Forward Explicit Congestion Notification (FECN) for Datacenter Ethernet Networks

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

IEEE 802.1au Congestion Notification Group Interim Meeting,
Monterrey, CA, January 24-26, 2007

These slides are also available on-line at
http://www.cse.wustl.edu/~jain/ieee/fecn701.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

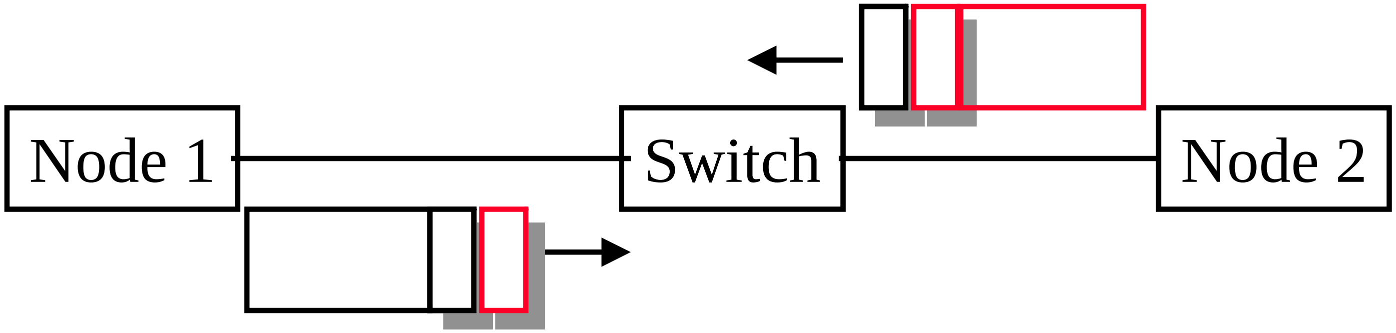
Datacenter Networks

- Bounded delay-bandwidth product
  - High-speed: 10 Gbps
  - Short round-trip delays
- Storage Traffic $\Rightarrow$ short access times $\Rightarrow$ Low delay
- Packet loss $\Rightarrow$ Long timeouts $\Rightarrow$ Not desirable

Top 10 Requirements for a Good Scheme

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

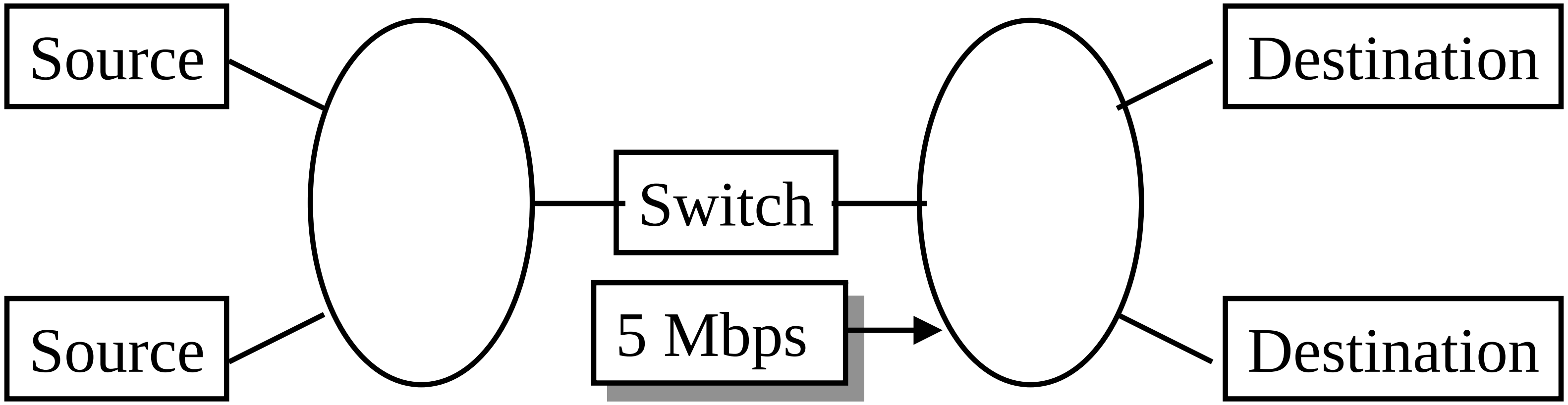


- Every n^{th} packet has two RLT tags (forward RLT tag and reverse RLT tag).
- The tags contain only rate in bps as a 32 bit integer. (Rate coding can be optimized) and Rate limiting Q ID
- The sender initializes the forward RLT tag with rate=-1 ($\Rightarrow\infty$)
- The switches adjust the rate down if necessary
- The receiver copies the forward RLT 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 an 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 RLT tags
- All sources passing through the switch get the same feedback.
- The sources send at the rate received.

A Simple Switch Algorithm

0. Start with an Advertised Rate of r
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. Every RLT tag forwarded set rate to min{rate in tag, r/z}
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***

Switch Algorithm with Q-Control

1. **Initialization**: $$r_0 = \frac{C}{N_0}$$

   Here C is the link capacity in bits/s. r_0 can be almost any value. It has little effect on convergence time.

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

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

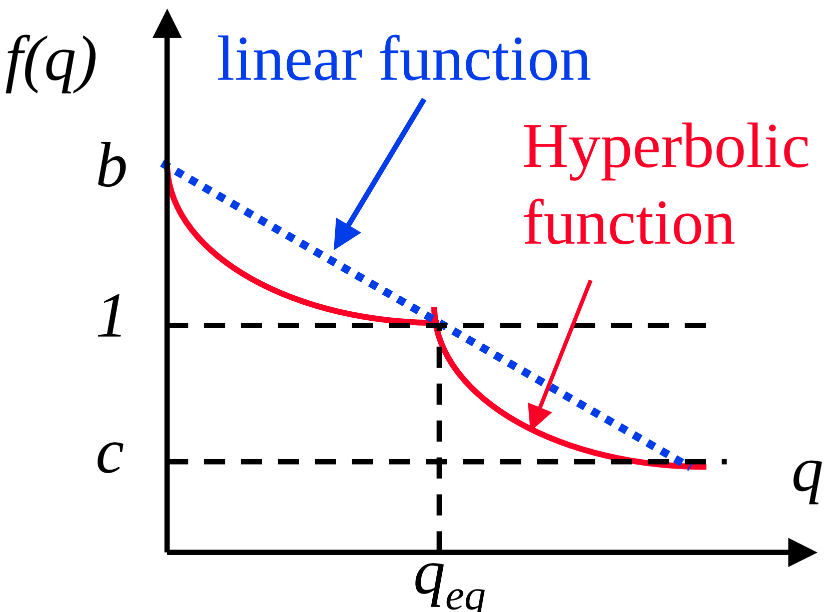
3. **Bandwidth Allocation**: $$r_{i+1} = \frac{r_i}{\rho_i}$$

Queueing 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: *f(q)*

- **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

Enhancements

1. **Exponentially weighted average in the Switch**:

$$r_i = \alpha \frac{r_{i-1}}{\rho_i} + (1-\alpha) r_{i-2}$$
$$\alpha \in (0,1)$$

Remembers recent history. In all simulations $\alpha = 0.5$

2. **Limited Rate Increases in the Switch**: (Tentative)

$$\text{If } r_i - r_{i-1} > \Delta r, \quad r_i = r_{i-1} + \Delta r$$

In all simulations $\Delta r = r_0$

3. **Time-based sampling at the source**: Packet tagged if time since the last time tag was sent is more than τ

In all simulations $\tau = T$

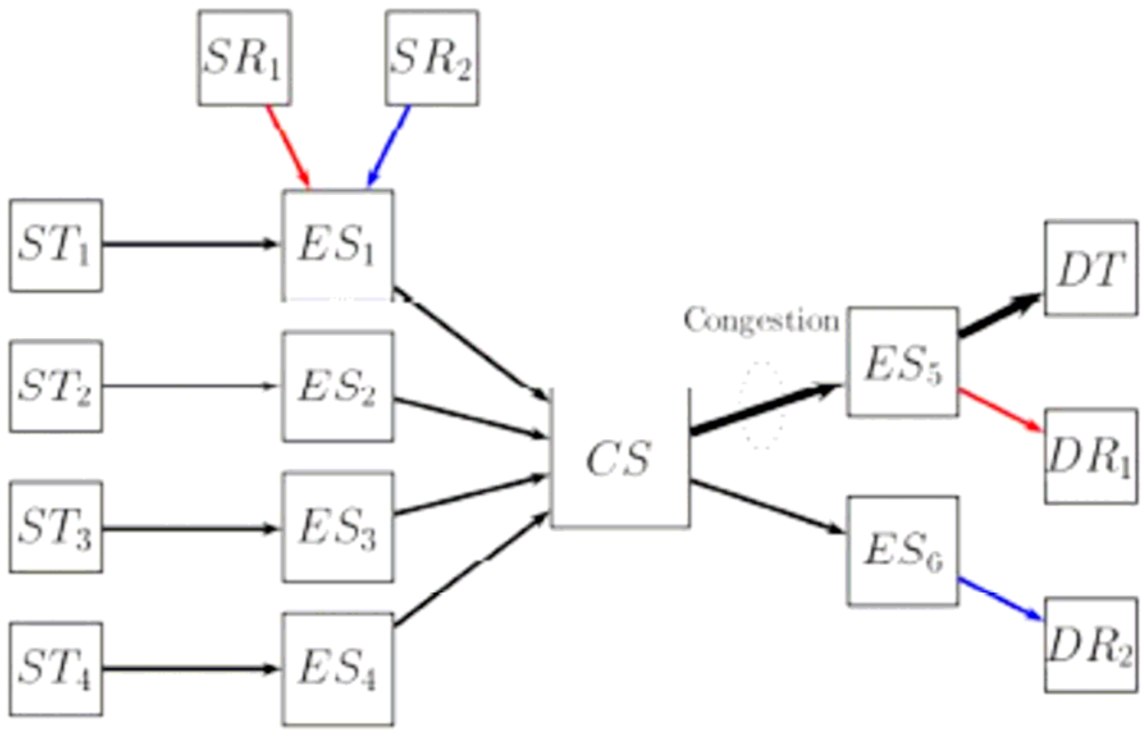
General Simulation Parameters

- 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

FECN with TCP flows



- 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

- T = Tau = 1 ms
- Workload
  - ST1-ST4: 10 parallel TCP connections transferring 1 MB each continuously
  - Reference flows: 1 TCP connection transferring 10kB each with average idle time 16 us for SR1 and 1 us for SR2

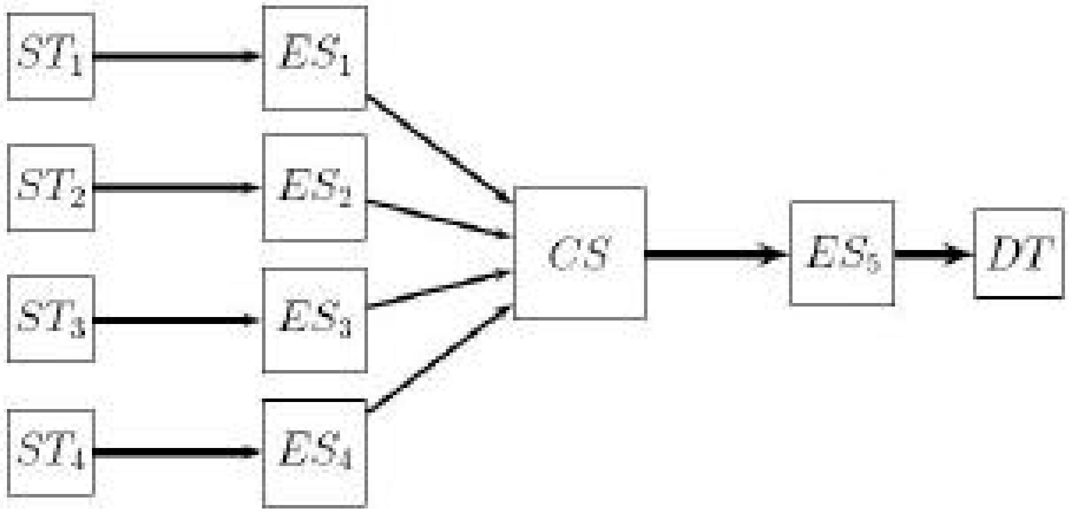
Simulation Results

	Reference Flow 1			Reference Flow 2		
CM	Throughput (Tps)	Throughput (Gps)	Latency (us)	Throughput (Tps)	Throughput (Gbps)	Latency (us)
None	556	0.06	1780.78	16634	1.44	59.11
FECN	6970	0.604	127.63	16630	1.44	59.16

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

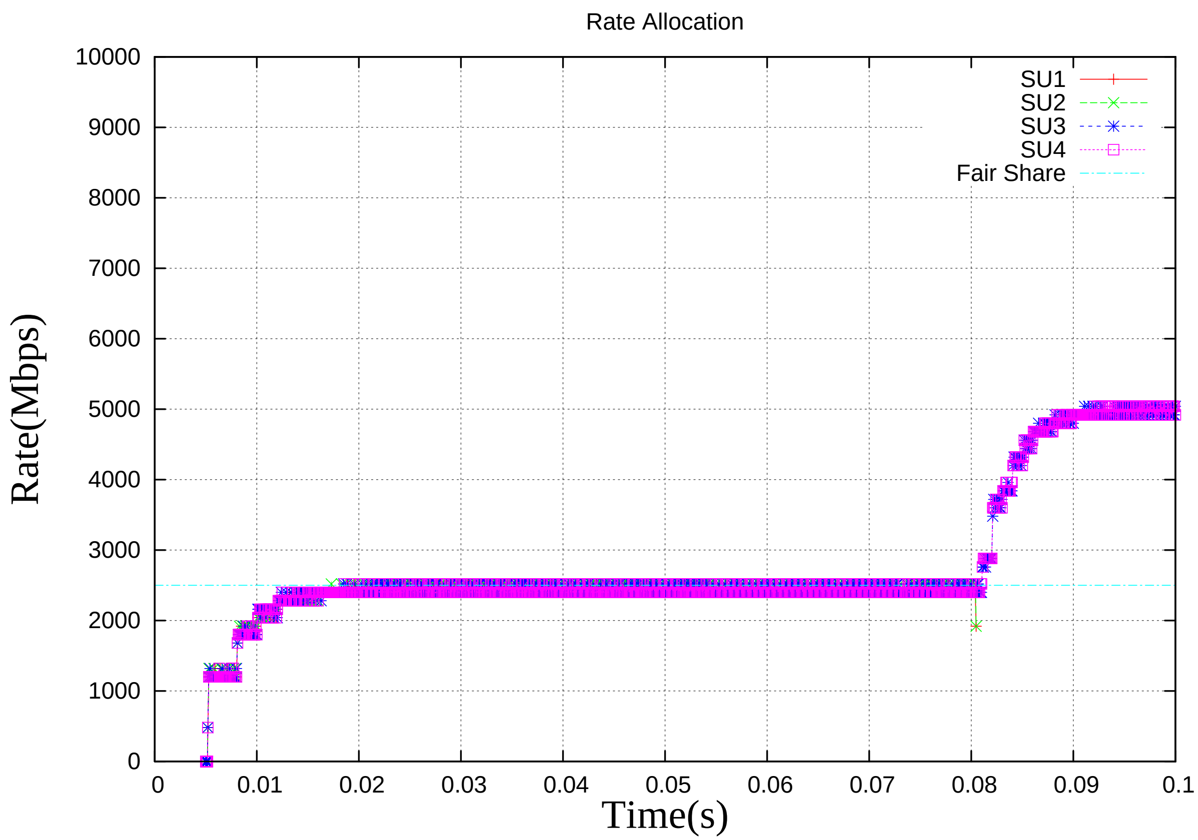
Conclusions: FECN can protect fragile TCP flows and improve its goodput and fairness significantly

Symmetric Topology: Configuration



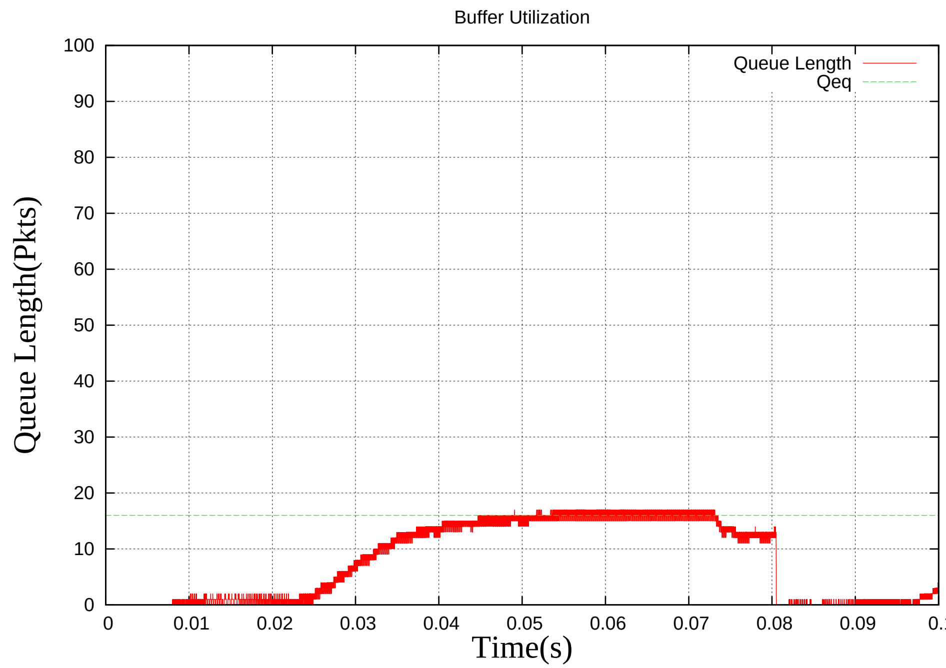
- UDP Bernoulli Traffic with average *5* Gbps rate
- Measurement Interval T is *1* ms
- Simulation Time is *100* ms, all sources starts at *5* ms
- At *80* ms, 2 sources stop

Symmetric Topology: Source Rate (T=1 ms)



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

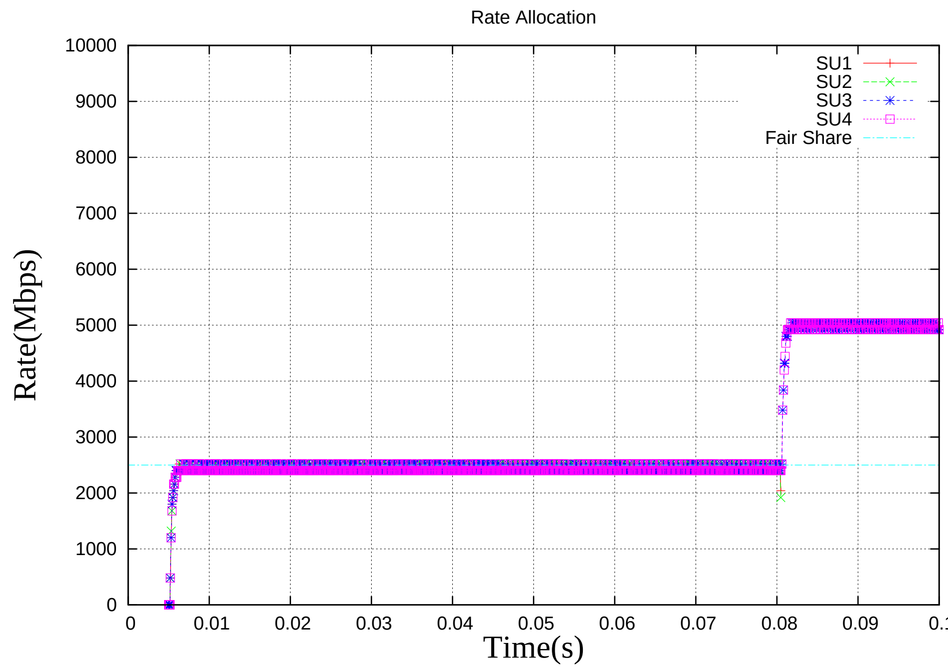
Symmetric Topology: Queue Length (T=1 ms)



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

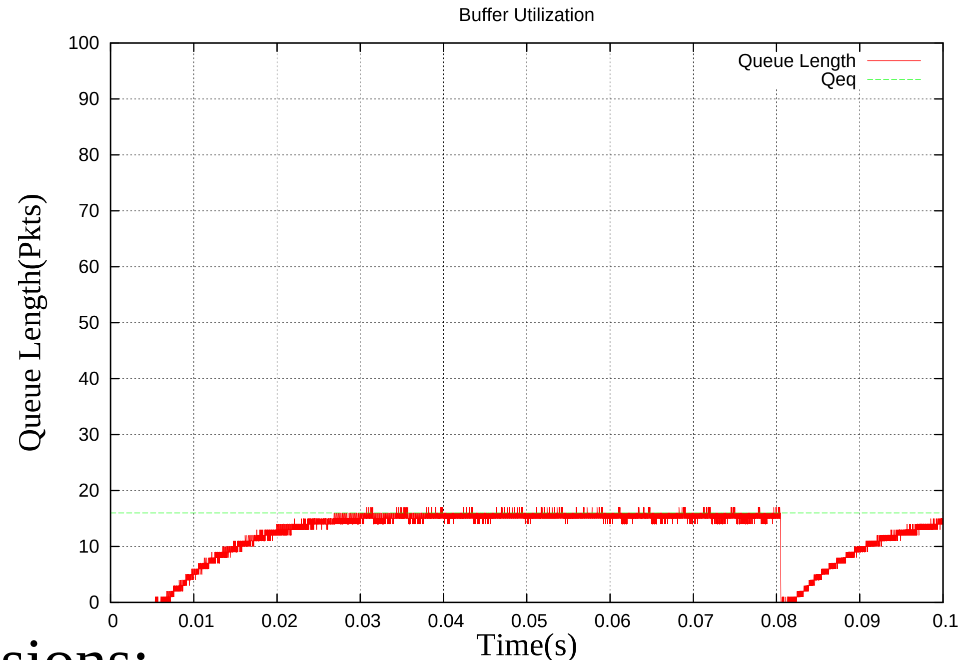
Symmetric Topology: Source Rates (T=0.1 ms)

- T=0.1ms



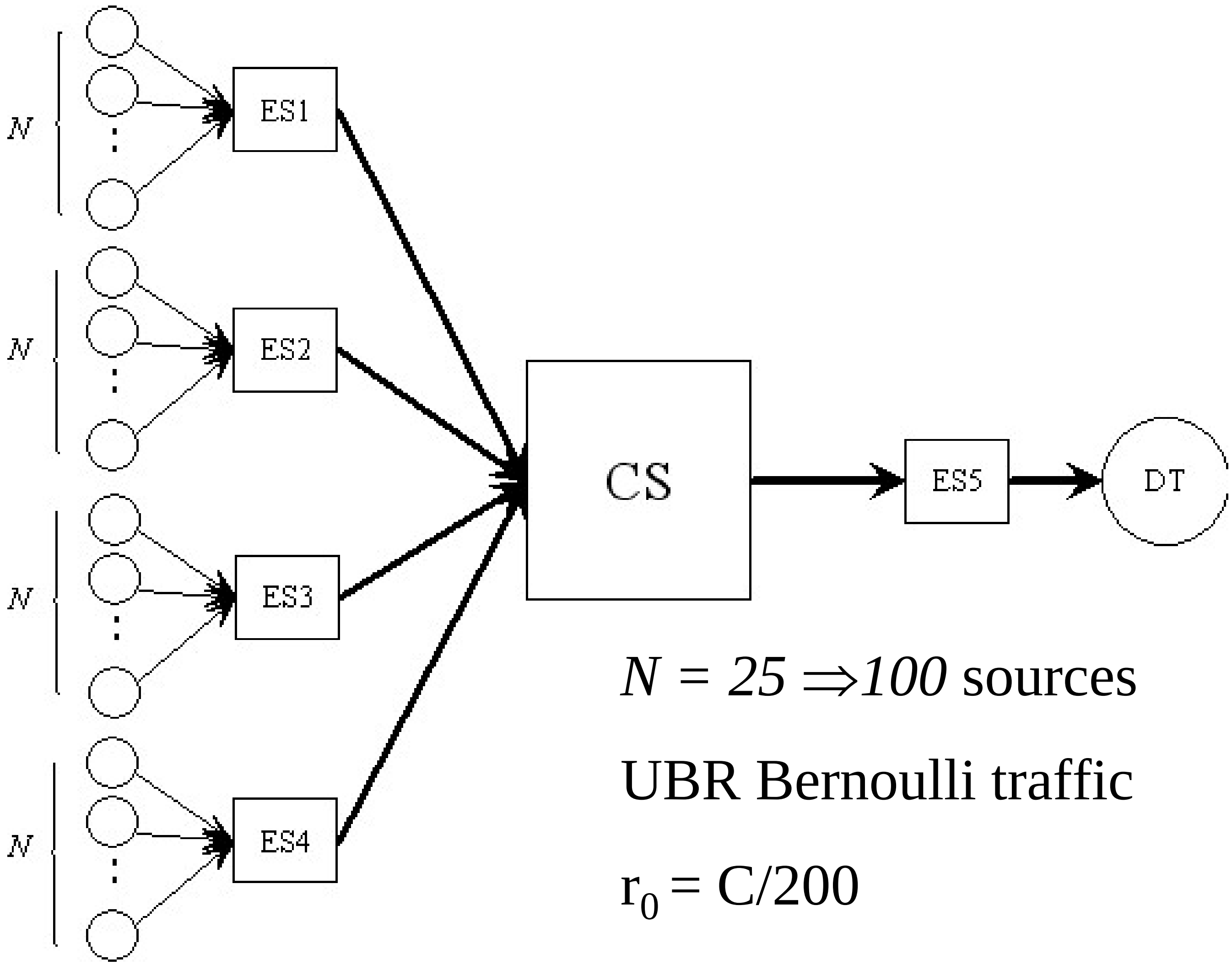
- **Conclusions**:
  - Convergence time is a small multiple of T
  - Smaller T leads to faster convergence.

Symmetric Topology: Queue Length (T=0.1 ms)



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

Large Topology: Configuration

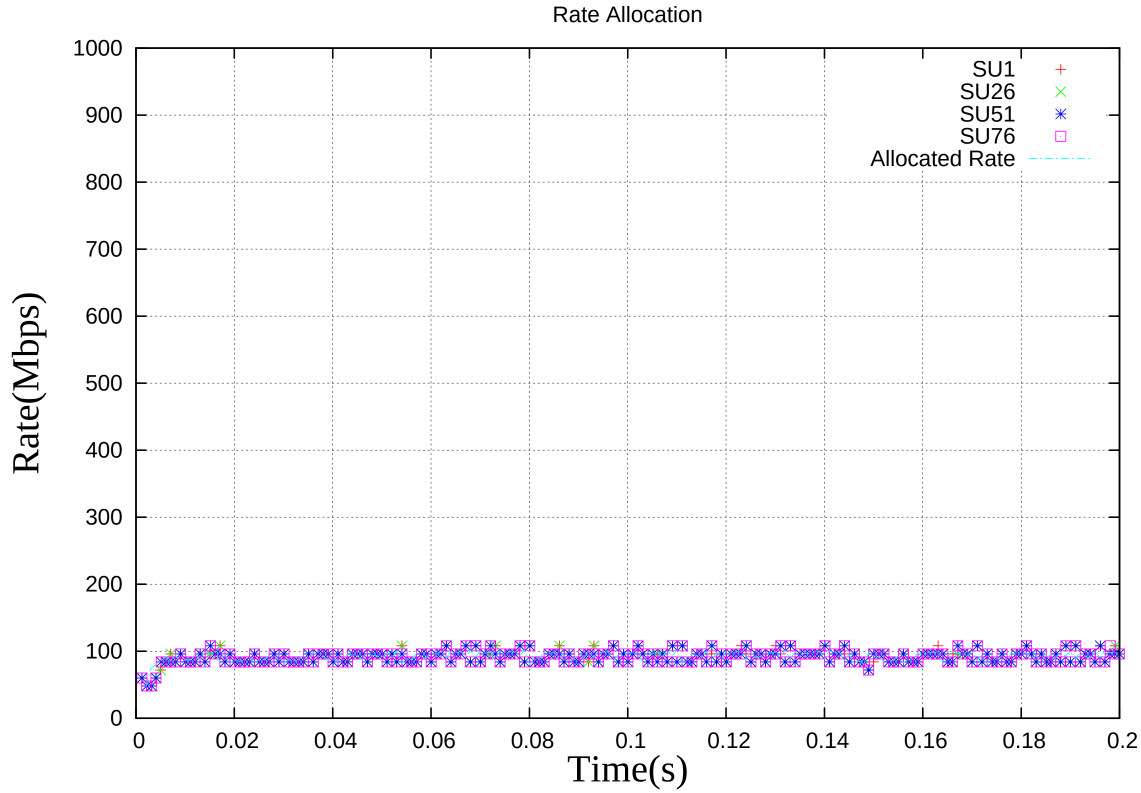


N = *25* ⇒*100* sources

UBR Bernoulli traffic

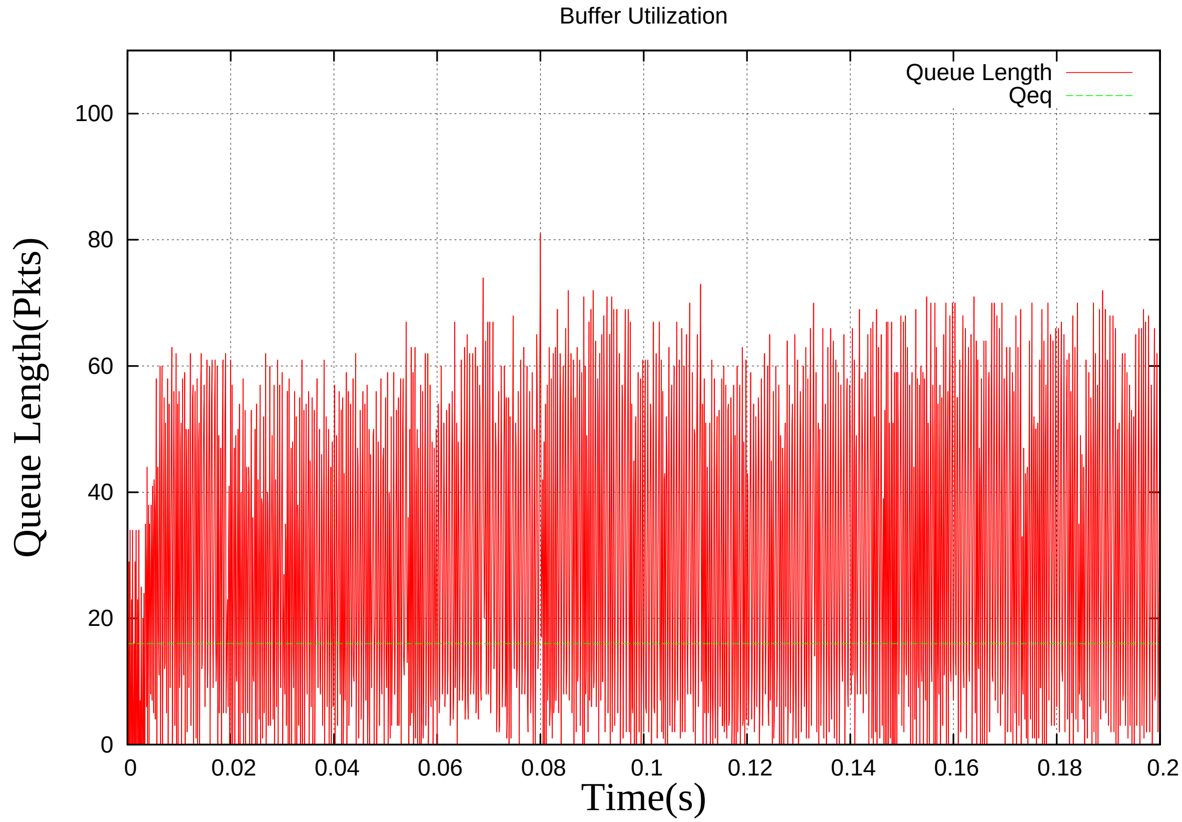
$r_0 = C/200$

Large Topology: Source Rates



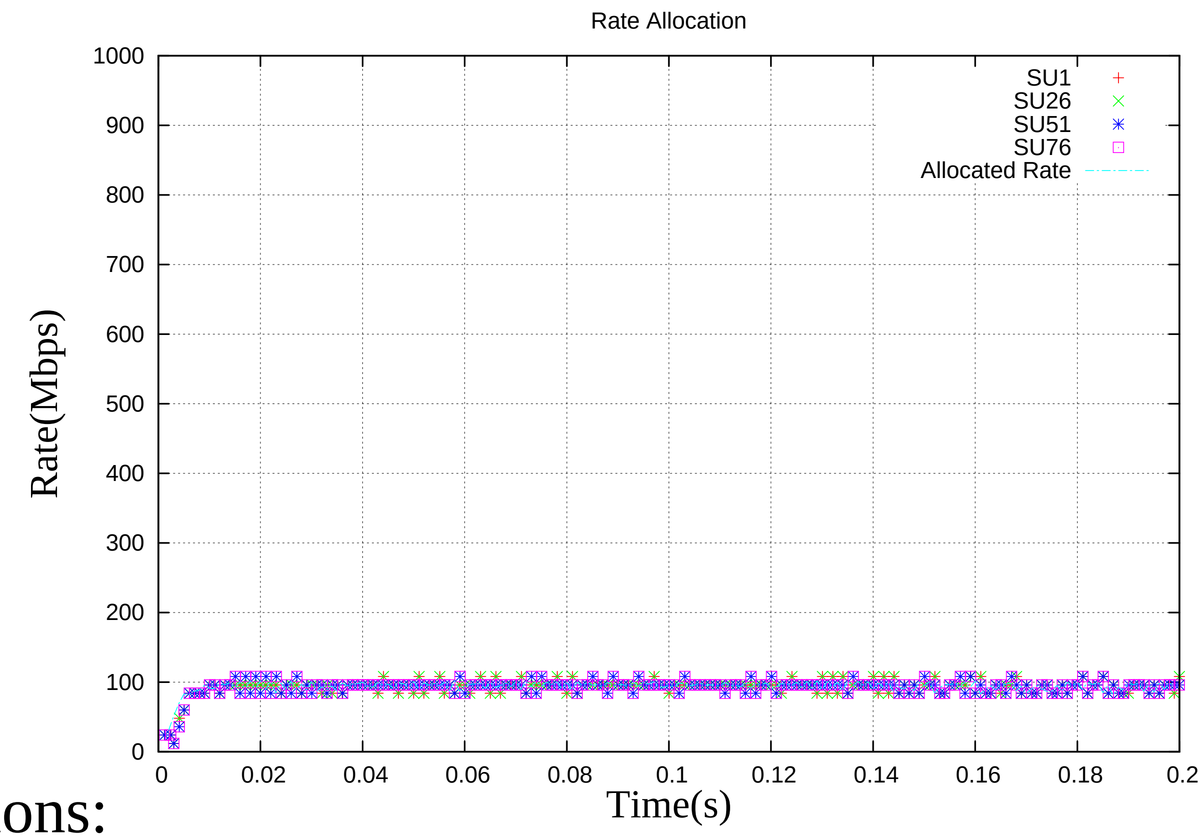
- Conclusions:
  - Perfect Fairness!
  - Fast Convergence: less than 10 ms

Large Topology: Queue Length



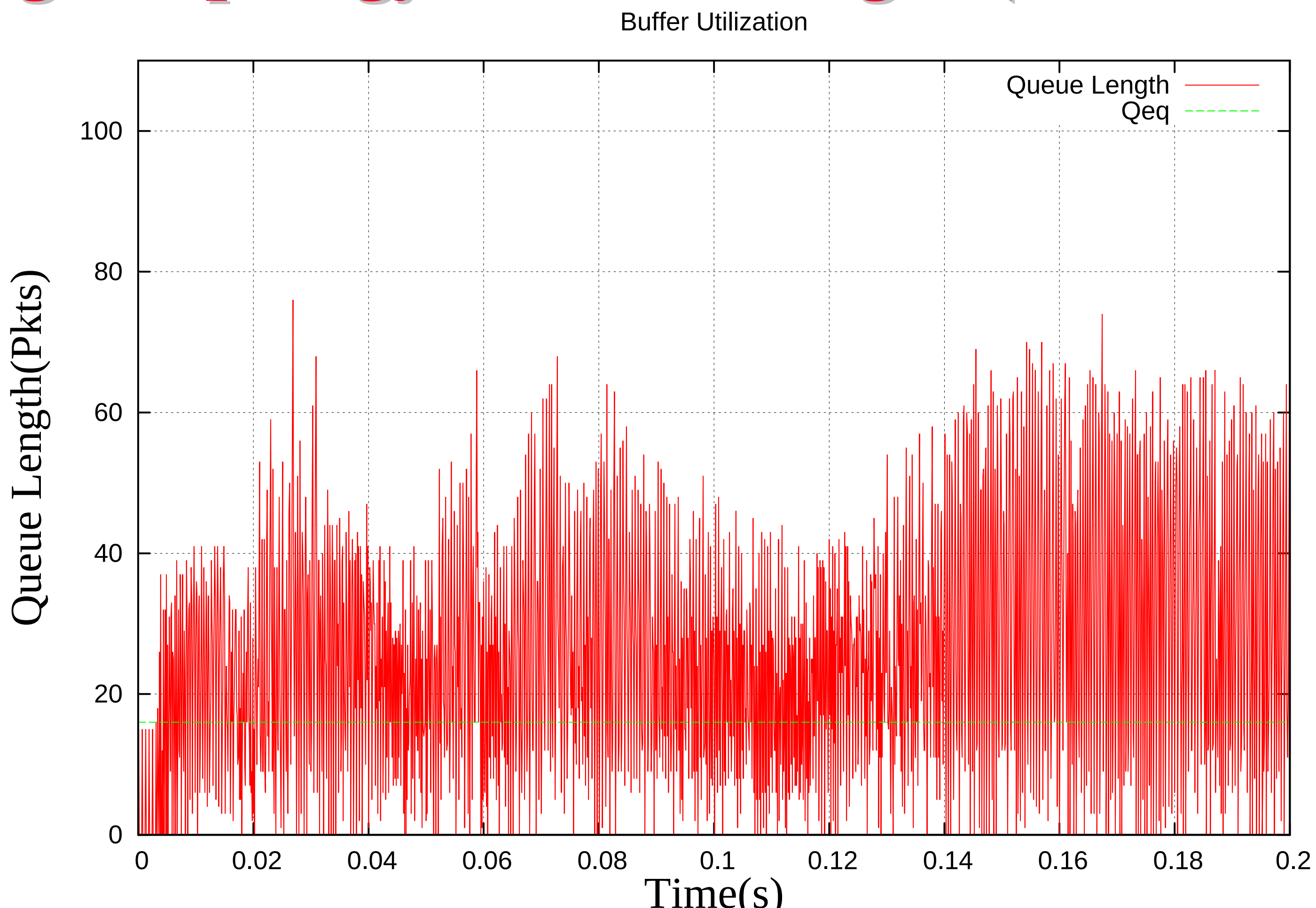
- Conclusions:
  - Queue does not overflow!
  - No PAUSE required or issued

Large Topology: Source Rates (N0=500)



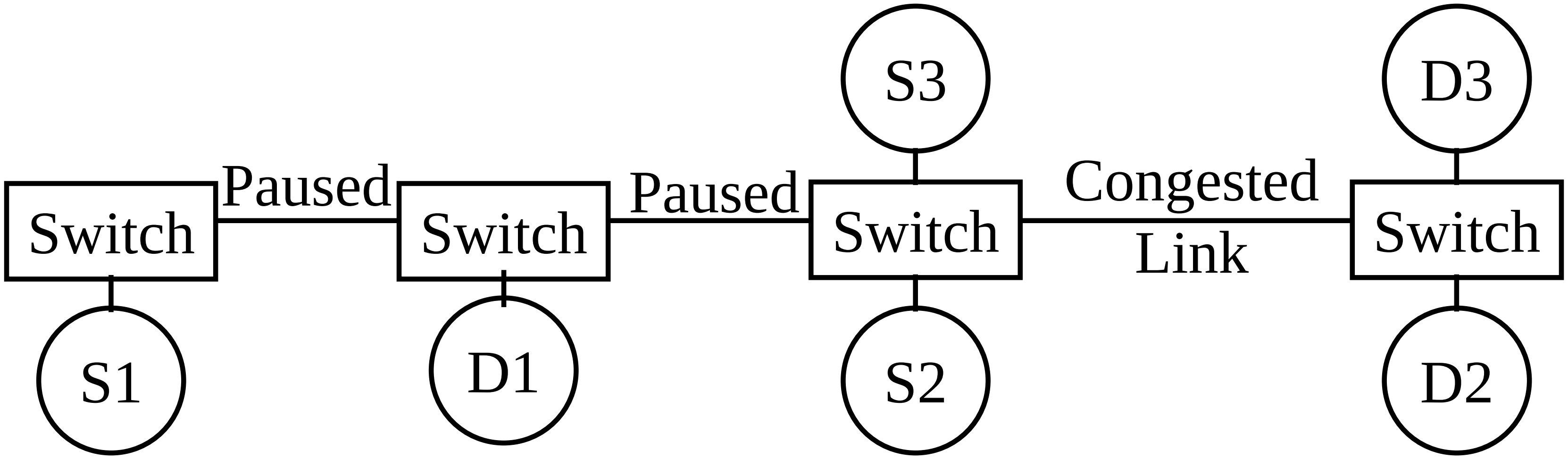
- Conclusions:
  - Perfect Fairness!
  - Fast Convergence: less than 10 ms,
  - A bit slower compared to N0=200

Large Topology: Queue Length (N0 = 500)



- Conclusion: Zero PAUSE issued.

PAUSE



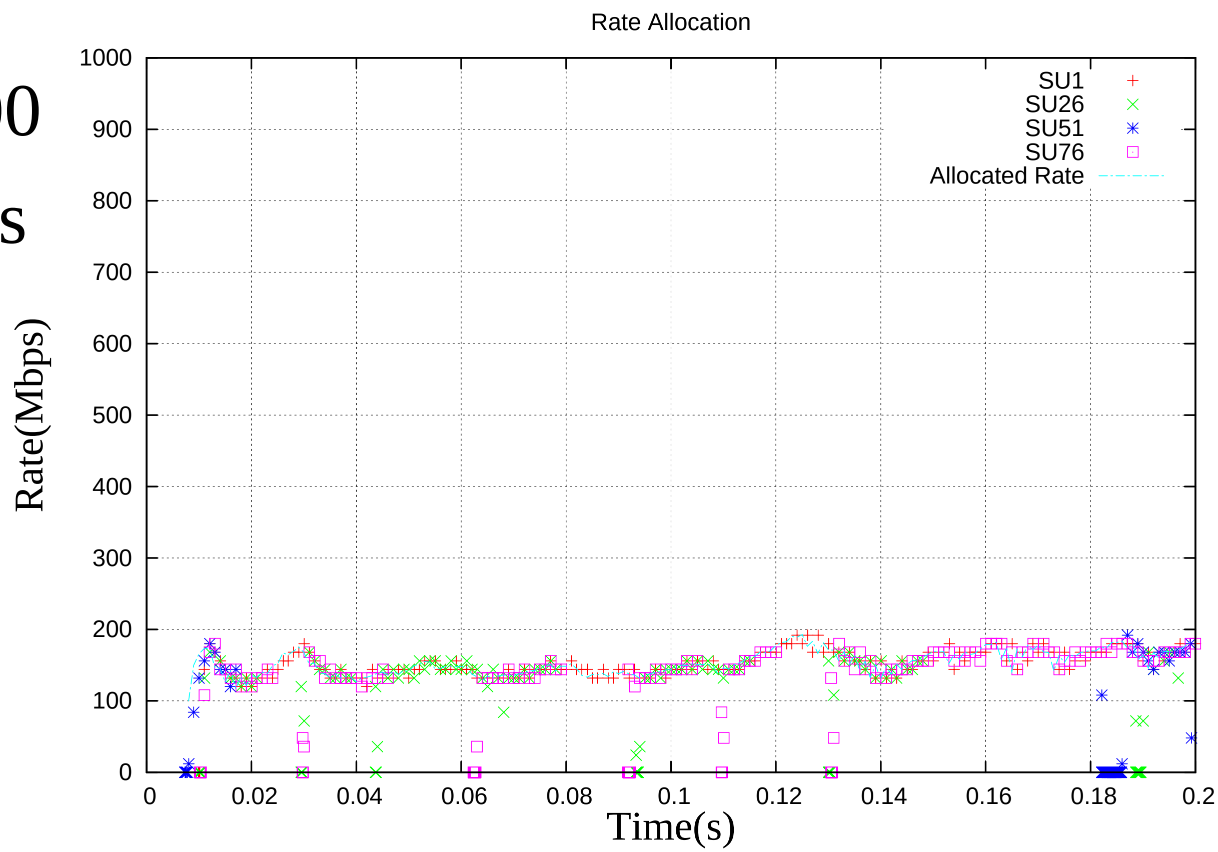
- S1-to-D1 flow is not using congested resources but is stopped by congestion caused by S2-to-D2 and S3-to-D3
- **Conclusion**:
  - Pause unfairly affects non-congestion causing flows
  - Pause should not be used as a primary or frequent mechanism
  - Pause can reduce loss but increase delays in the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

Bursty Traffic: Configuration

- Large Topology
- The sources come on and go off after transmitting a burst.
- The ON/OFF period is Pareto distributed
- Average ON/OFF period is *20* ms

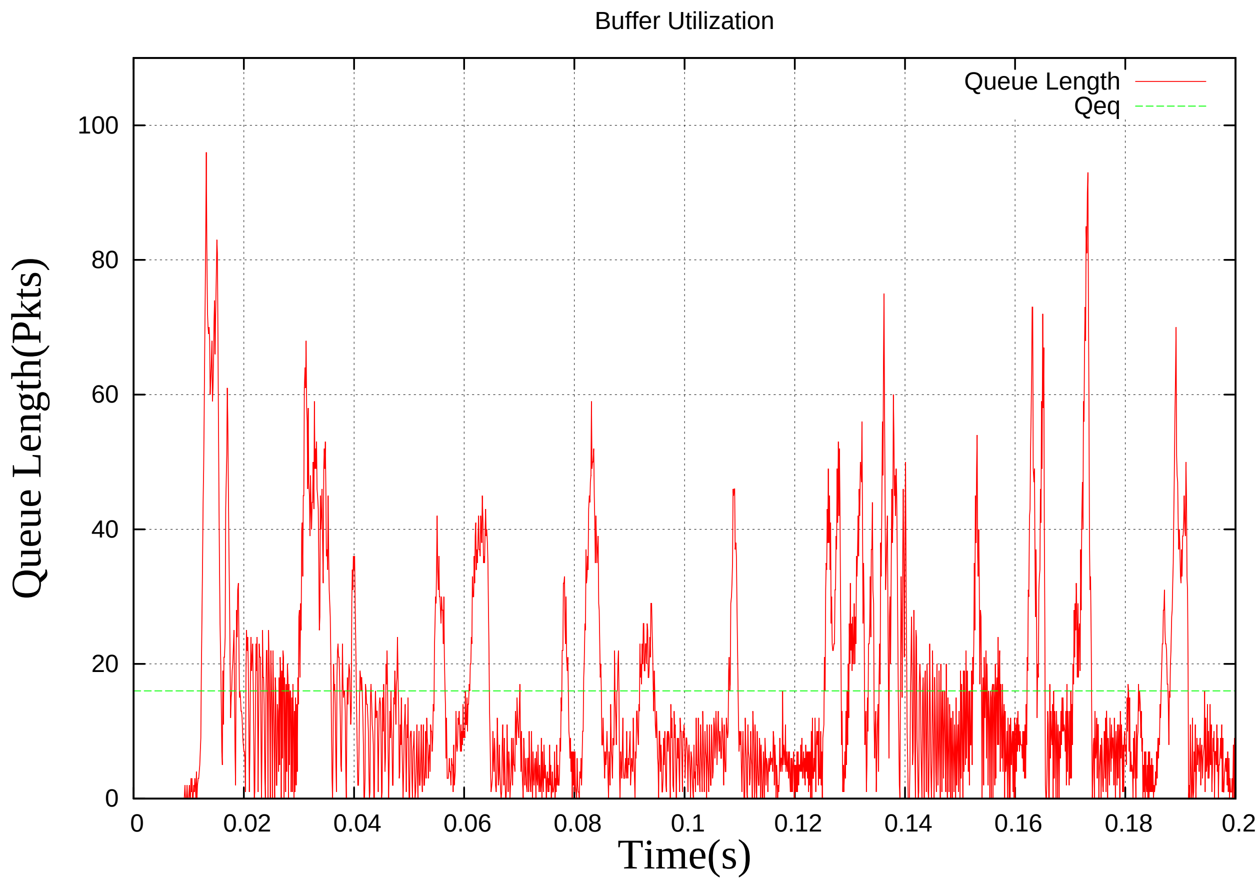
Large Topology Bursty Traffic: Rates

- N0=200
- T=1 ms



- Conclusions: Perfect Fairness!

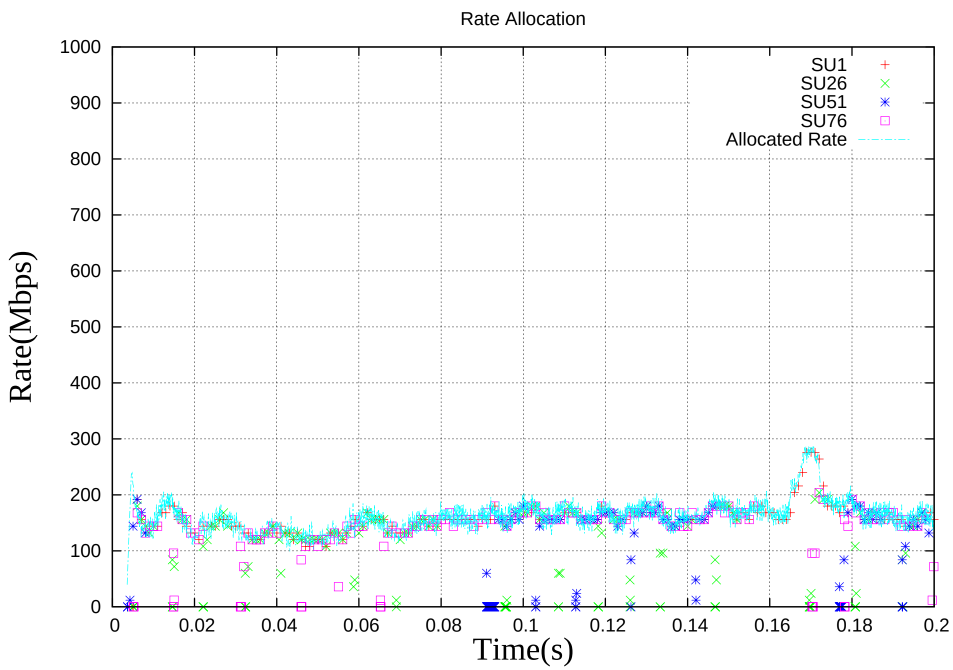
Large Topology - Bursty Traffic: Queue



- Conclusion: No PAUSE issued!

Large Topology – Bursty Traffic: Rates

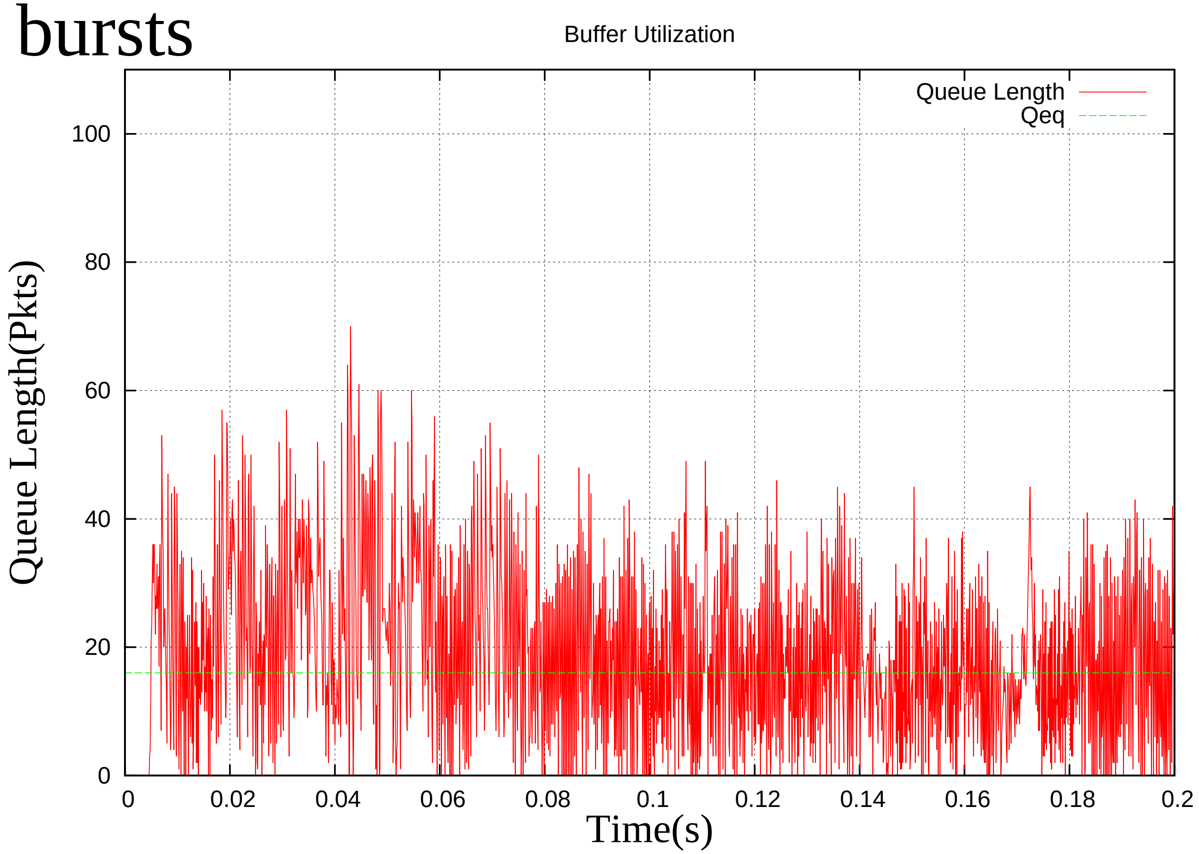
- 10 ms bursts – Pareto distributed burst on/off times
- N0 = 200
- T = 0.1ms



- Conclusion: FECN works efficiently, fairly, and quickly even for 10 ms bursts from 100 sources.

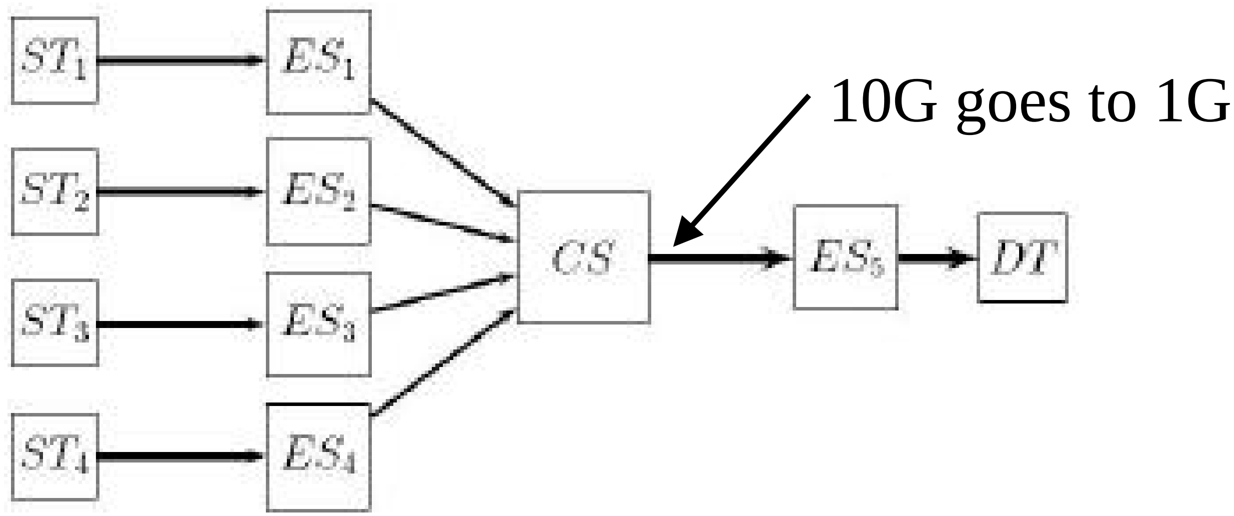
Large Topology – Bursty Traffic: Queues

- 10ms bursts



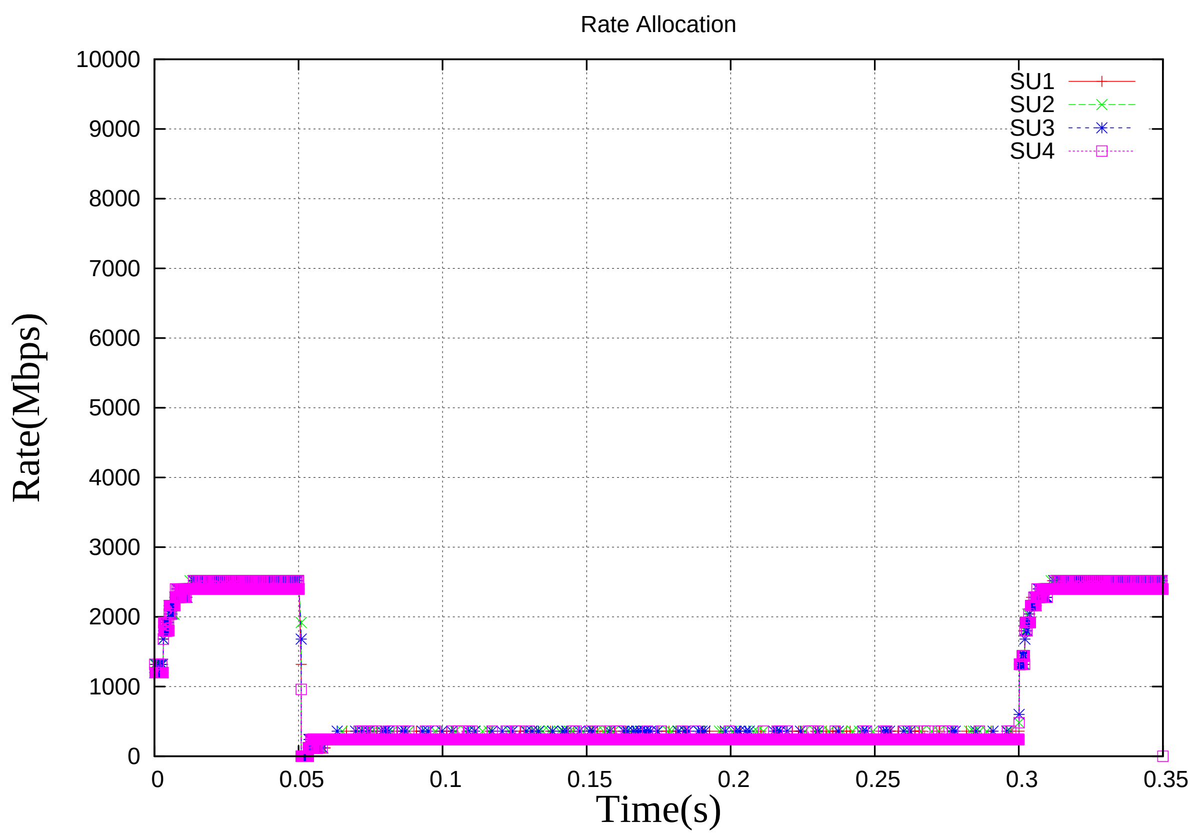
- Conclusions: No PAUSE required.

Output Generated Hotspot Scenario



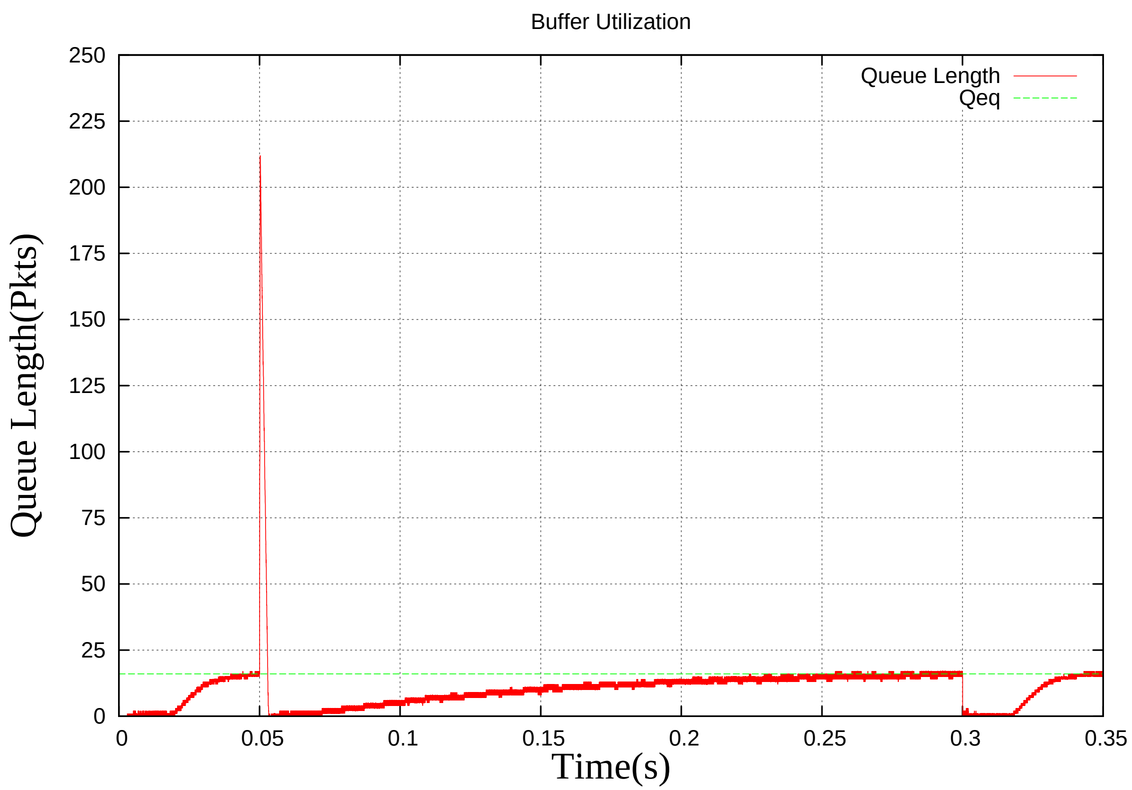
1. Capacity from CS to ES5 goes to *1* G from 0.05ms to 0.30 ms, then come back to *10* Gbps
2. We study per flow behavior instead of per node behavior
3. Symmetric topology configuration is used
4. Capacity C(t) is known from the idle time and bits transmitted.

Hotspot Scenario: Source Rate



Conclusion: FECN converges around 250 Mbps when the capacity of congested link shrinks to 1Gbps

Hotspot Scenario: Queue Length

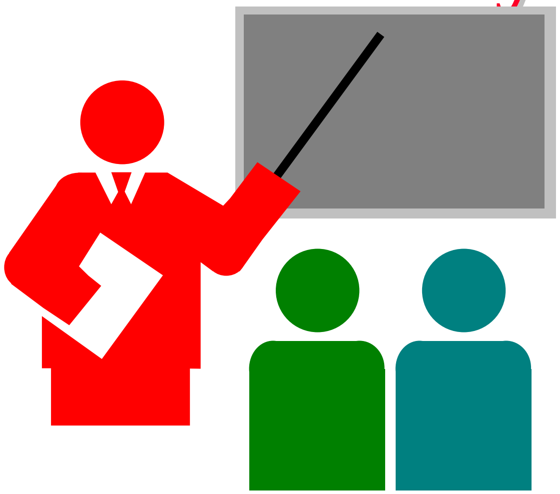


Conclusion: The queue can converges to q_{eq}.
Even the initial peak is manageable.

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

5. RLT tags in the packets are simple – just rates and RLQ ID.
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 $\Rightarrow$ Switch algorithms and parameters can be easily changed
9. Very few parameters: T and N0.
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