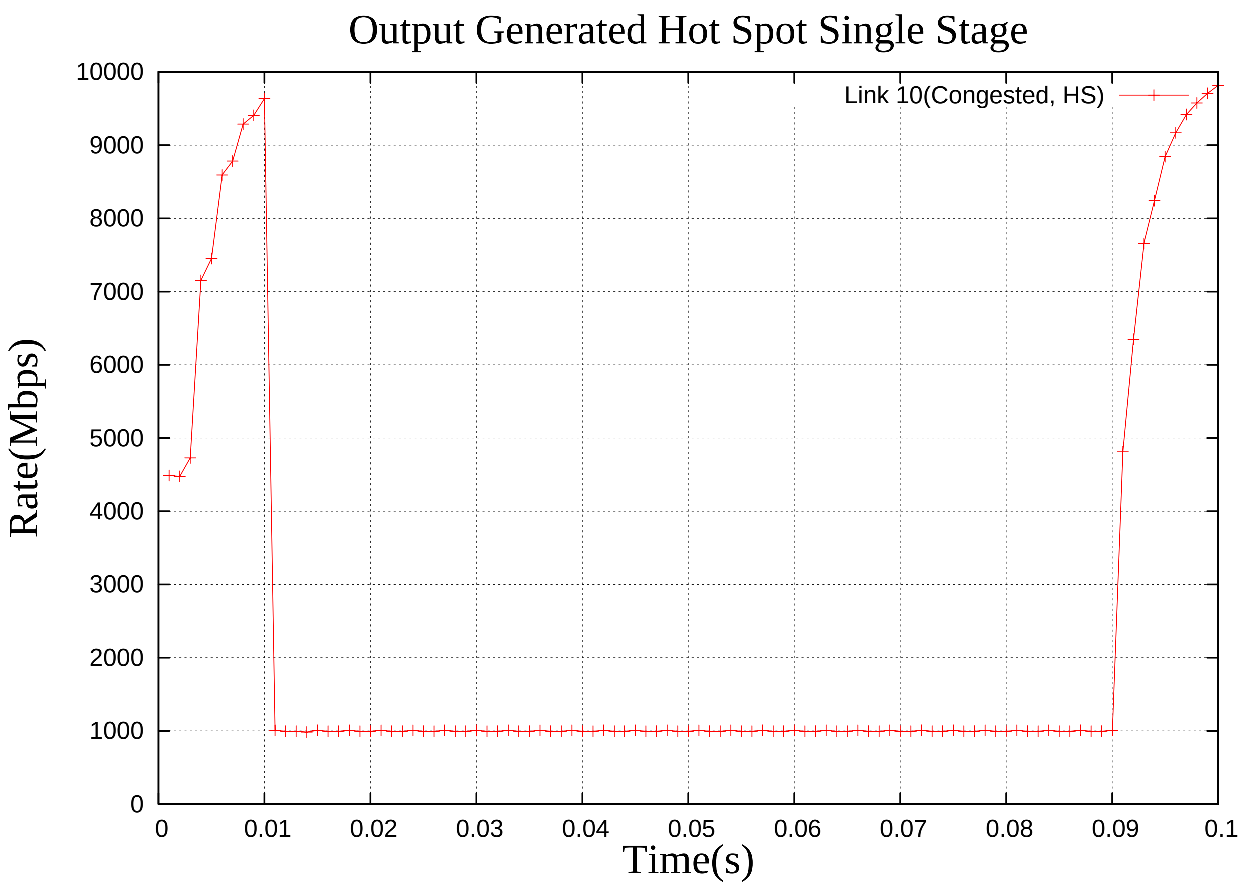
1. Capacity from CS to ST0 goes to *1* G from 0.01s to 0.09 s, then come back to *10* Gbps
2. We study per flow behavior instead of per node behavior
3. Symmetric topology configuration is used
4. Capacity C(t) is known from the idle time and bits transmitted.

OGHS – Uncongested Link (ST2 to CS)



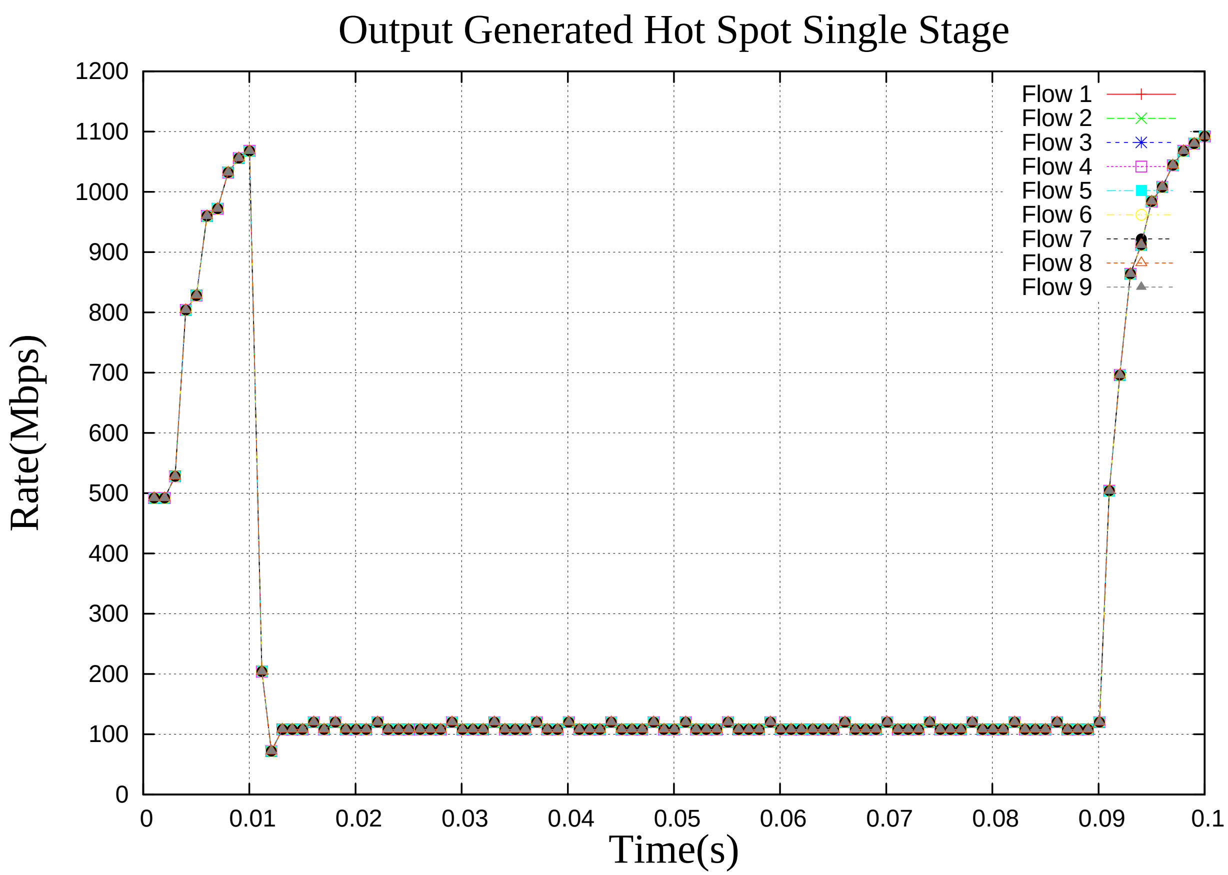
- **Conclusion**: achieve expected throughput

OGHS – Congested Link (CS to ST0)



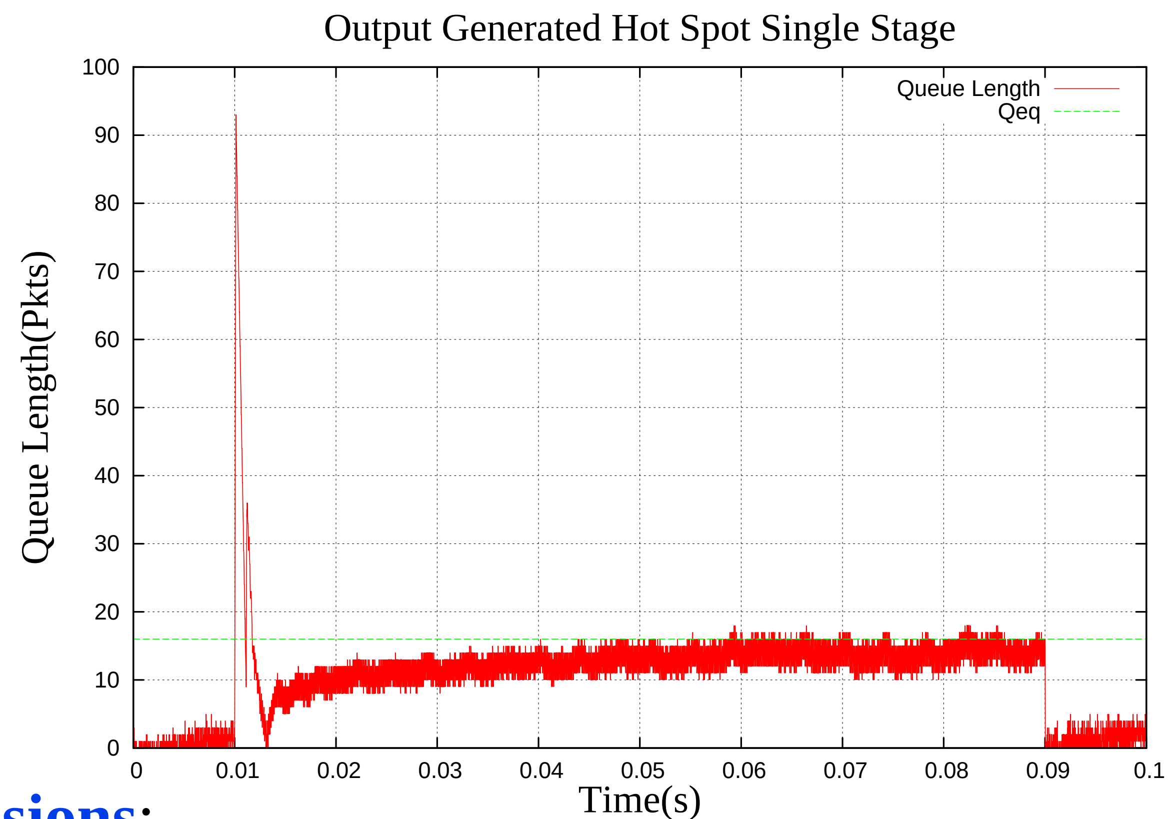
- **Conclusion**: Fast convergence. Highly utilized link!

OGHS – 9 Congested Flows



- **Conclusion**: Perfect fairness among 9 flows!

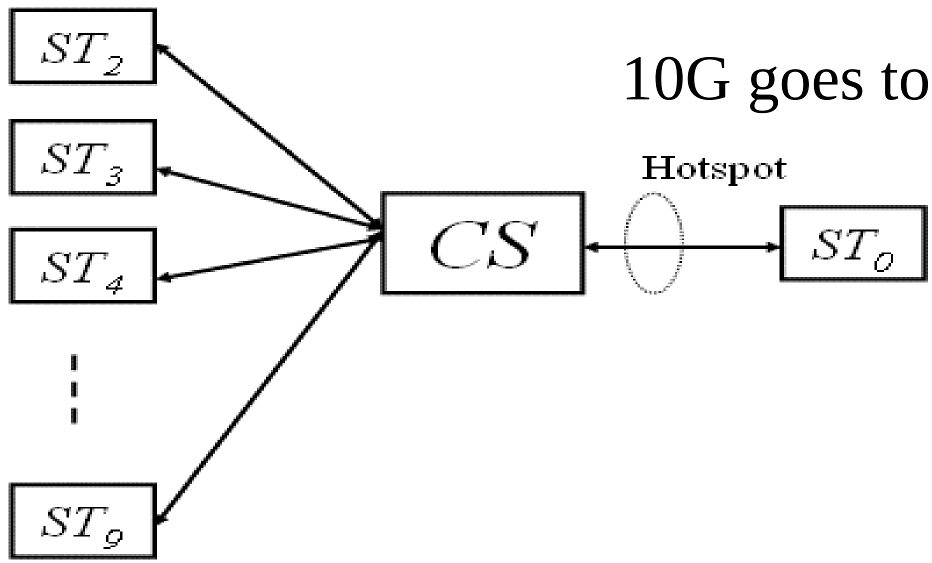
OGHS - Queue Length



Conclusions:

- One very short Pause event (capacity reduced from 10G to 1G).
- The queue is very stable at the equilibrium point

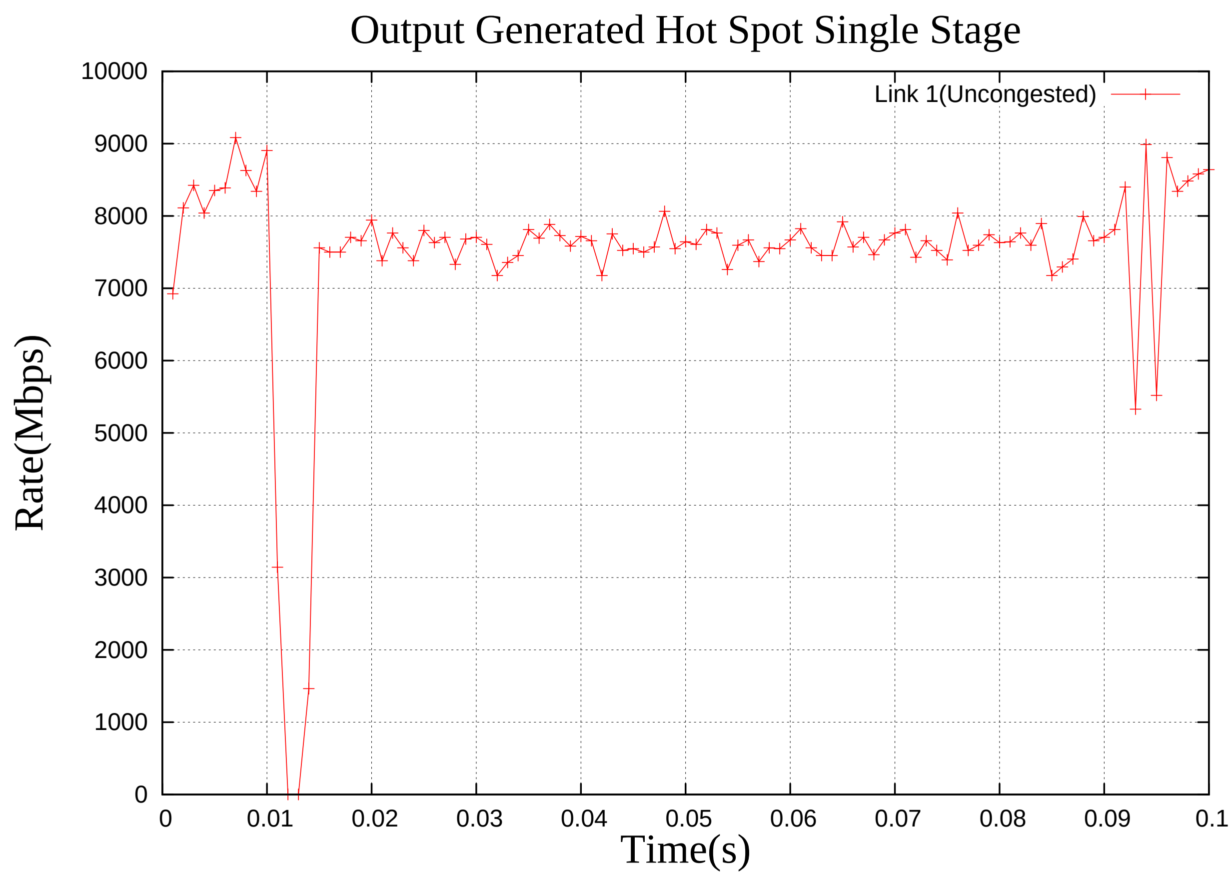
OGHS – Long Delay



10G goes to 1G

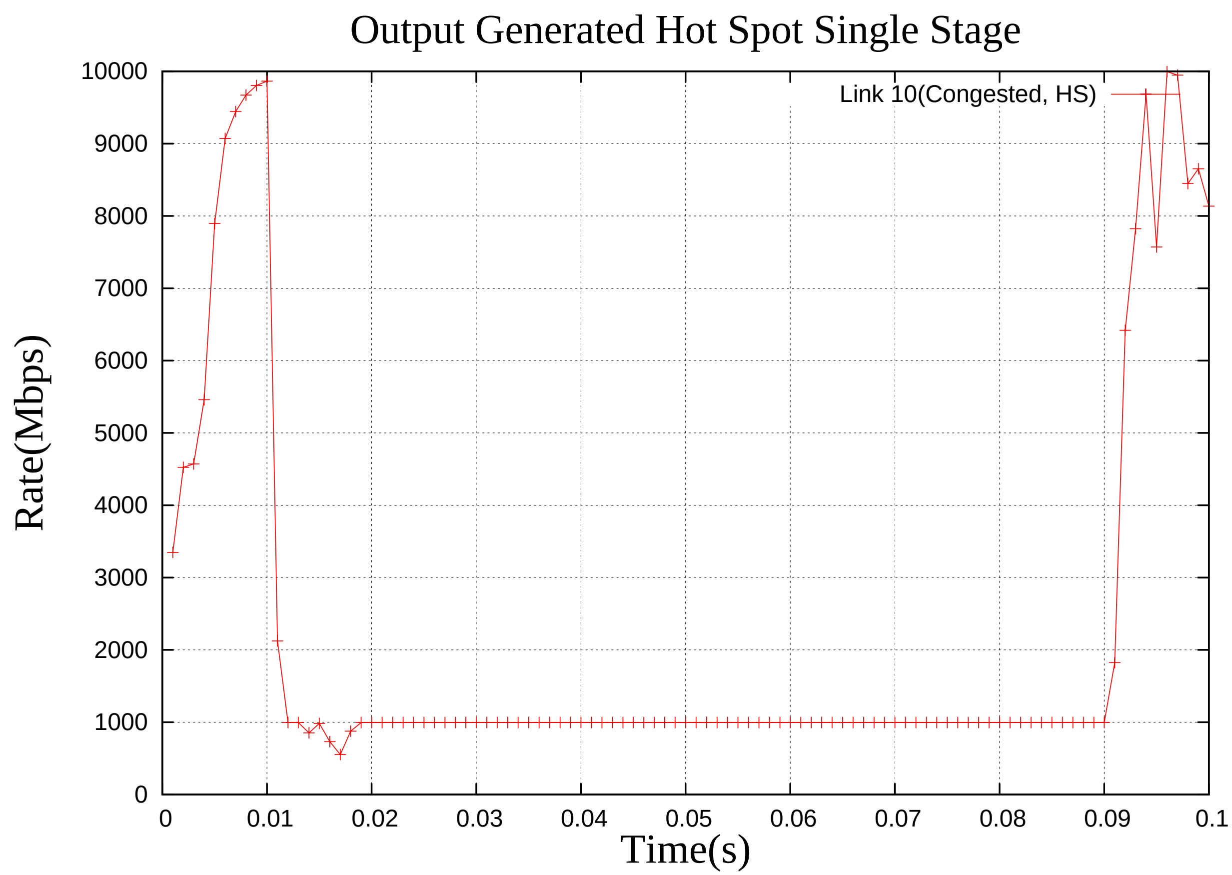
1. Each link ?? us long
2. Total feedback delay = 500 us

OGHS-LD: Uncongested Link - ST2 to CS



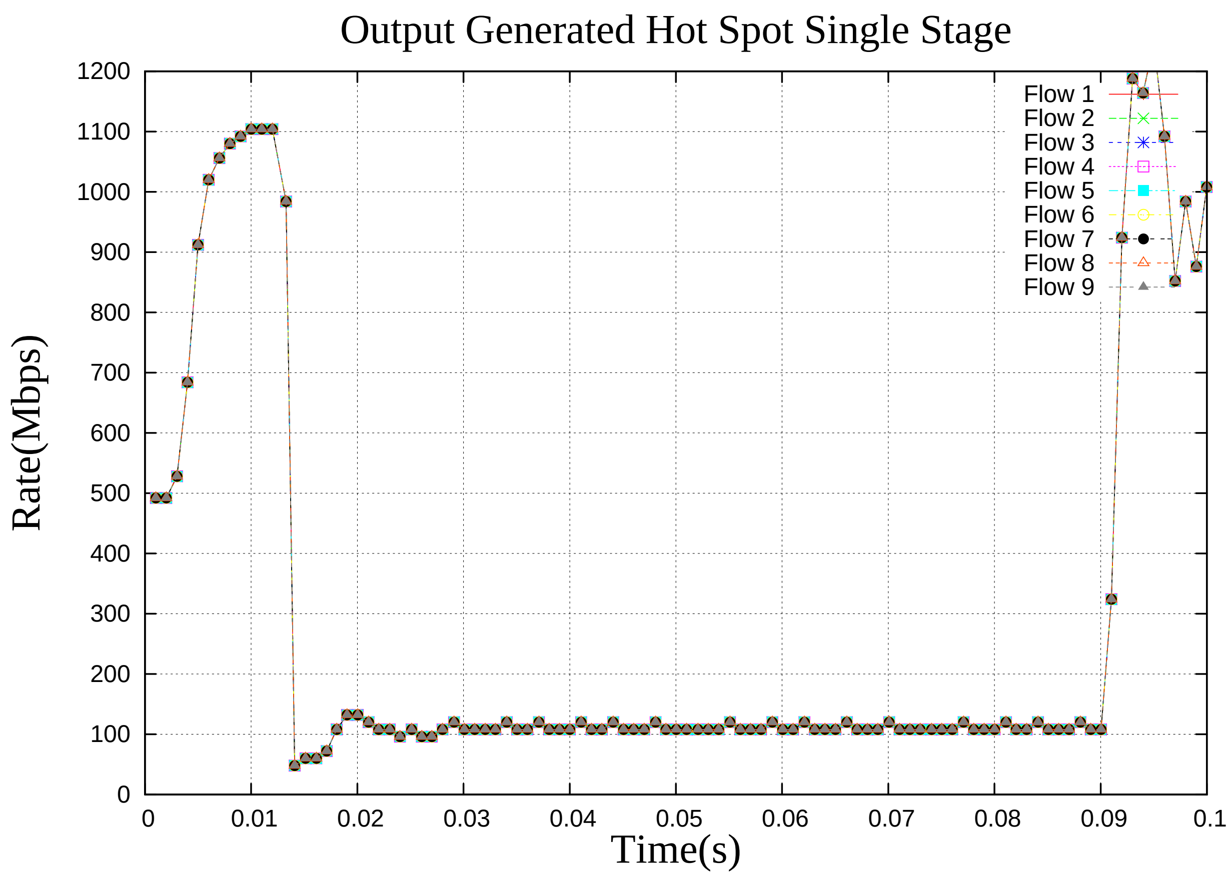
Conclusion: FECN recovers quickly

OGHS-LD: Congested Link (CS to ST0)



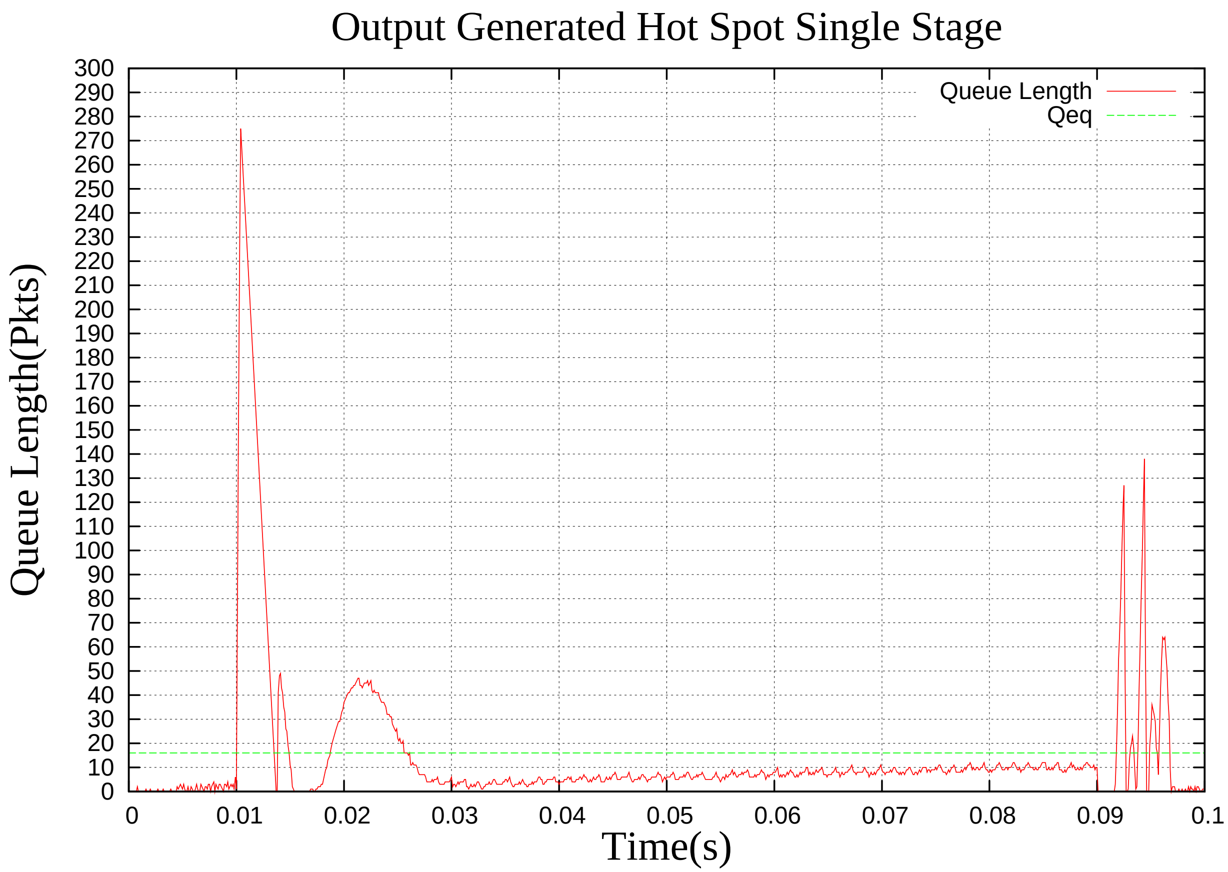
- **Conclusion**: Link throughput is stable.

OGHS-LD: 9 Congested Flows



- **Conclusion**: Perfect fairness

OGHS-LD: Queue Length



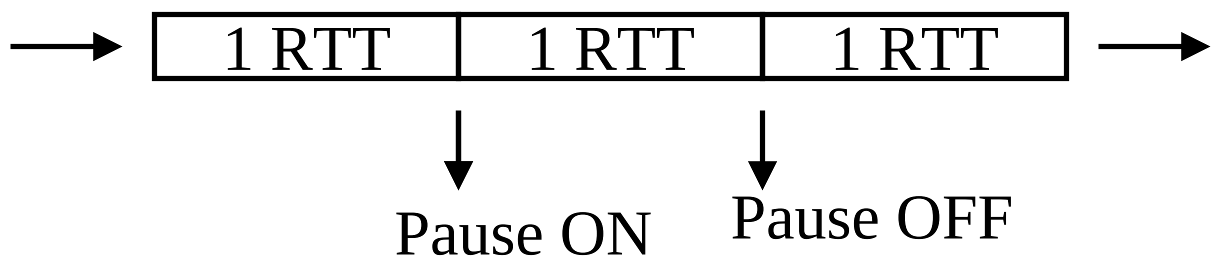
- **Conclusion**: FECN works for long delay without any change in any parameters

OGHS-LD: Other Observations

- Pause On/Off Threshold is 90/8 packets…
- 3 Pause events
- Total pause duration 0.0045 s

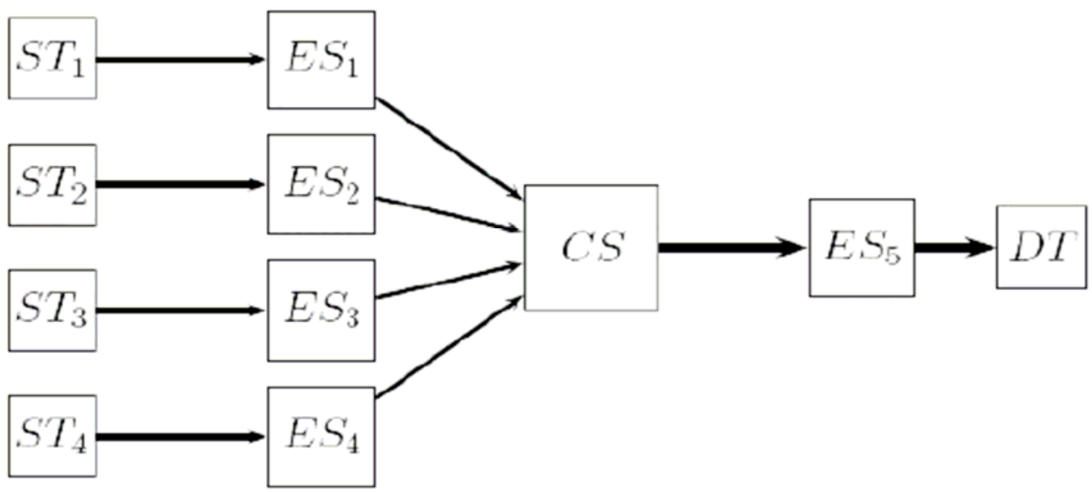
Minimum Buffering Required w Pause

- Need 1 RTT buffer to allow queue to not go to zero after a Pause OFF
  $\Rightarrow$ Pause OFF threshold = 1 RTT
  $\Rightarrow$ Pause ON Threshold = 2 x Pause OFF = 2 RTT
- Need 1 RTT extra buffer to not drop any packets after a Pause ON
- Total Buffer = 3 RTT

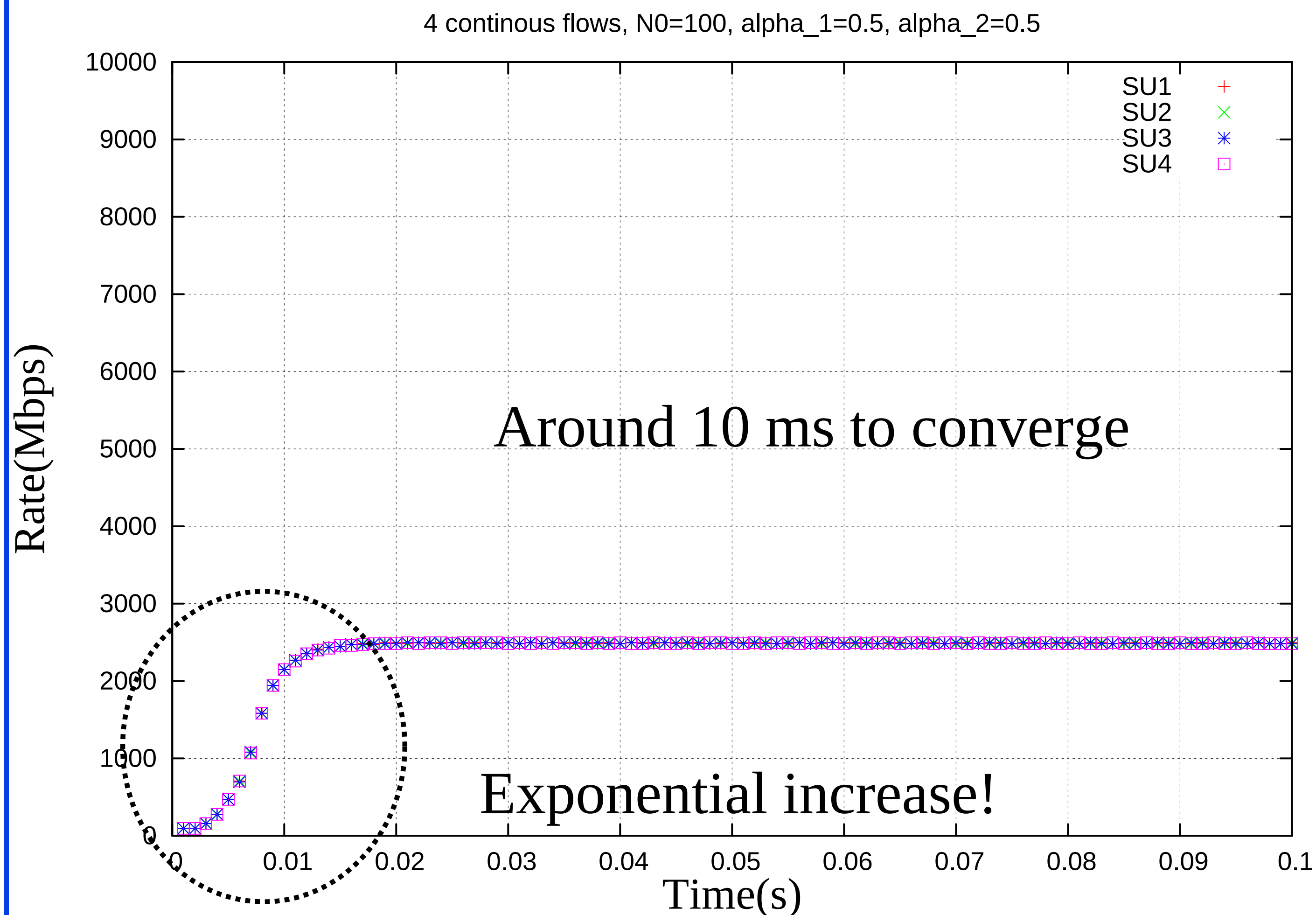


Sensitivity Analysis

- All configurations analyzed so far used same parameter values, except for N0.
- How N0 affects the scheme?
- Continuous traffic, N0=100, 80, 40, 20

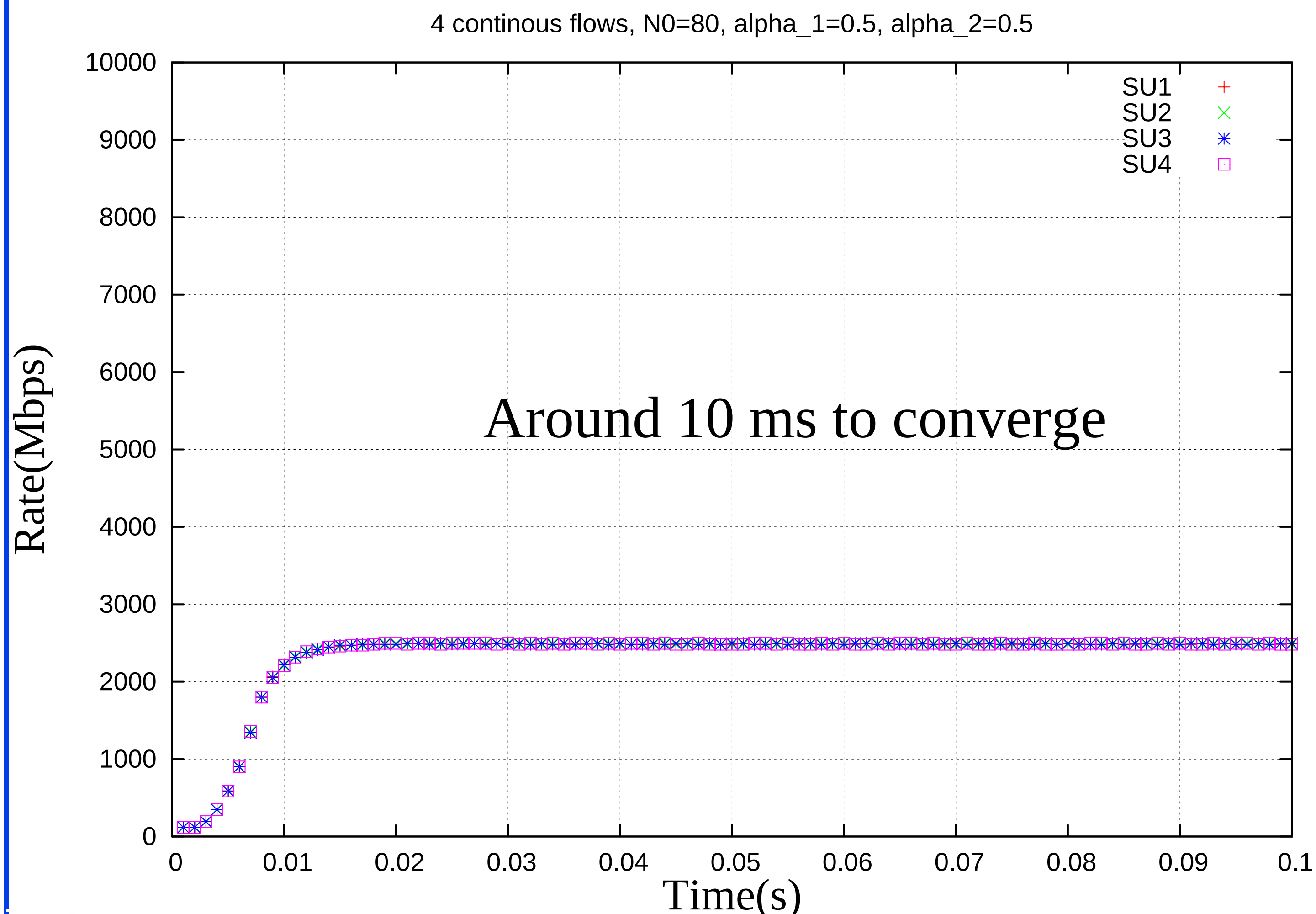


N0=100

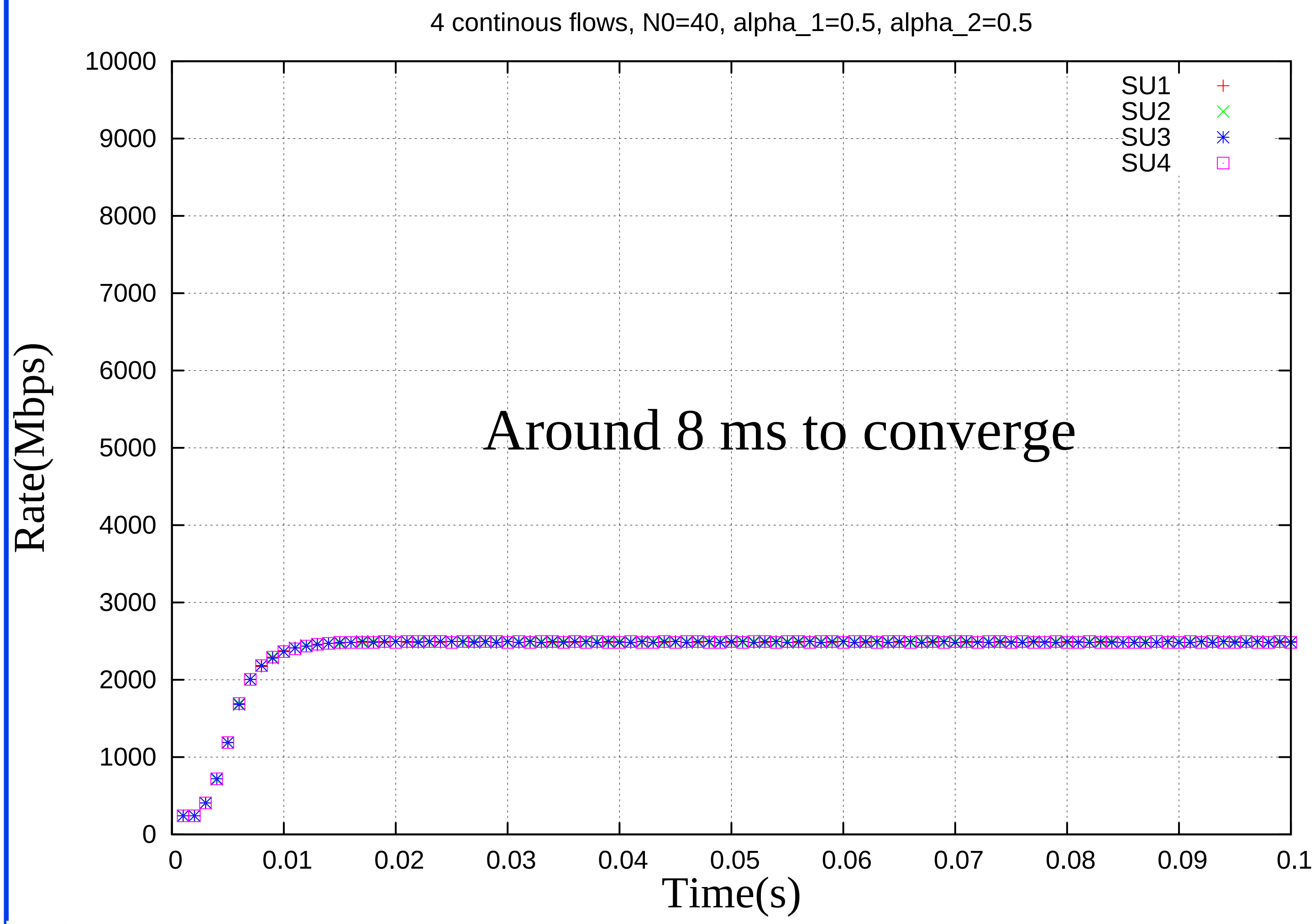


4 continous flows, N0=100, alpha_1=0.5, alpha_2=0.5
Rate(Mbps)
Time(s)
SU1
SU2
SU3
SU4
10000
9000
8000
7000
6000
5000
4000
3000
2000
1000
0
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
0.09
0.1
Around 10 ms to converge
Exponential increase!

N0=80

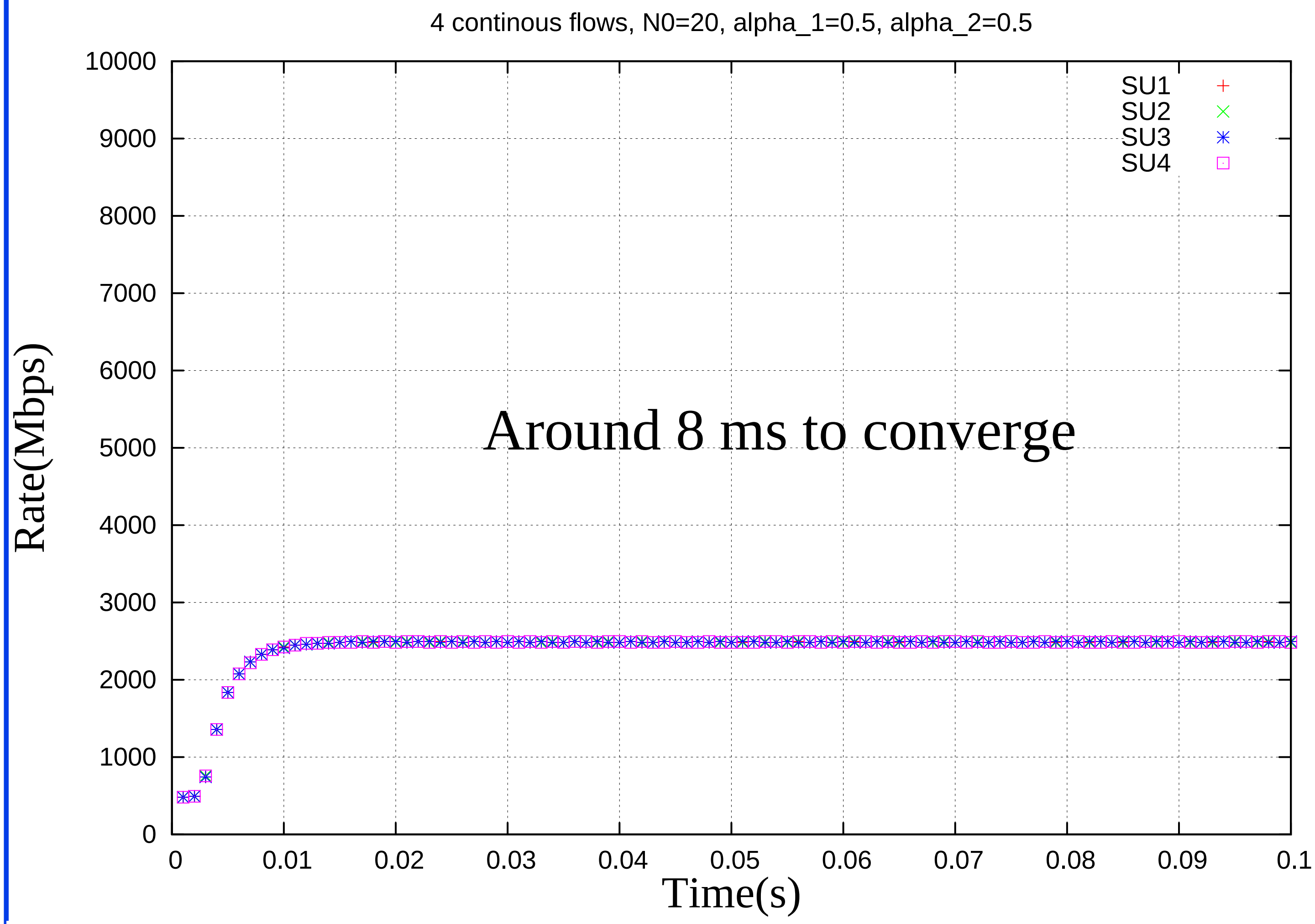

4 continous flows, N0=80, alpha_1=0.5, alpha_2=0.5
Rate(Mbps)
Time(s)
SU1
SU2
SU3
SU4
Around 10 ms to converge
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
0
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
0.09
0.1

N0=40



4 continous flows, N0=40, alpha_1=0.5, alpha_2=0.5
Rate(Mbps)
Time(s)
SU1
SU2
SU3
SU4
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
0
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
0.09
0.1
Around 8 ms to converge

N0=20


4 continous flows, N0=20, alpha_1=0.5, alpha_2=0.5
Rate(Mbps)
Time(s)
SU1
SU2
SU3
SU4
Around 8 ms to converge
10000
9000
8000
7000
6000
5000
4000
3000
2000
1000
0
0
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
0.09
0.1

Sensitivity to N0

- The limited rate increase results in a logarithmic rise

Convergence time to go from N0 to N = $\log_{\lambda}\left(\frac{N0}{N}\right)$

Here λ is the multiplier used in limited increase

So now N0 does not have a significant effect. It can be set to a large value.

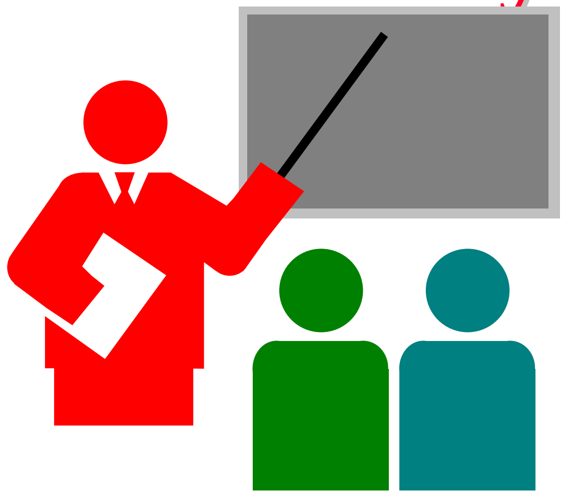
Overhead of FECN

- Given the configuration of the network, FECN has almost deterministic overhead
- Each flow generates one tag every T interval.
- For N flows in a simulation of duration t:
  - t*N/T FECN tags are added to forward data packets
  - t*N/T FECN control messages returned by the destinations
- Alternative designs where T is dynamically varied depending upon the stability, load, or rate were tried successfully but deemed unnecessary. A simple two T strategy consists of using a larger T if the system is operating near optimal region.
- It is also possible to use count based rate discovery, where every nth packet is tagged. This works but convergence to fairness takes slightly longer.

Advantages of FECN

- Flexibility:
  - Switches can base rates on resources other than one queue, e.g., sum of input and output queues, utilization of shared buffers, # of channels available on a wireless link, etc.
  - Switches can give different rate to a flow based on traffic type, class of service, types of sources, VLANs
- Works perfectly on variable link speeds, e.g., wireless links
- Vendor differentiation

Summary



1. Convergence of rates is very fast
2. Convergence time is a small multiple of measurement interval T
3. Convergence to fairness is built in. All active sources get the same rate.
4. **Bursty traffic** can be supported and can get fair and efficient allocation due to fast convergence

Summary (Cont)

5. RD tags in the packets are simple – just rates, RLQ ID, and direction.
6. Source algorithm is quite simple
7. Switch enhancements minimize queue buildup and avoid the need for PAUSE
8. No internal parameters or details of the switch are shared outside with the sources ⇒ Switch algorithms and parameters can be easily changed
9. Very few parameters: T
10. Parameters are easy to set.
11. Scheme not very sensitive to parameters
12. Potential for vendor differentiation for switch algorithms.

References

- Bobby Vandalore, Raj Jain, Rohit Goyal, Sonia Fahmy, "Dynamic Queue Control Functions for ATM ABR Switch Schemes: Design and Analysis," Computer Networks, August 1999, Vol. 31, Issue 18, pp. 1935-1949. http://www.cse.wustl.edu/~jain/papers/cnis_qctrl.htm