CSFQ

Core-Stateless Fair Queueing

Presented

by

Nagaraj Shirali

Choong-Soo Lee

About the Authors

Ion Stoica – CMU

- PhD degree from Carnegie Mellon University
- Assistant Professor at University of California, Berkeley
- Networking with an emphasis on Quality of Service and traffic management in the Internet

Hui Zhang – CMU

- PhD degree from University of California, Berkeley
- Associate Professor at Carnegie Mellon University
- Internet, multimedia systems, resource management, and performance analysis

Scott Shenker – Xerox PARC

- Chair for the Integrated Services (INTSERV) charter

Outline

- **Introduction**
- Background: Definitions and Previous Work
- CSFQ and its Algorithms
- Simulations
- Evaluations of CSFQ
- Conclusions and Future Work

Introduction

- **Main Idea**:
  - Achieve fair bandwidth allocations at the router without the implementation complexity usually associated with it.

- **Goals**:
  - Achieve fair allocation close to Fair Queueing and comparable or better than RED and FRED under most scenarios.
  - Reduce complexity by not having the core node maintain per flow state.
  - Approximate weighted FQ.

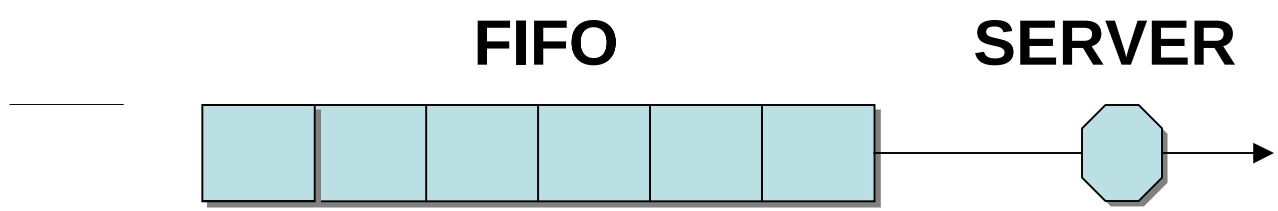
Outline

- Introduction
- **Background: Definitions and Previous Work**
- CSFQ and its Algorithms
- Simulations
- Evaluations of CSFQ
- Conclusions and Future Work

Previous Work

- FIFO queueing with Drop Tail
- Random Early Drop (RED)
- Flow Random Early Drop (FRED)
- Fair Queueing (FQ)

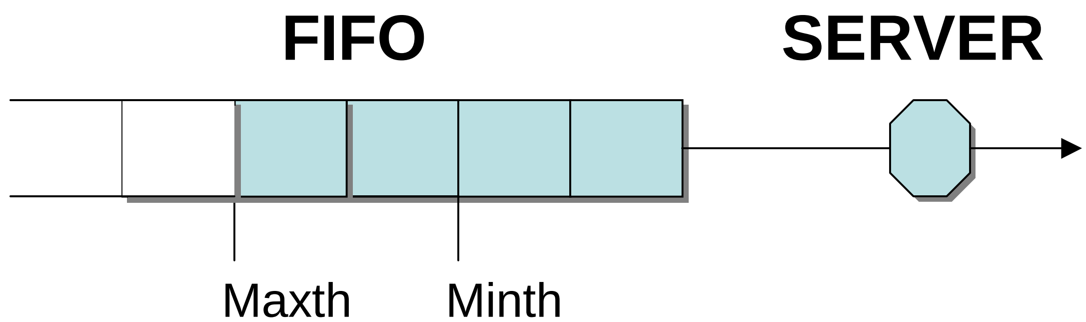
FIFO queueing with Drop Tail



Disadvantages:

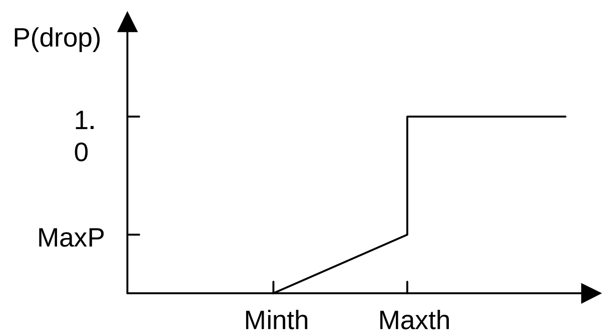
- Pushes congestion control out to end hosts (TCP)
- Introduces *global synchronization* when packets are dropped from several connections

Random Early Drop (RED)

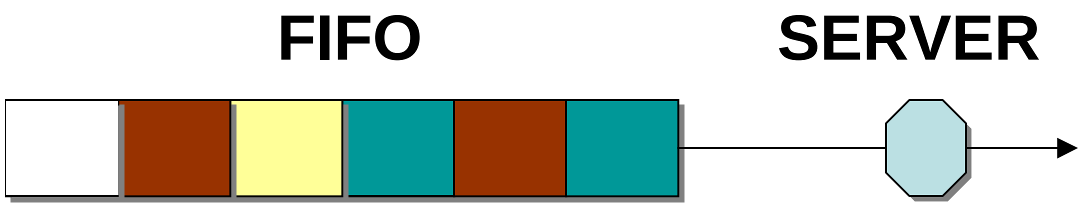


Disadvantage:

- For web traffic, RED provides no clear advantage over tail-drop FIFO for end-user response times



Flow Random Early Drop (FRED)



Disadvantage:

- Complex to implement – maintain state on per-flow basis

Fair Queueing

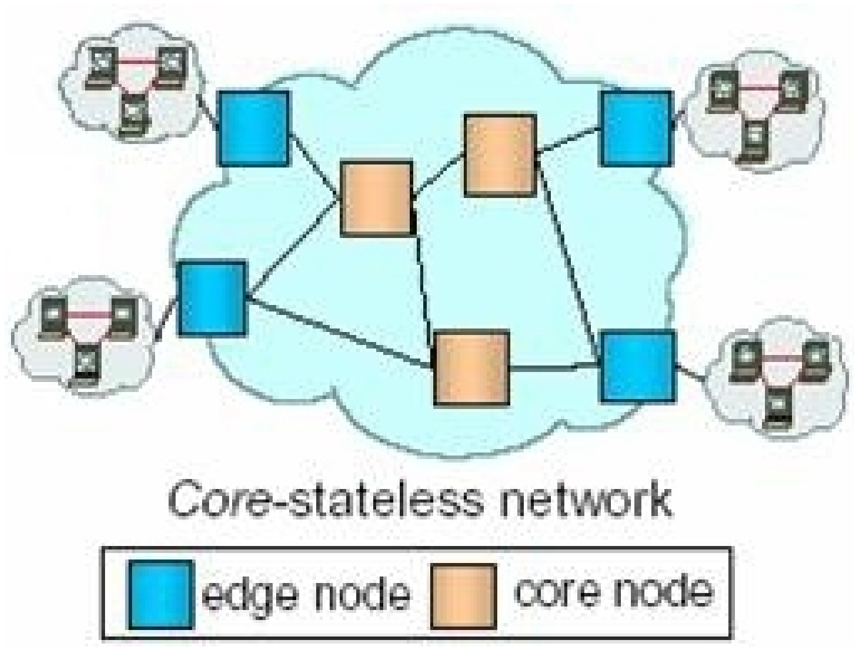
Disadvantage:

- Need to perform packet classification and maintain state and buffers on per-flow basis and perform operations on per-flow basis

Definitions

- ***Island of routers*** – a contiguous portion of the network with well defined interior and edges.
- ***Edge Router*** – computes per-flow rate estimates and *labels* the packets with these estimates.
- ***Core Router*** – uses FIFO queueing and keeps no per-flow state, employs a probabilistic dropping algorithm that uses the packet label and its own measurement of aggregate traffic.
- ***Stateless*** – absence of per-flow state at the core routers.

Island of Routers



Source: CSFQ, Stoica, Berkeley

Outline

- Introduction
- Background: Definitions and Previous Work
- **CSFQ and its Algorithms**
- Simulations
- Evaluations of CSFQ
- Conclusions and Future Work

CSFQ and its Algorithms

Assumptions:

- Fair Allocation methods like FQ are necessary for congestion control.
- The complexity involved is a major hindrance to their adoption.

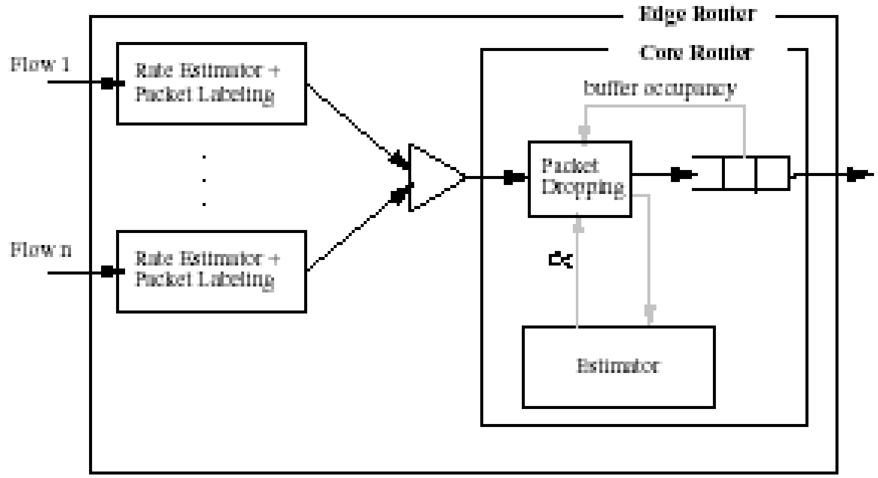
CSFQ

- In an island of routers, edge routers compute per-flow rate estimates and label the packets with these estimates.
- Core routers use FIFO queueing and keep no per-flow state, they employ a probabilistic dropping algorithm based on packet labels and own aggregate traffic estimates.

CSFQ

- Bandwidth allocations using this method are approximately fair.
- Core routers keep no per-flow state and avoid using complicated packet scheduling and buffering algorithms, hence are easier to adopt.

CSFQ



- Assume that flow *i* has arrival rate $r_i(t)$ and the fair rate is $a(t)$.
- If $r_i(t) < a(t)$, all of its traffic is forwarded.
- If $r_i(t) > a(t)$, then a fraction $(r_i(t) - a(t))/ r_i(t)$ will be dropped; each packet of the flow is dropped with probability $(1-a(t)/r_i(t))$. Thus the output rate of any flow *i* will be $max(r_i(t), a(t))$.

CSFQ

- The problem now becomes how to calculate the flow rate $r_i(t)$ values and the fair rate $a(t)$, without keeping per flow state in the core routers.
- Flow rates $r_i(t)$, are calculated at edge routers which keep per flow state and then insert the rate value inside the packet header of packets belonging to that flow.

CSFQ

- To estimate the fair rate *a(t),* an iterative procedure is used: core routers estimate aggregate arrival rate A and the aggregate rate of accepted traffic F (arrival rate – dropped packets).

- Based on these, the *fair rate a* is computed periodically as:

  - if there is no congestion (A<=C where C is the link's capacity), then a is set to the maximum $r_i(t)$

  - if the links are congested, then $a_{new} = a_{old} * C/F$

CSFQ - Example

Assume we have two flows f_1 and $f_{2,}$ with rates $r_1 = 20$ and $r_2 = 30$ and the link's capacity is $C = 30$. Initially let's say that only r_1 is active and the link is not congested, so $a_1 = 20$. Then r_2 becomes active. Since no packets were dropped, $F = 50.$

Since $A = 50>C$, $a_2 = a_1 * C/F = 20 * 30/50 = 12$
Therefore, for f_1 $(1\text{-}12/20 = 40\%)$ of its packets are dropped while for f_2 $(1\text{-}12/30 = 60\%)$ of its packets are dropped and $F = 12+12 = 24$
Since A>C, $a_3 = a_2 * C/F = 12 * 30/24 = 15$

Now F = 30, and $a_4 = a_3 * C/F = 15 * 30/30 = 15.$ Therefore, *a* has converged to the right *fair rate*.

Source: Network Reading Group, Stoica

CSFQ

Estimation of flow arrival rates:

$$R_{new} = (1-e^{-T/K})*l/T + e^{-T/K}*R_{old}$$

where T *= packet interarrival time*

l *= packet size*

K *= constant*

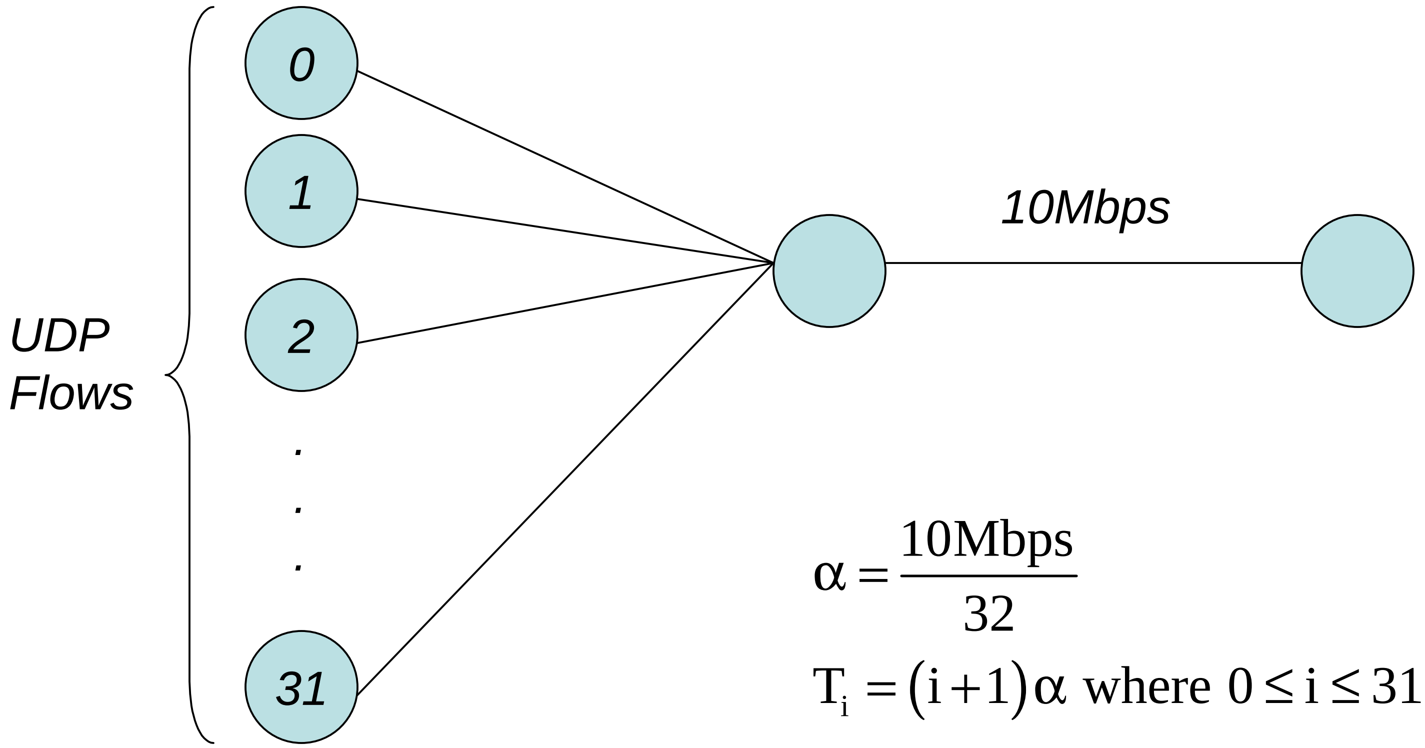
To summarize, *Edge routers* needs to

1) Classify the packet to a flow
2) Update the fair share rate estimation for the outgoing link
3) Update the flow rate estimation
4) Label the packet

Outline

- Introduction
- Background: Definitions and Previous Work
- CSFQ and its Algorithms
- **Simulations**
- Evaluations of CSFQ
- Conclusions and Future Work

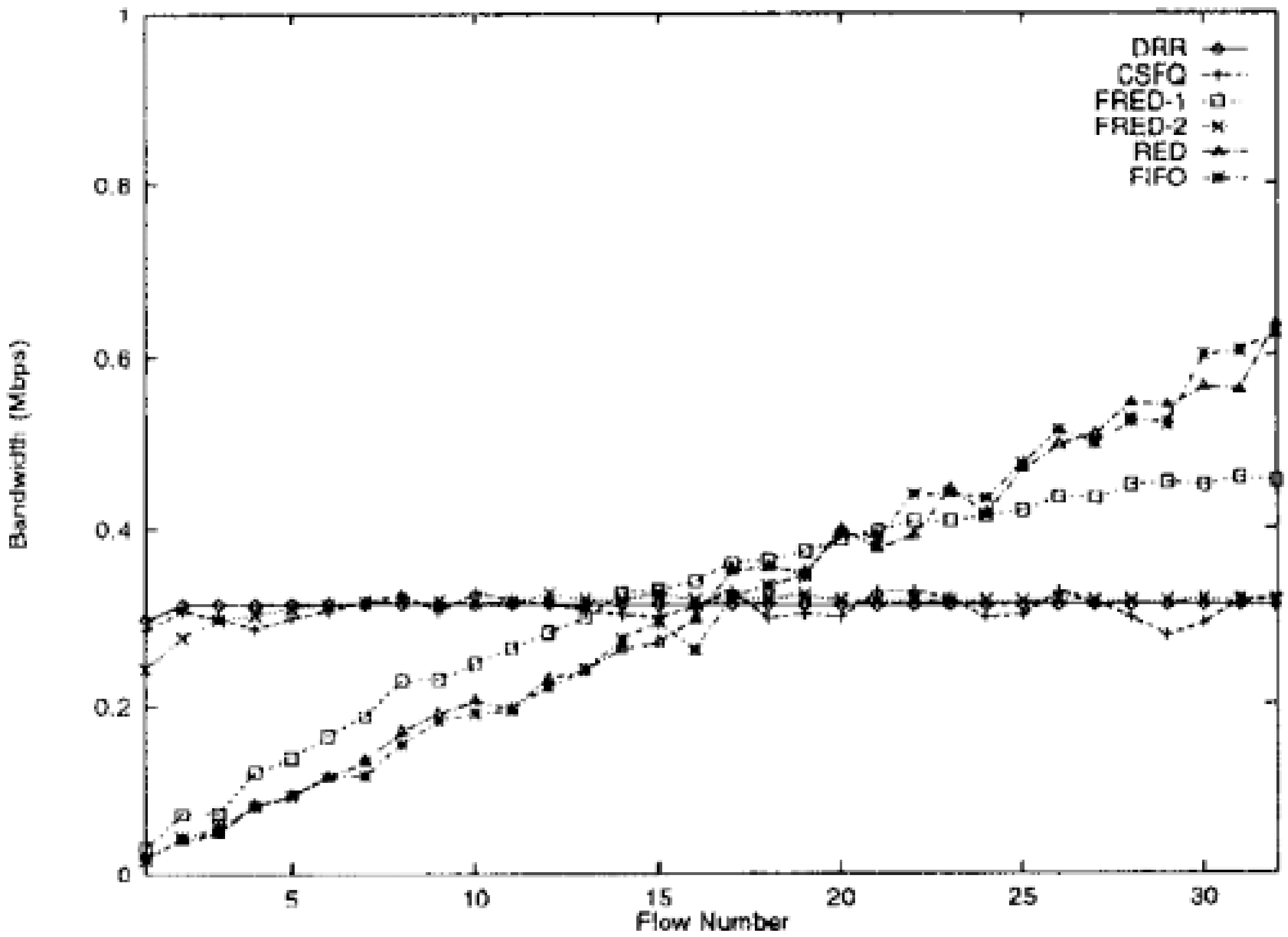
Simulations – Single Congested Link



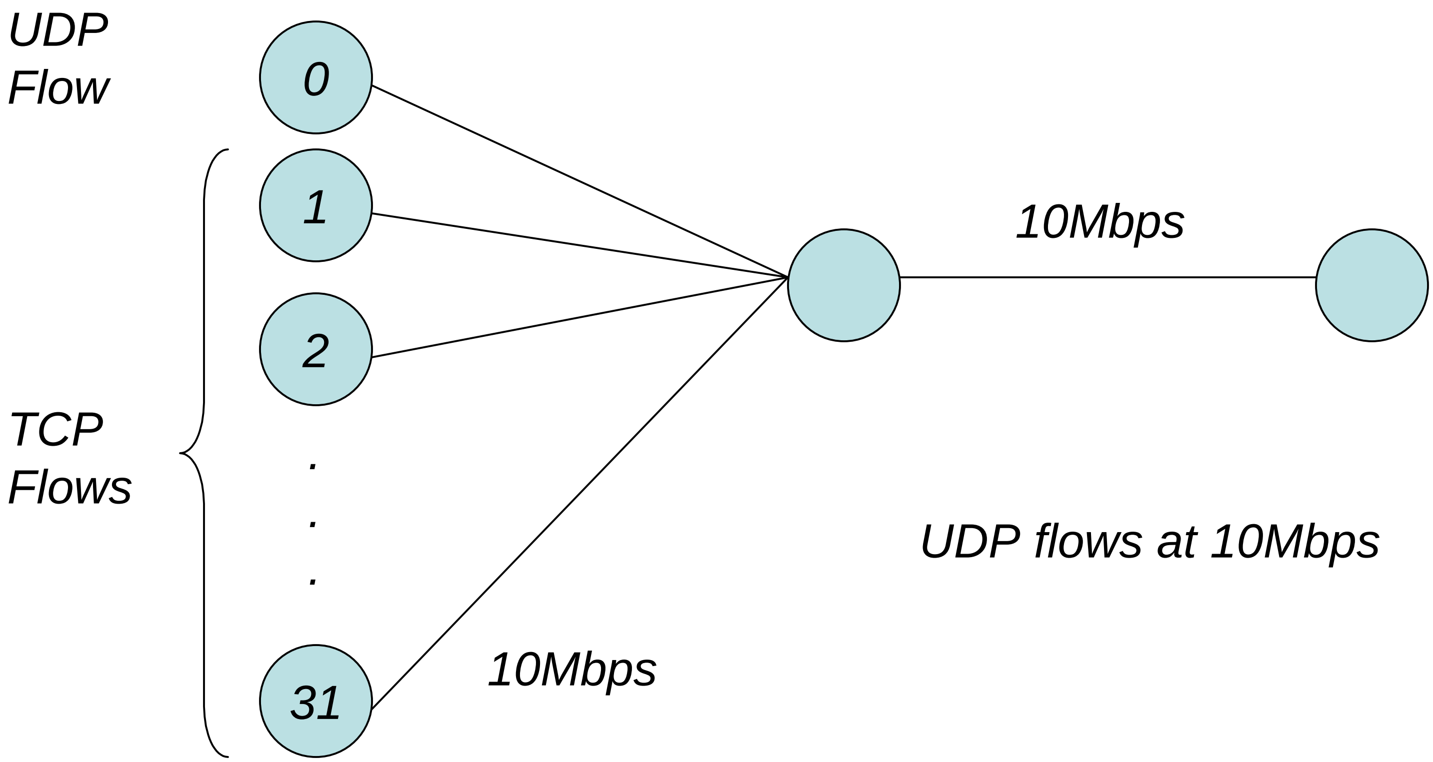
$$\alpha = \frac{10\text{Mbps}}{32}$$

$$T_i = (i+1)\alpha \text{ where } 0 \le i \le 31$$

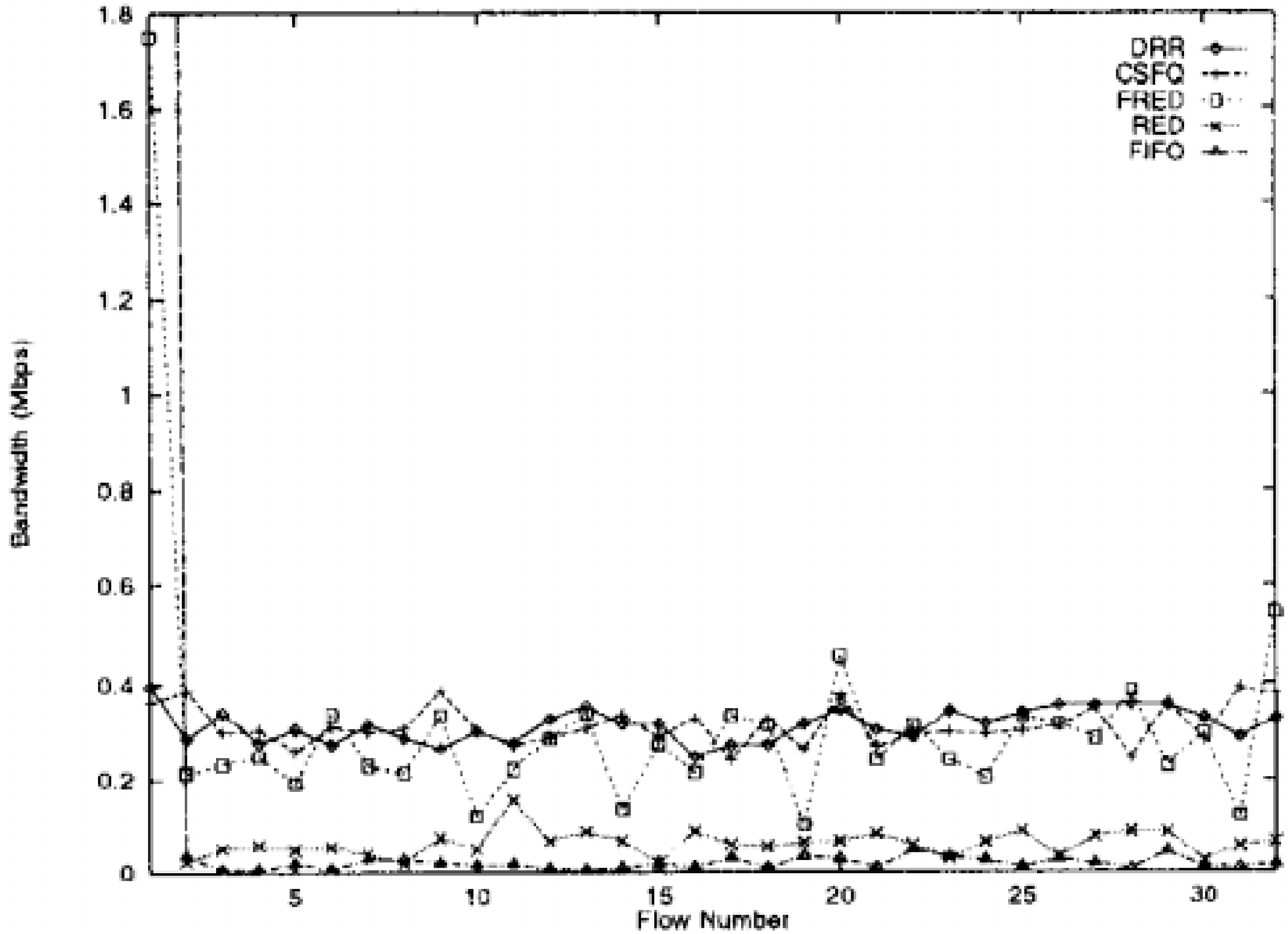
Simulations – Single Congested Link



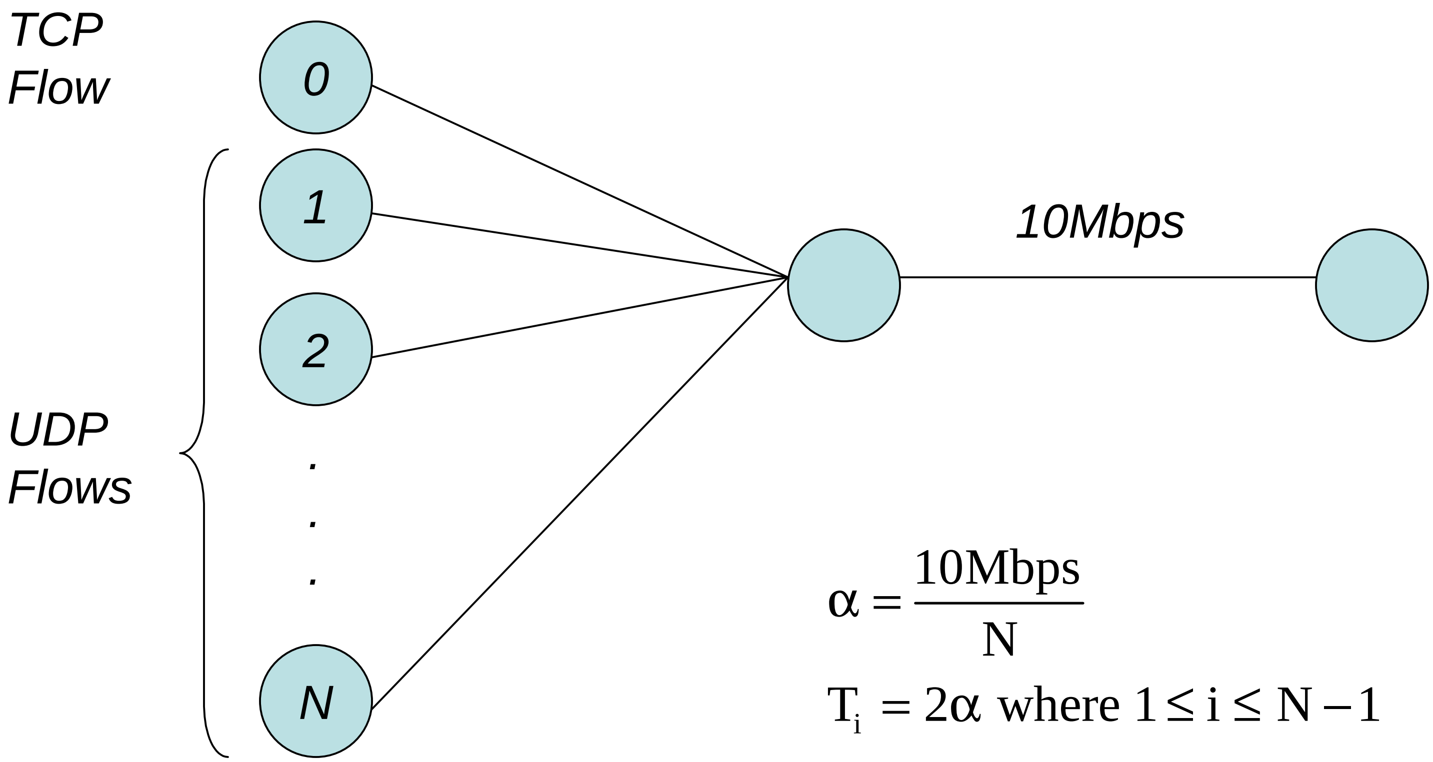
Simulations – Single Congested Link



Simulations – Single Congested Link



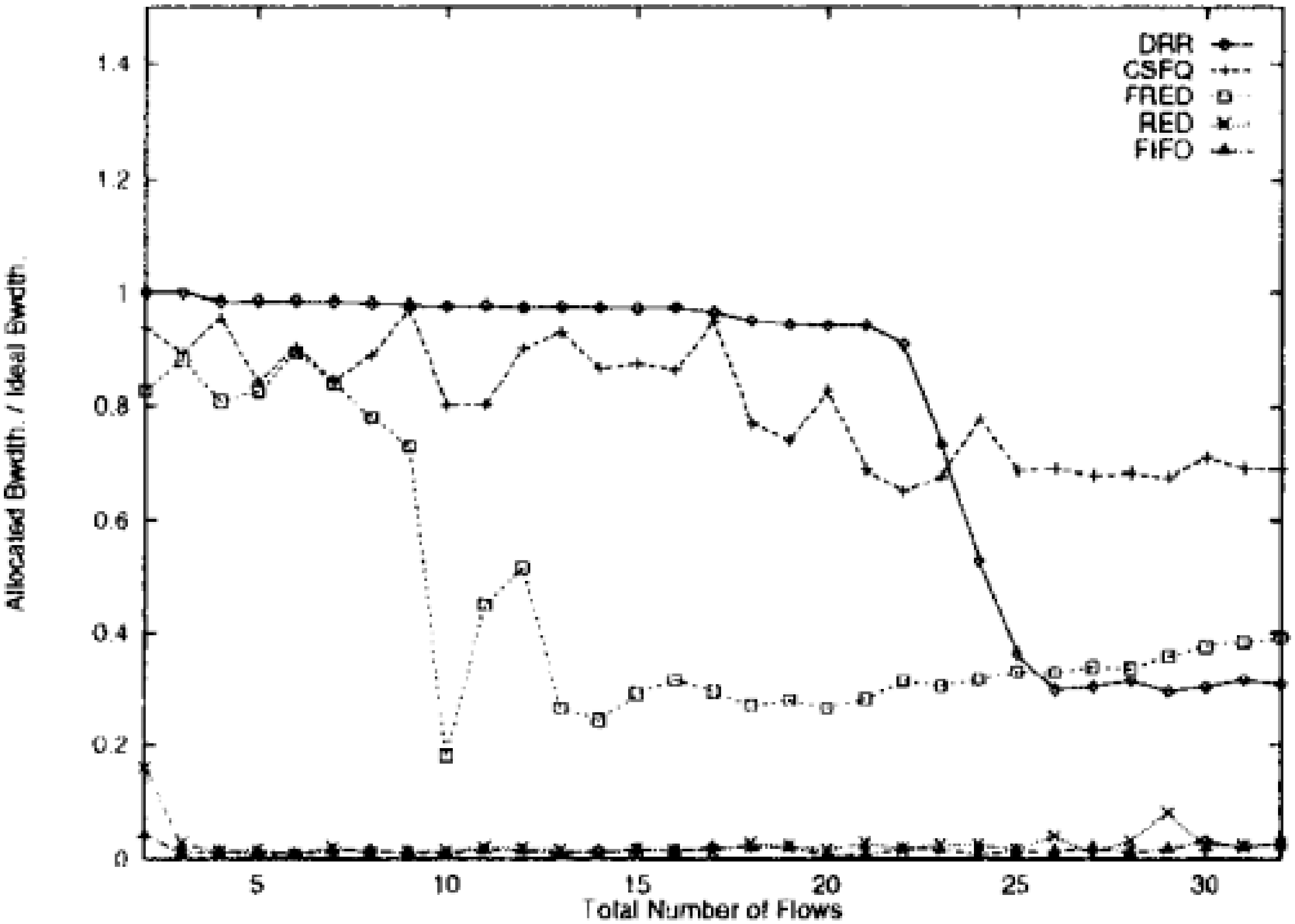
Simulations – Single Congested Link



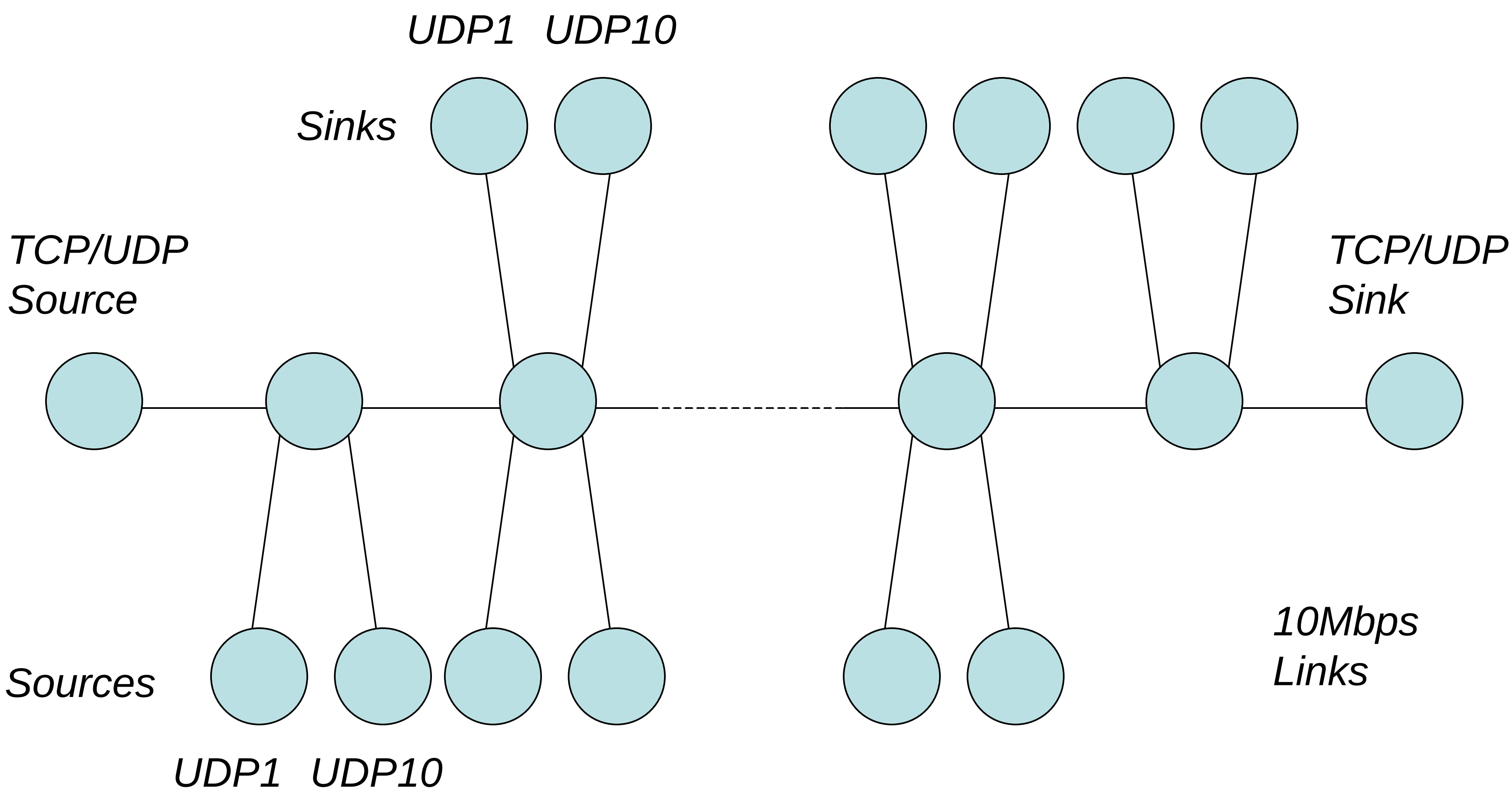
$$\alpha = \frac{10\text{Mbps}}{N}$$

$$T_i = 2\alpha \text{ where } 1 \le i \le N-1$$

Simulations – Single Congested Link

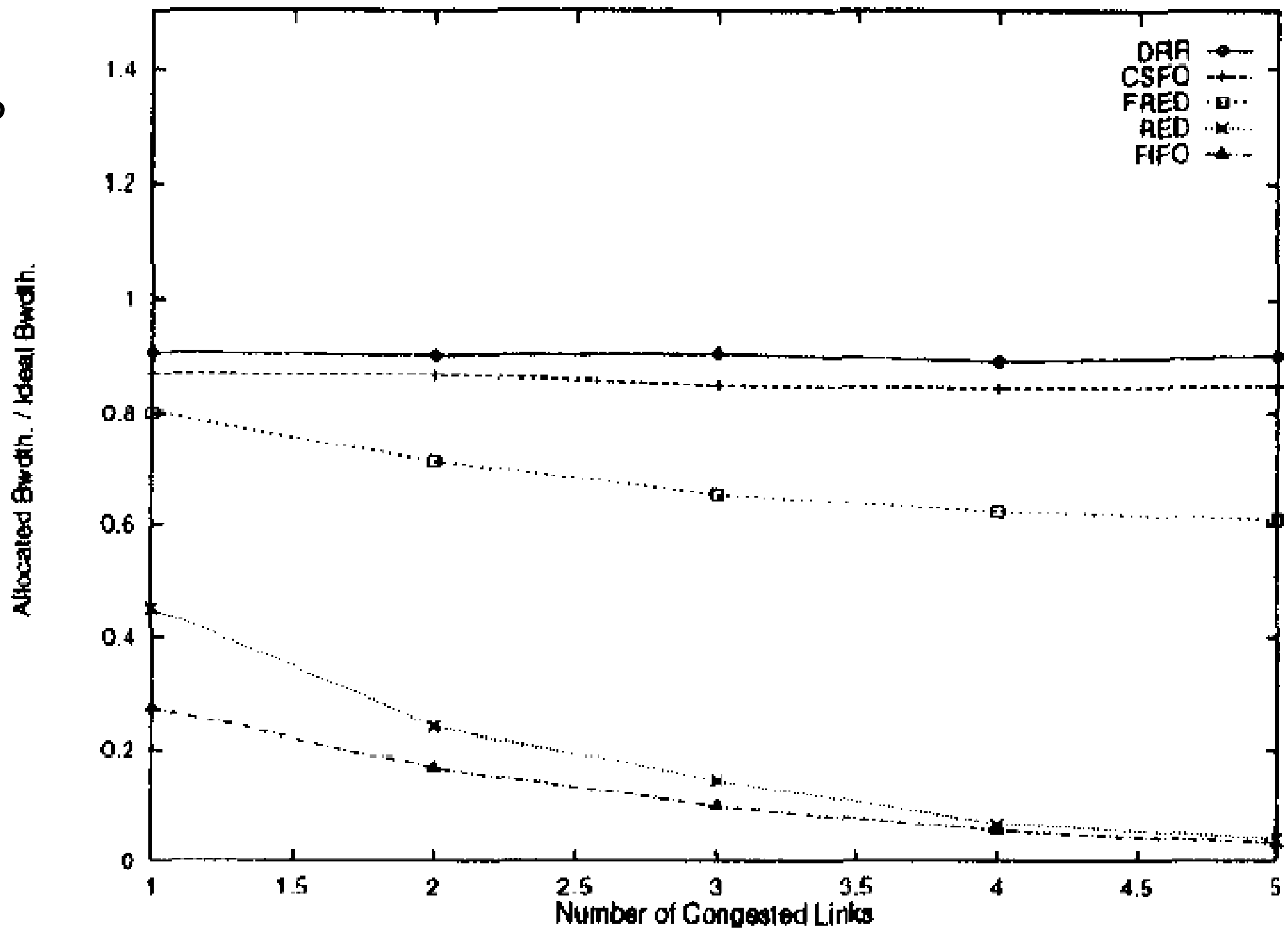


Simulations – Multiple Congested Links



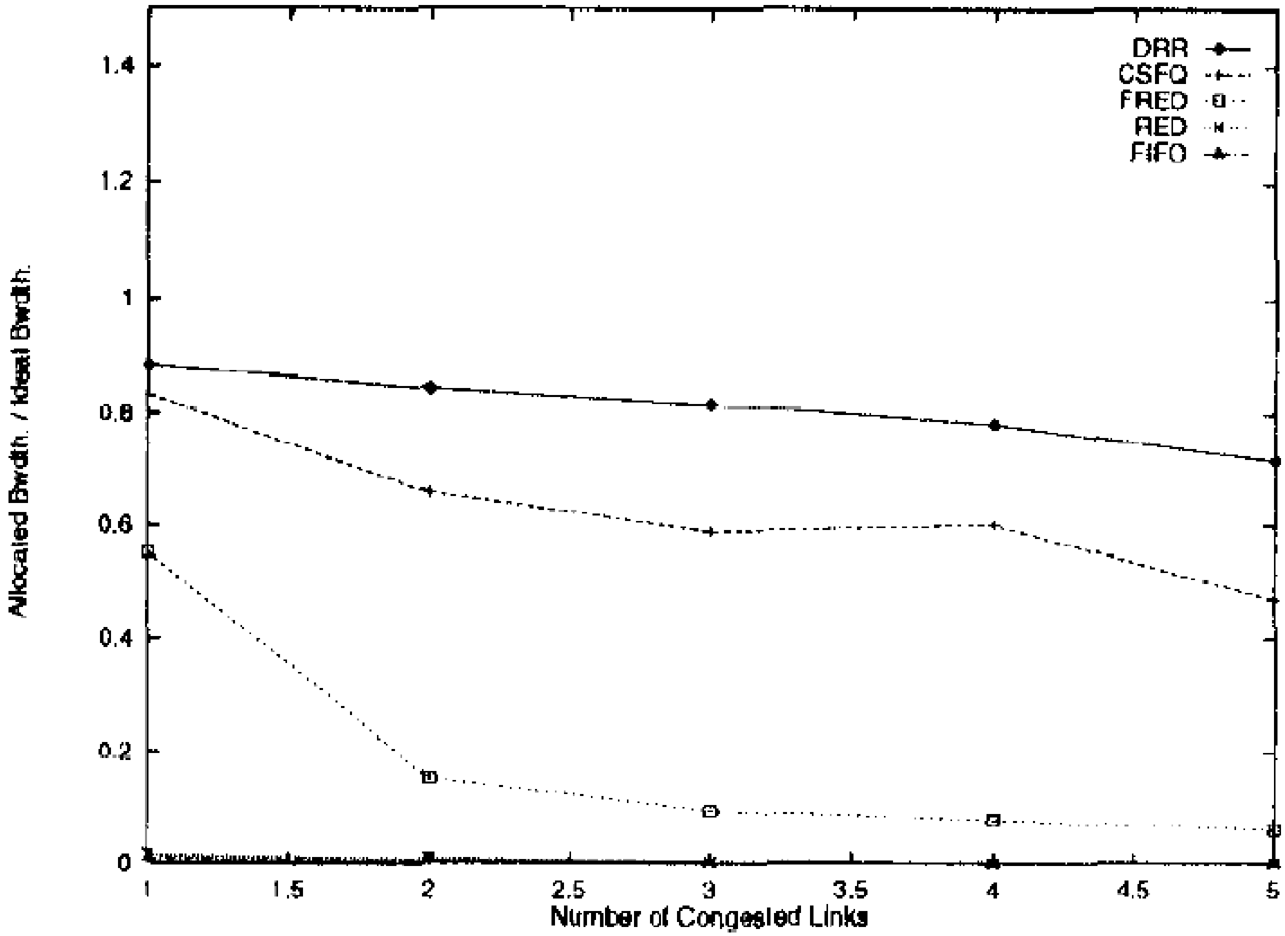
Simulations – Multiple Congested Links

UDP



Simulations – Multiple Congested Links

TCP

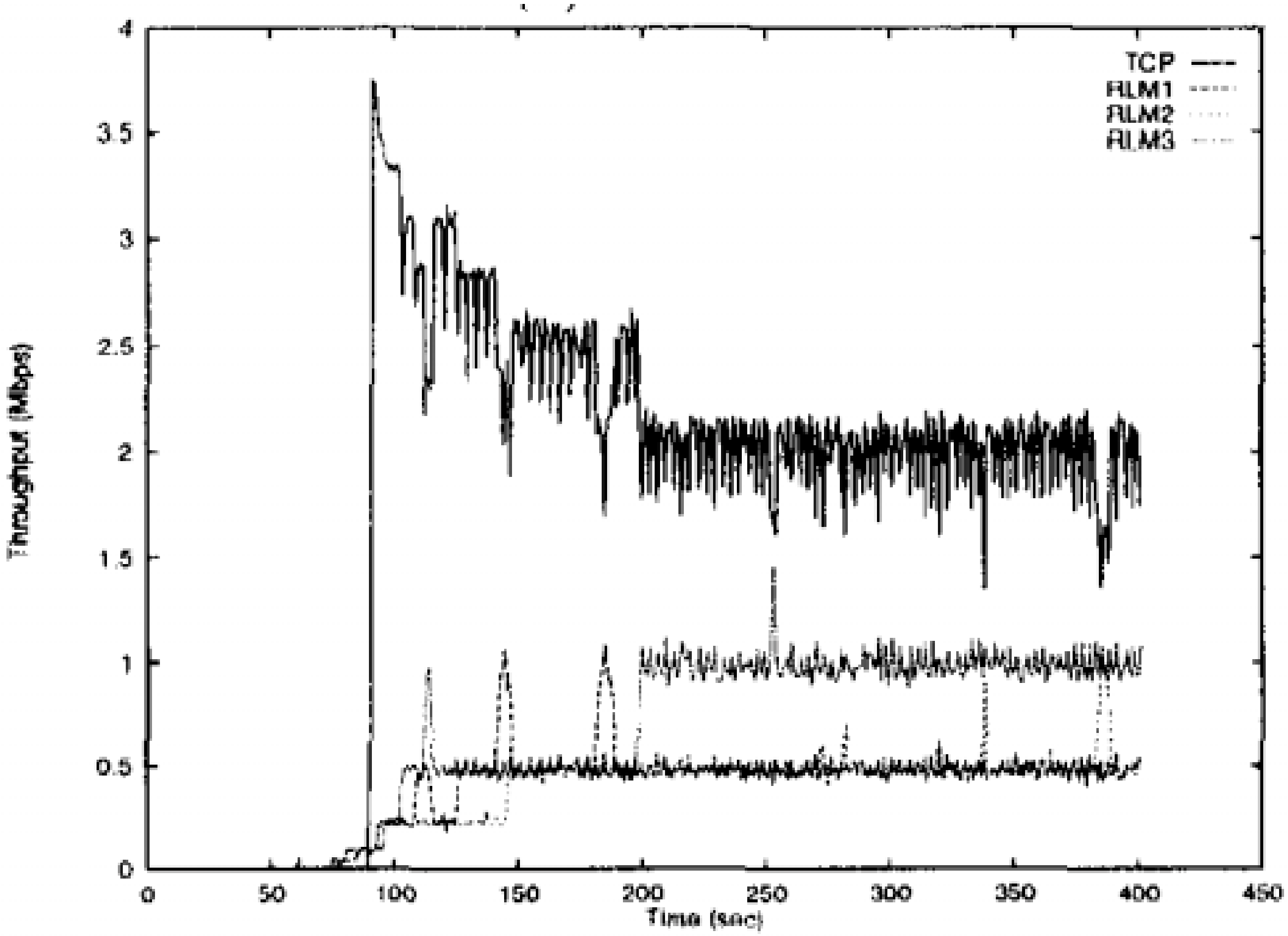


Simulations – Coexistence of Adaptation Schemes

- RLM (Receiver-driven Layered Multicast)
  - Only first 5 layers (~0.992Mbps)
  - TCP-friendly like
- 3 RLM flows and 1 TCP flow

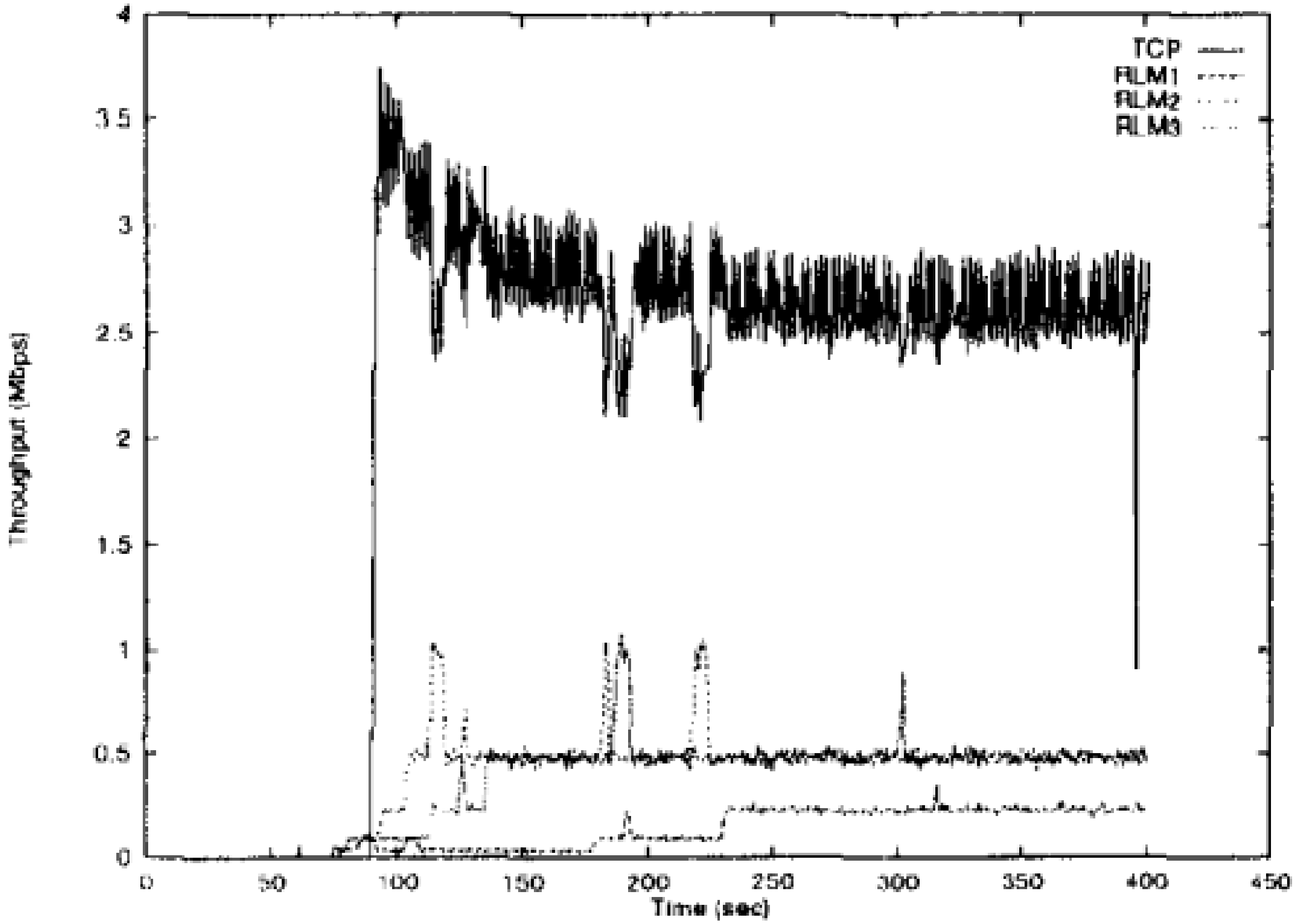
Simulations – Coexistence of Adaptation Schemes

FIFO



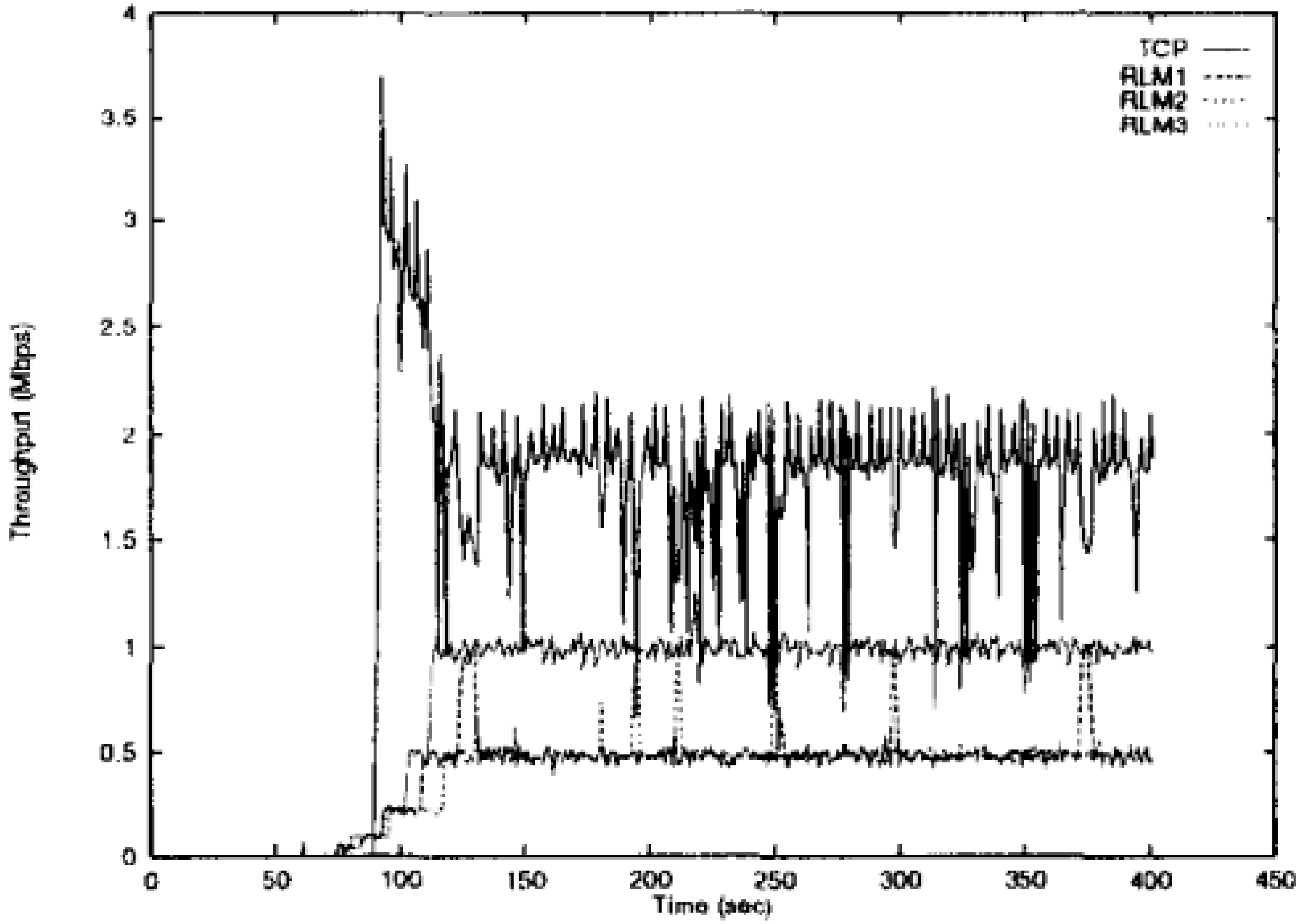
Simulations – Coexistence of Adaptation Schemes

RED



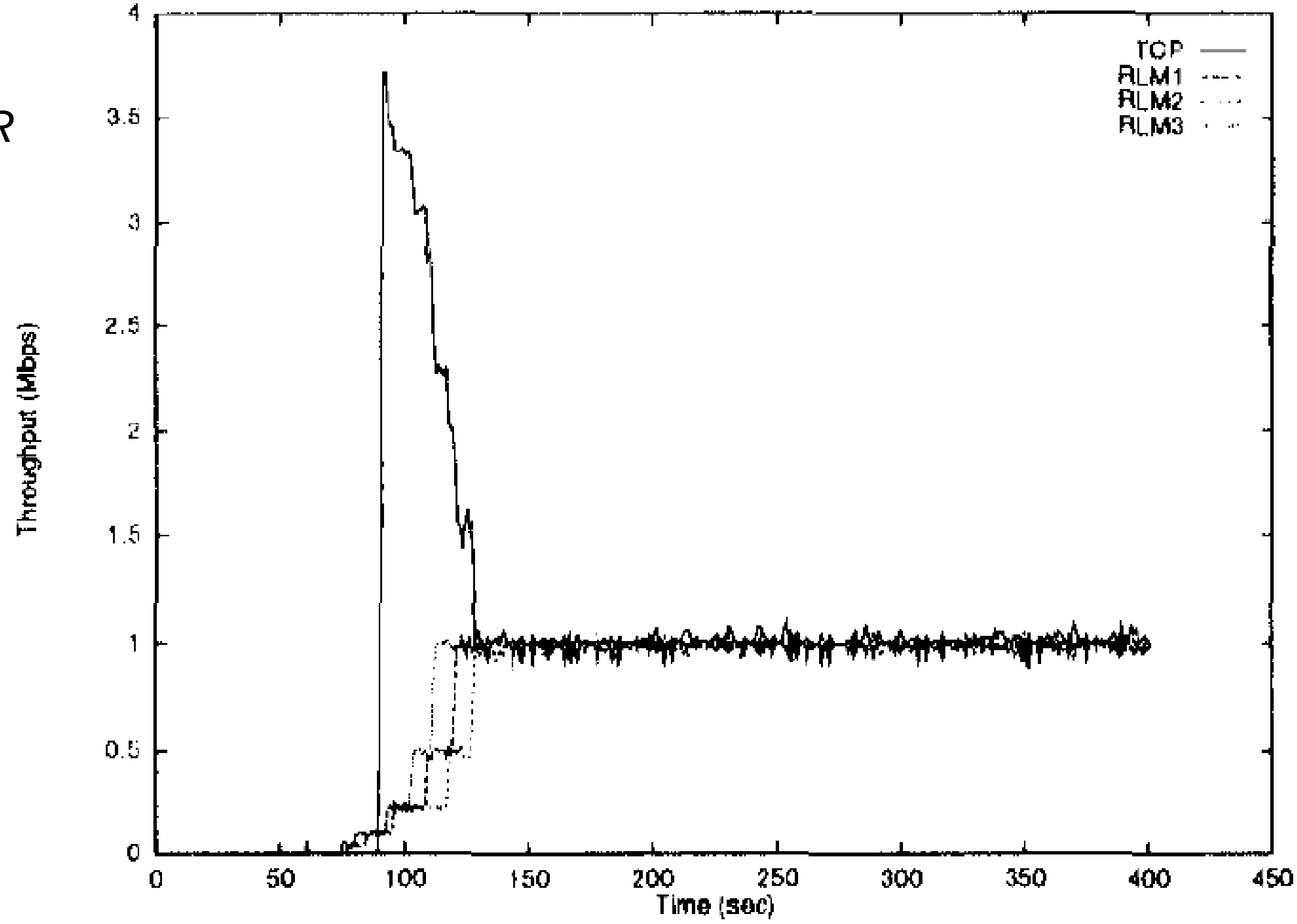
Simulations – Coexistence of Adaptation Schemes

FRED



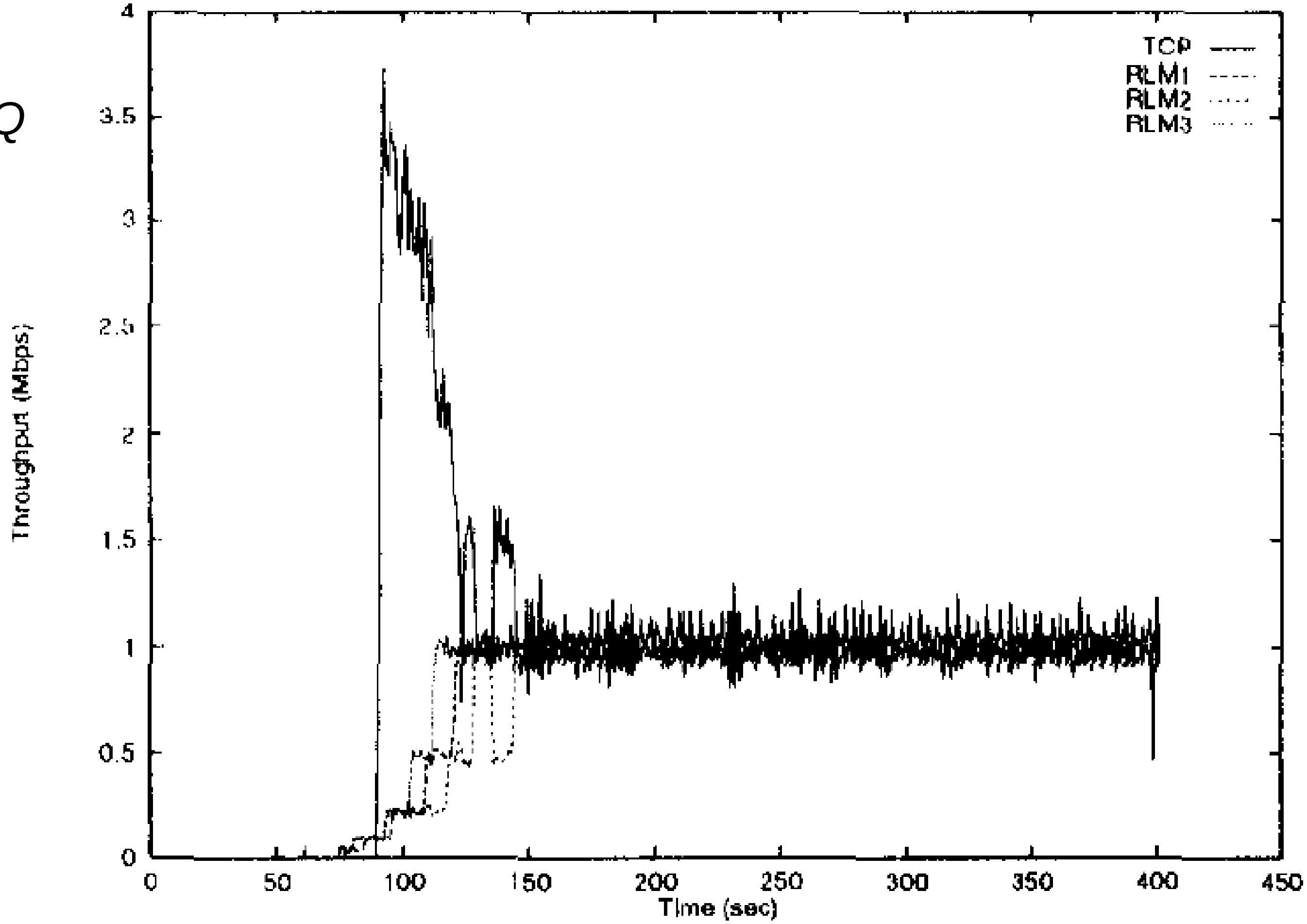
Simulations – Coexistence of Adaptation Schemes

DRR



Simulations – Coexistence of Adaptation Schemes

CSFQ



Simulations – Different Traffic Models

- 1 On/Off Flows
  - 100ms on, 1900ms off
  - Rate : 10Mbps
  - Sends 6758 packets
- 19 competing TCP flows

Simulations – Different Traffic Models

Algorithm	Delivered	Dropped
DRR	601	6157
CSFQ	1680	5078
FRED	1714	5044
RED	5322	1436
FIFO	5452	1306

Simulations – Different Traffic Models

- 60 TCP Flows
  - Exponentially distributed inter-arrival times with mean of 0.05ms
  - Pareto distributed transfer time with mean of 20 packets
- 1 UDP flow (10Mbps)

Simulations – Different Traffic Models

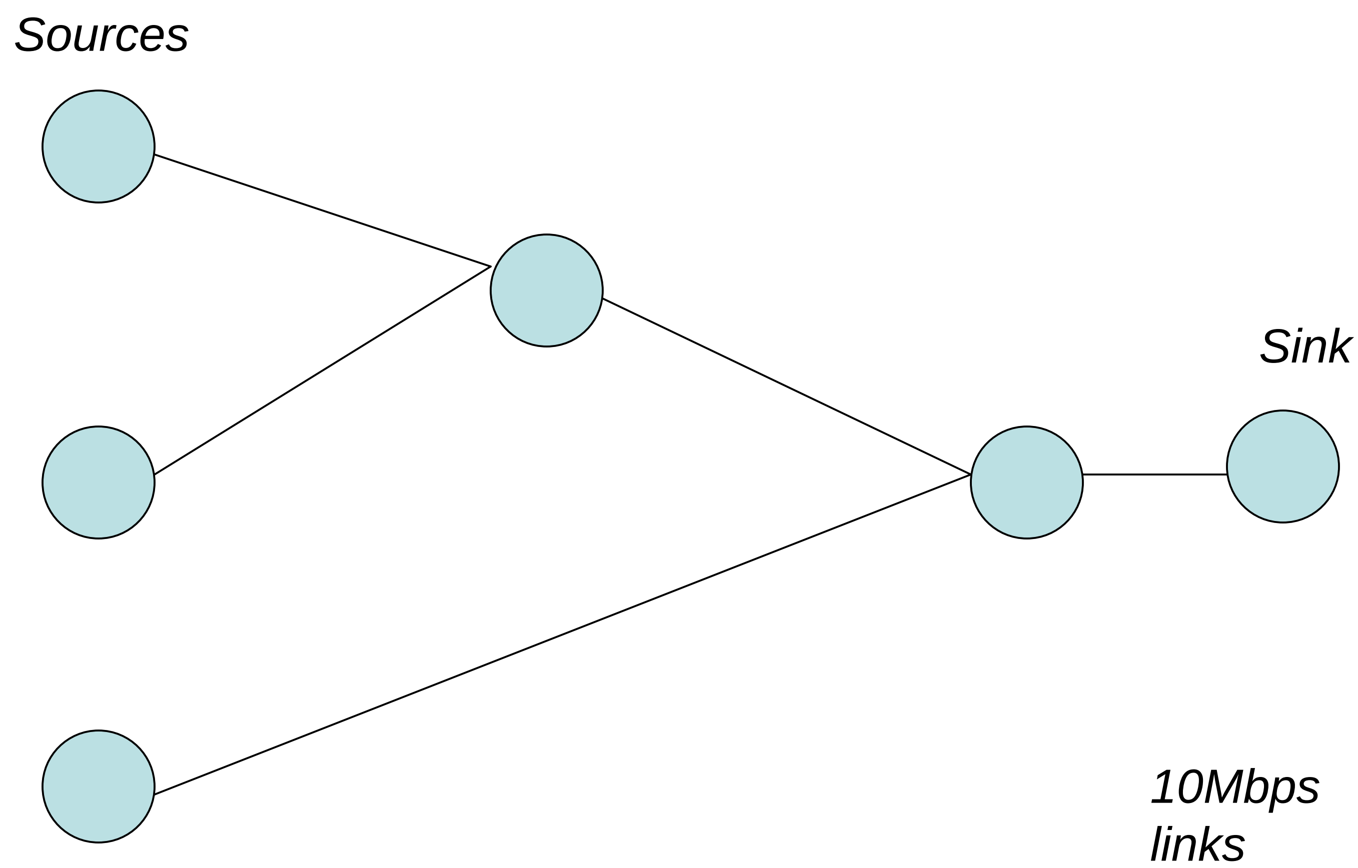
Algorithm	Mean time	Std. dev
DRR	25	99
CSFQ	62	142
FRED	40	174
RED	592	1274
FIFO	840	1695

Simulations – Large Latency

- 10Mbps link with 100ms latency
- 1 UDP flow at 10Mbps
- 19 TCP flows

Algorithm	Mean	Std. dev
DRR	6080	64
CSFQ	5761	220
FRED	4974	190
RED	628	80
FIFO	378	69

Simulations – Packet Relabeling



Simulations – Packet Relabeling

Traffic	Flow 1	Flow 2	Flow 3
UDP	3.36	3.32	3.28
TCP	3.43	3.13	3.43

Outline

- Introduction
- Background: Definitions and Previous Work
- CSFQ and its Algorithms
- Simulations
- **Evaluations of CSFQ**
- Conclusions and Future Work

Evaluations of CSFQ

- Reasonable approximation of fair share
- Roughly comparable performance to FRED
  - Sometimes much better than FRED
  - Note : FRED has per-packet overhead
- Not quite as fair as DRR

Outline

- Introduction
- Background: Definitions and Previous Work
- CSFQ and its Algorithms
- Simulations
- Evaluations of CSFQ
- **Conclusions and Future Work**

Conclusions and Future Work

- CSFQ
  - rate-based active queue management
  - Rate estimation at the edge and packet labels for core routers
- Large latency effect
- Possible extension of CSFQ for QoS

Back-up Slide(s)

- Slide 2
  - Ion Stoica – research interest is to develop techniques and architectures that allow powerful and flexible network services to be deployed in the Internet without compromising its scalability and robustness.
  - Scott Shenker - The working group will focus on defining a minimal set of global requirements which transition the Internet into a robust integrated-service communications infrastructure.
- Slide 4
  - Congestion today (1998) is controlled by end-hosts (TCP)
  - FQ – has to maintain state, manage buffers, perform packet scheduling on per-flow basis.
- Slide 8
  - SFloyd, Jacobson, 93. For long-lived TCP connections like file transfer, it might make a difference.
- Slide 9
  - Dong Lin, Robert Morris in 1997 – works well with different traffic – TCP and UDP etc.
- Slide 10
  - DDR – Deficit Round Robin or WFQ.
- Slide 21
  - Exponential average to estimate the rate of flow since this closely reflects a fluid averaging process which is independent of the packetizing structure. And the solution is bounded as it converges to a real value.