Approximate Fairness through Limited Flow List

Addisu Tadesse Eshete

Yuming Jiang

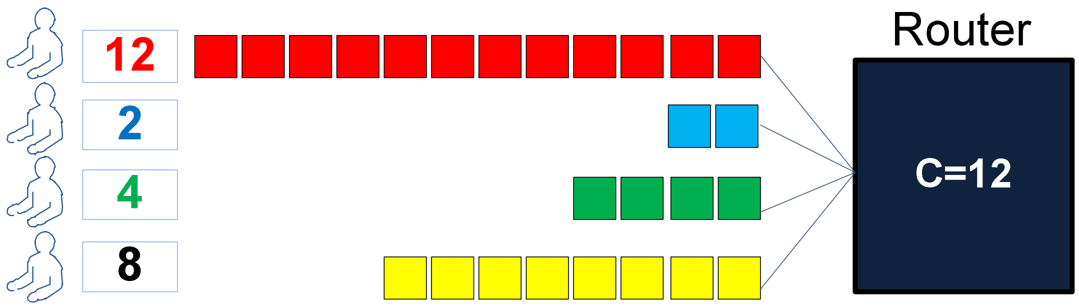
23rd International Teletraffic Congress, San Francisco, USA

September 7, 2011

Outline

- The Research Problem
- Background and Motivation
- Approximate Fairness through Partial Finish Time
  - Main idea
  - Results
- Conclusion

The Rearch Problem



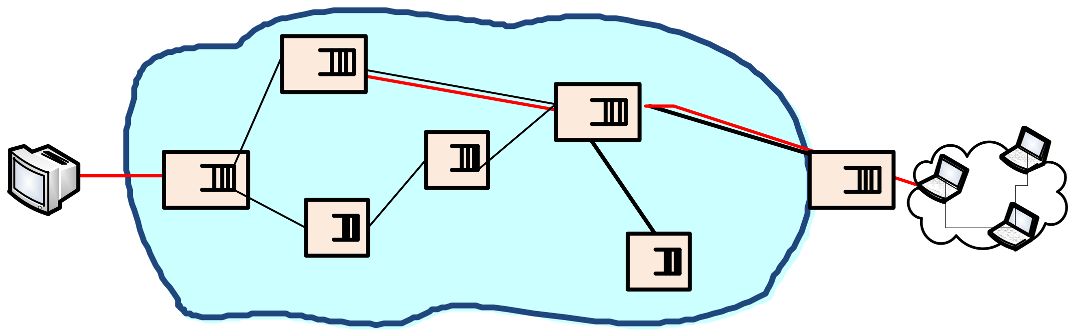
- Max-min Fairness (congestion)

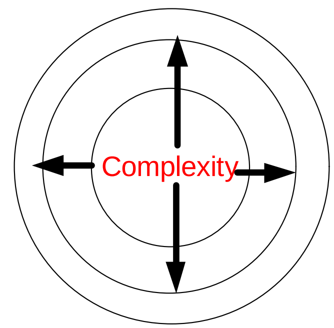
$$C = \sum_{f} \min(r_{fair}, r_f)$$

Background & Motivation

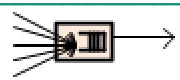
E2E design principle

FIFO Routers + E2E Congestion Avoidance





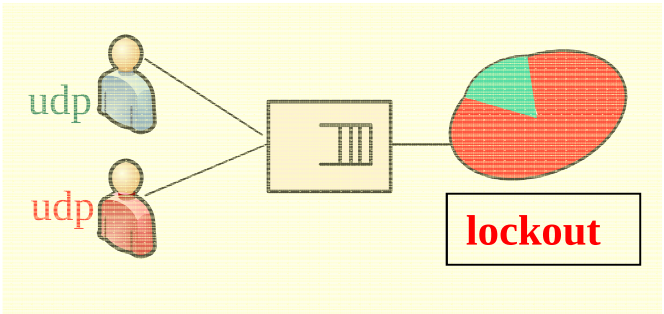
FIFO Routers

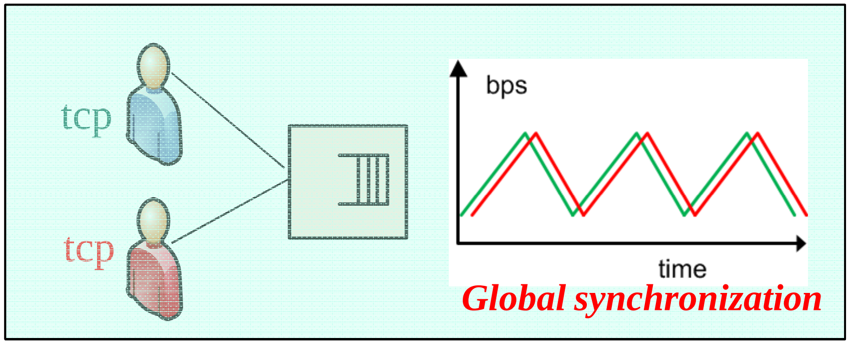
- Simple 
- Stateless: «fate-sharing»
- Robust: «survivability»
- Scalable: 100x '95 to '09

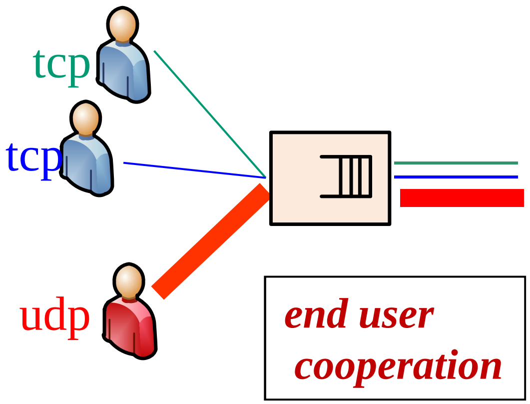
E2E Congestion Avoidance

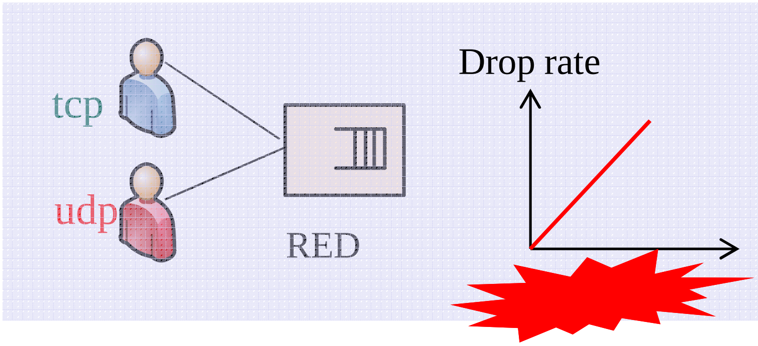
FIFO Routers + E2E Congestion Avoidance (CA)

- Simple *best effort* point-to-point *egalitarian* forwarding
- *No service differentiation or flow protection*





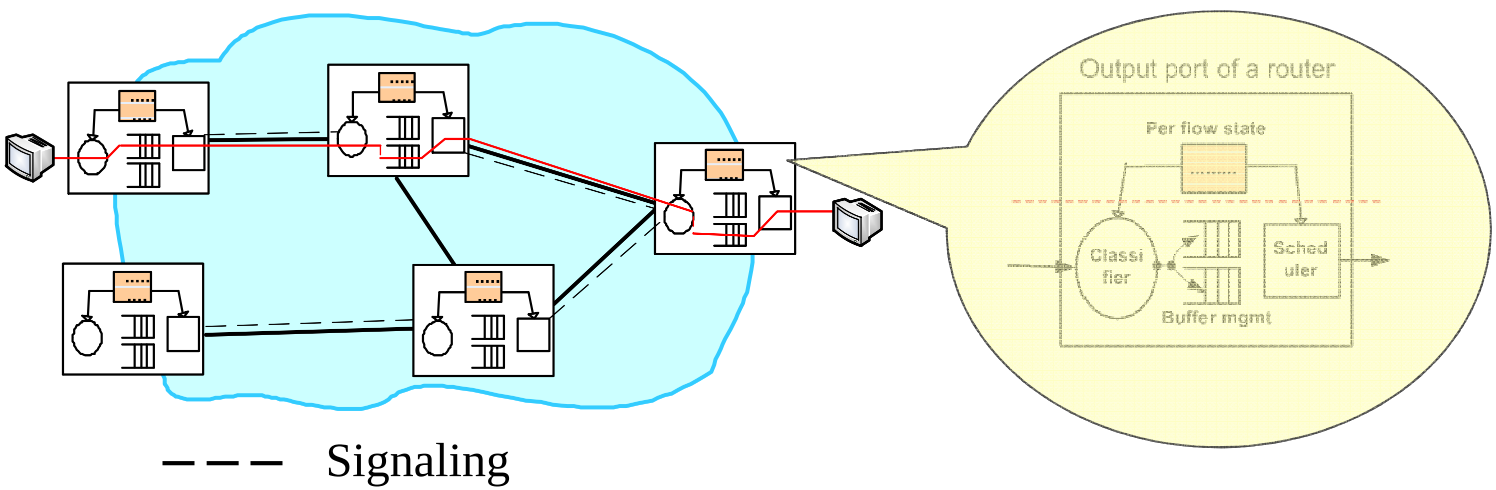




Per Flow Fair Queueing

- **IntServ - Intelligence in the network (routers)**

- *Fine grained flow state stored* and maintained at routers
- *Signaling* for resource reservation and admission control



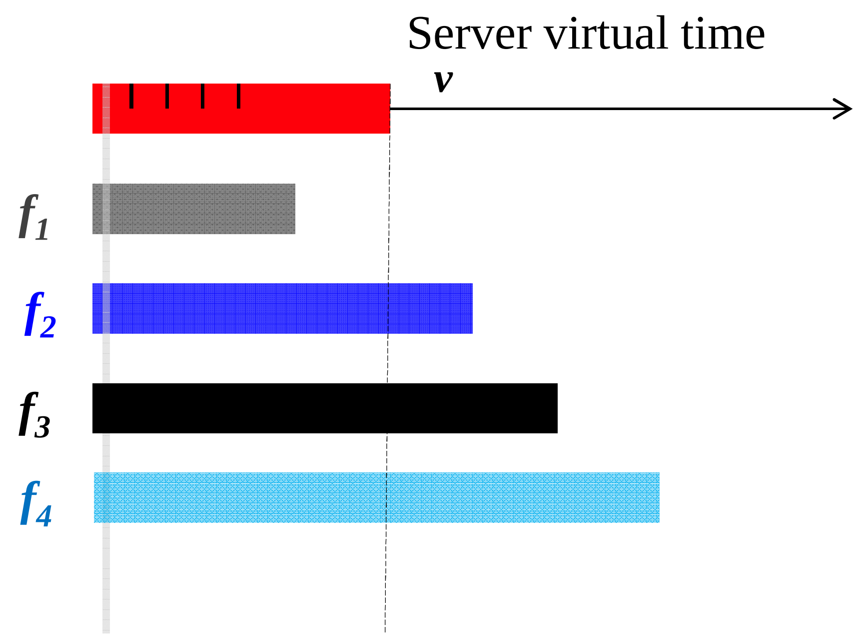
PFFQ Implementation

- Based on Packet Tags:
  - Start and Finish tags on packet arrival
  - Server virtual time v(t)
  - **Examples: WFQ, WF^2Q, WF^2Q+, SFQ, SCFQ**

$$S(p_f^j) = \max\left(v\left[A(p_f^j)\right], F\left(p_f^{j-1}\right)\right)$$

$$F(p_f^j) = S(p_f^j) + \frac{l_f^j}{r_f}$$

$$where\ F(p_f^0) = 0.$$



NTNU

Per Flow Fair Queueing may be complex!!

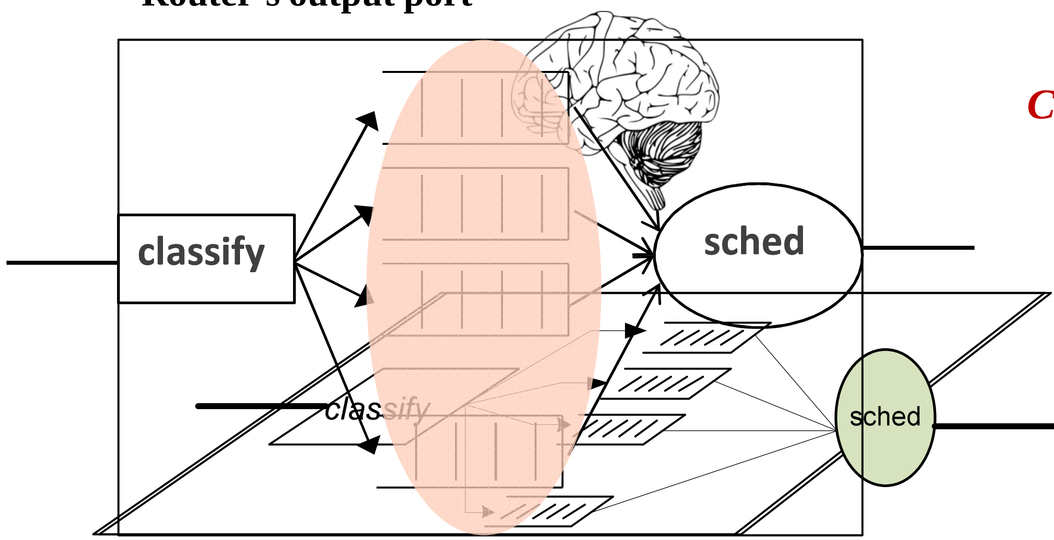
Q2S

$$S(p_f^j) = \max\left(v\left[A(p_f^j)\right], F\left(p_f^{j-1}\right)\right)$$

$$F(p_f^j) = S(p_f^j) + \frac{l_f^j}{r_f}$$

$$where\ F(p_f^0) = 0.$$

Router's output port



Control plane

Data plane

Buffer partitioning

Our Research Question

Can we design stateless or partially stateful single queue architectures to enforce fairness (flow protection)?

NTNU

Related works: repeat

Q2S

Existing proposals:

- **RED/REDvariations (one aggregate FIFO queue)**
- **PFFQ (one FIFO queue per flow *)**
- **CSFQ (one aggregate non-FIFO queue, dynamic packet state)**

Issues with existing protocols:

1.Stateful and ***complex***

– SFQ, WFQ

2.Target ***specific*** traffic type

– Fair RED—TCP

– CSFQ—UDP

3.Less Robust—require proper router configurations

– CSFQ

Approximate Fairness through Partial Finish Time

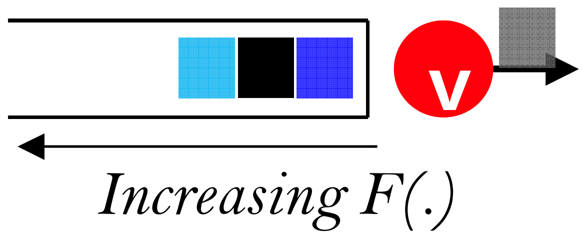
«approximate fairness of PFFQ with partial state»

Key AFpFT components

1. Virtual tag computation and maintenance
   - E.g., Limited flow data in FLOW LIST
2. Flow and router relationship (see router roles in paper)
   - Edge or core router per flow
3. Buffer management (see paper for details)
   - Recovering impaired fairness for high rate flows
   - Avoiding loss synchronization

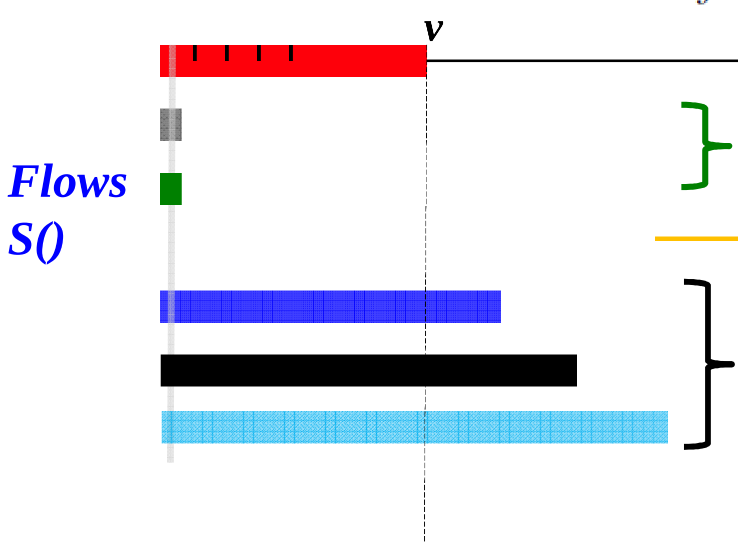
AFpFT: Insights for Tag computation

Insight 1



Insight 2

$$S(p_f^j) = \max\left(v[A(p_f^j)],\ F(p_f^{j-1})\right)$$



$$S(p_f^j) = v[A(p_f^j)]$$

Encoding 2 // cheap

$$S(p_f^j) = F(p_f^{j-1})$$

Encoding 1 //expensive

From the flow record in flow list.

AFpFT has two tag encodings

$$S(p_f^j) = \max\left(v[A(p_f^j)], F(p_f^{j-1}) \right)$$

Cheap encoding

- Directly plucked from server
- New flows
- Flows rarely fill up the buffer
- Relative small and TCP friendly flows
- Brought to queue front

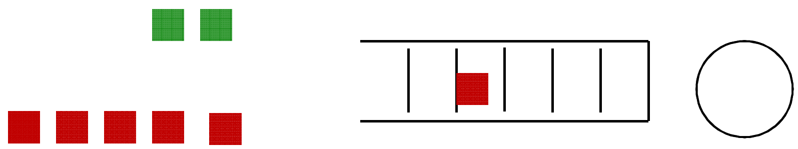
Expensive encoding

- *F(.)* taken from Flow list ***FL***
- Flows that occupy the buffer frequently
- Fast, big and less TCP friendly
- Pushed to queue tail
- ***FL*** size ~ Buffer size

- Traditional routers buffer capacity= RTT × C_bottleneck
- Fast, big flows are a minority!!

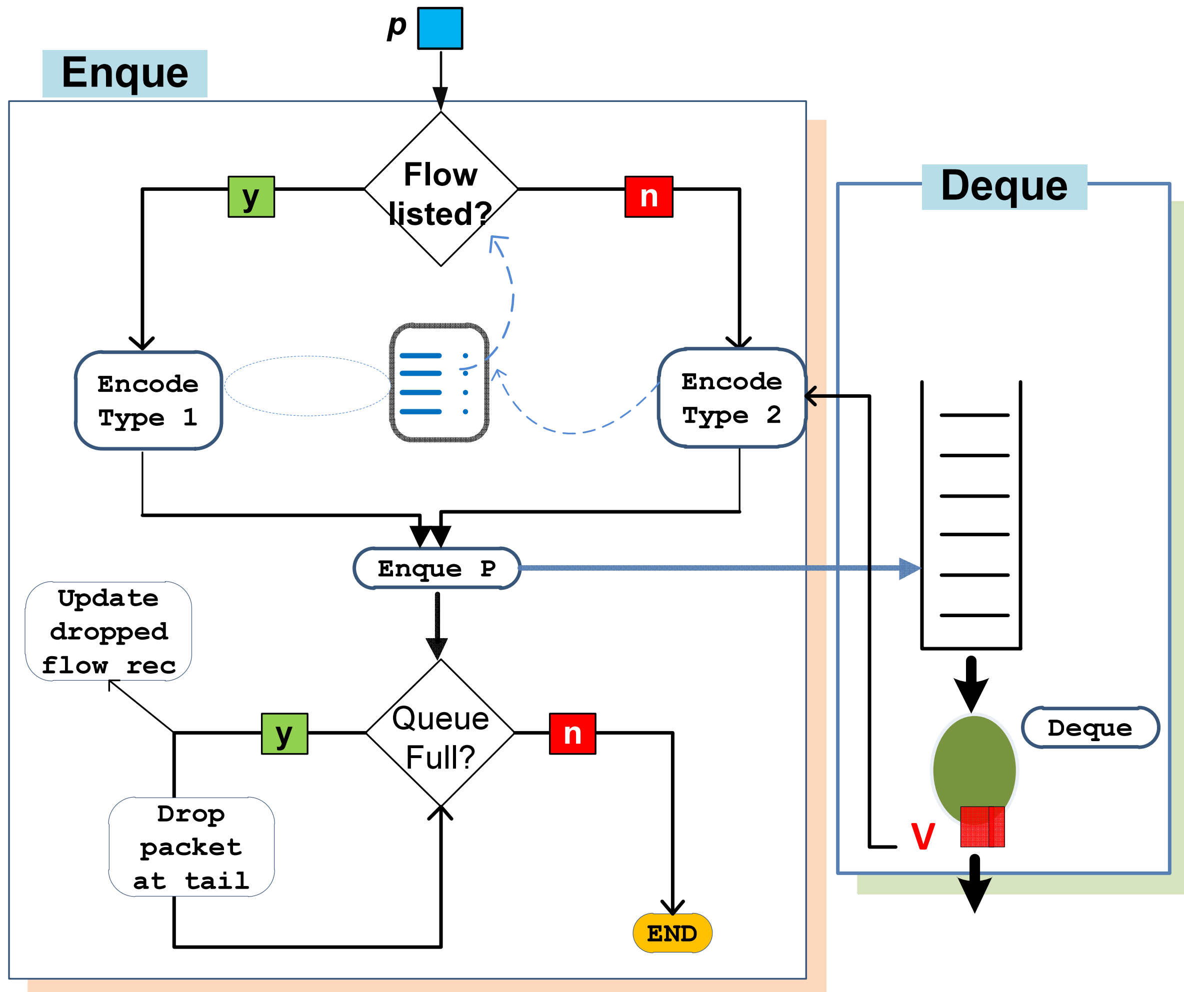
Other Details

- Frequently buffered flows
  - Also ***listed*** **and** ***high rate*** flows
  - High rate flows are ***minority*** in the Internet

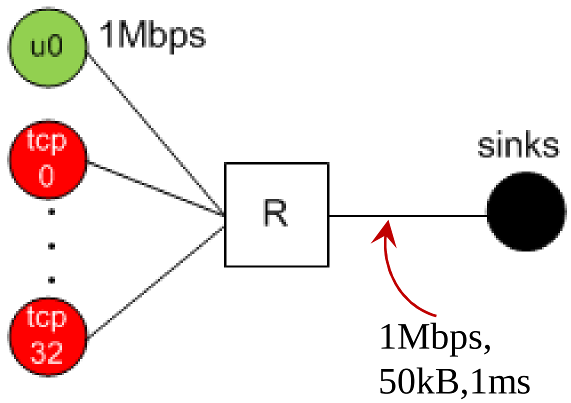


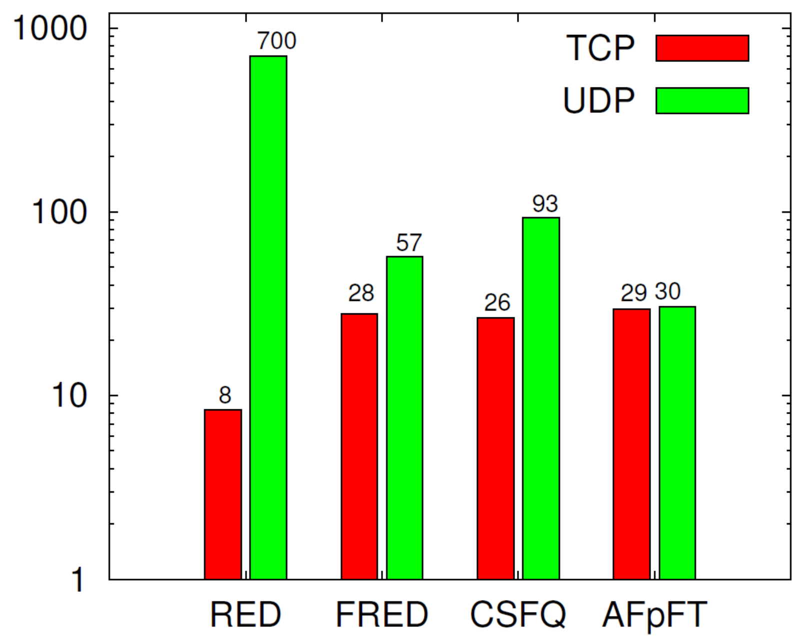
- Router roles
  - Edge: keep state for all flows
  - Inner: keep state of buffered flows only
- Update flow list:
  - Packet dropped: decrement packet's flow contribution
  - No flow packet in buffer: remove flow from list

AFpFT: Detailed architecture



Selected Results/Single Link





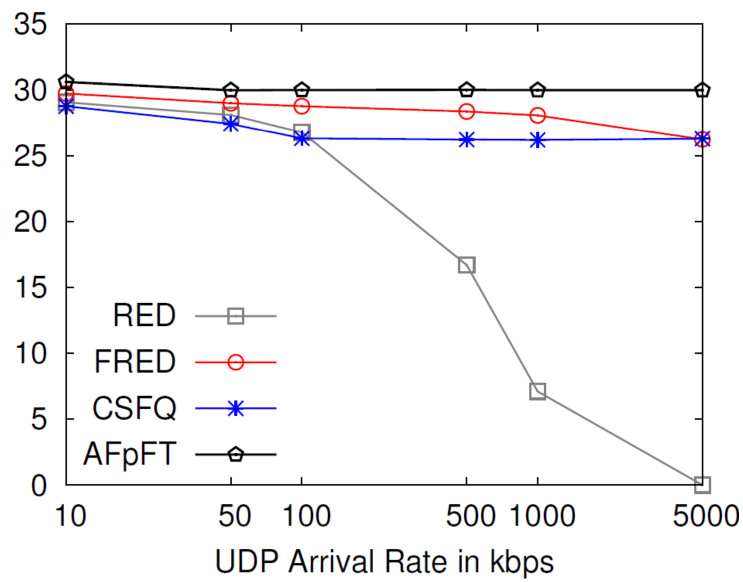
Jain's Fairness Index /TCP/	
RED	*0.9329*
FRED	*0.9905*
CSFQ	*0.9994*
AFpFT	*0.9999*

$$F=\frac{\left(\sum_i x_i\right)^2}{n\sum_i {x_i}^2}$$

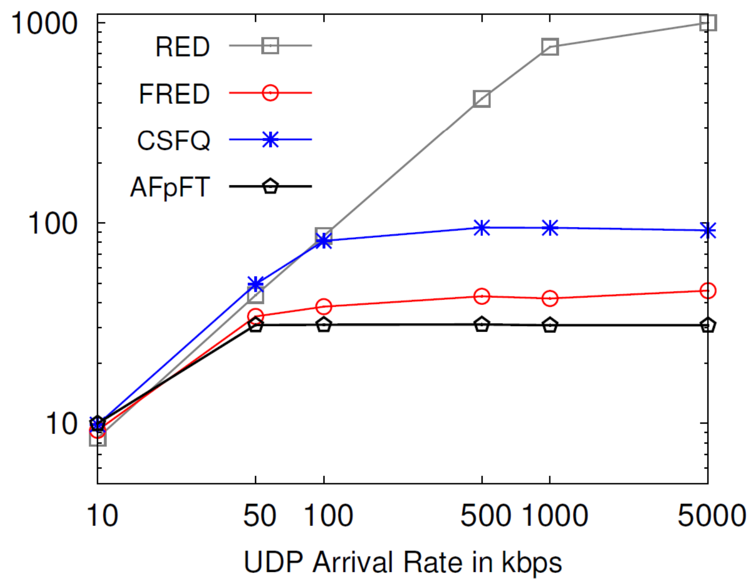
Selected Results/Single Link

- We steadily increase UDP rate to 5Mbps

Average TCP throughput



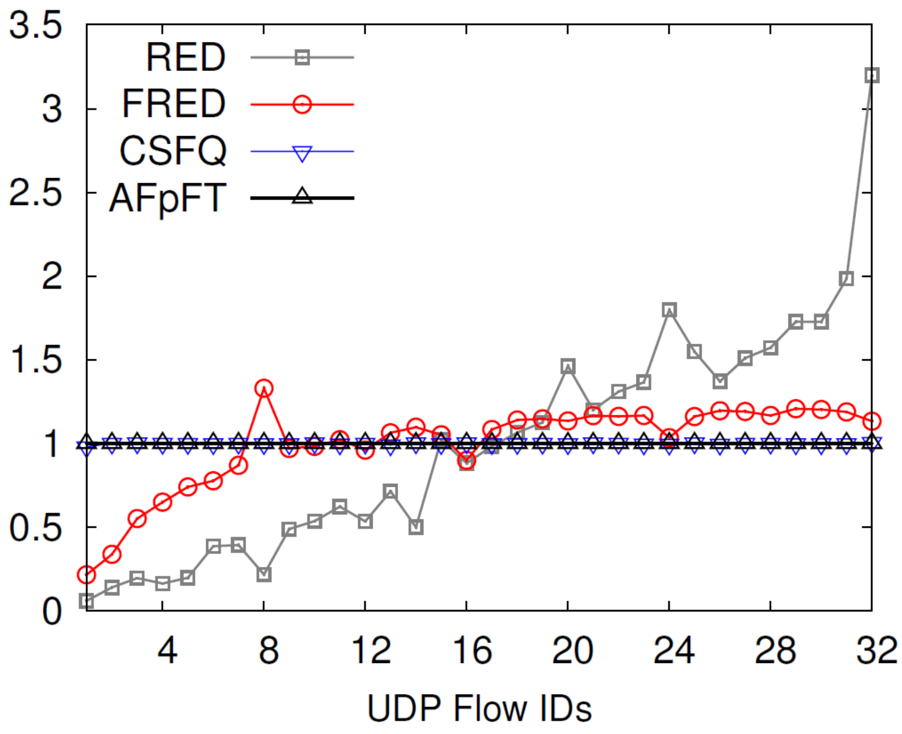
UDP throughput (logscale)



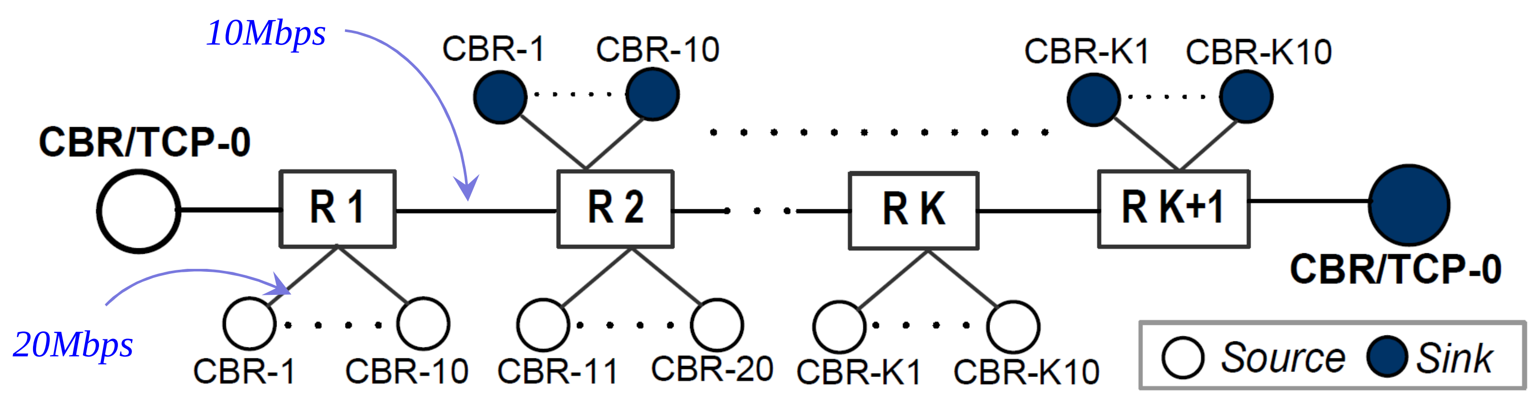
Selected Results/Single Link

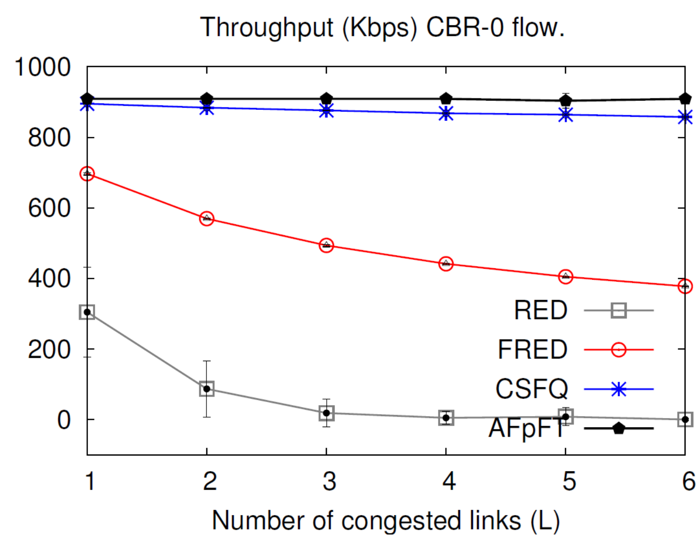
- Only CBR flows
- Flow i sends at $0.3125*i$ Mbps ($i=1, ..., 32$)

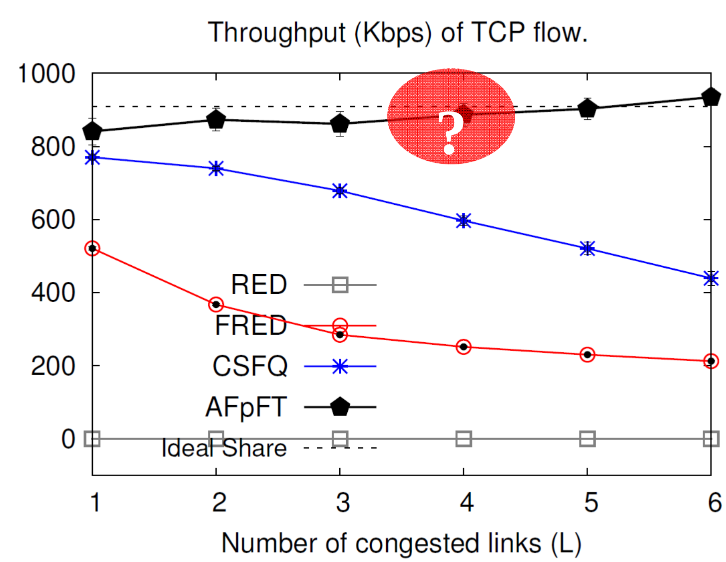
Normalized throughput



Selected Results/Multiple Congested Links







Conclusion

- AFpFT:
  - Partially stateteful but highly fair
  - Robust cf. CSFQ
  - Scheduling integrated with buffer management
  - No loss synchronization as in aggregate queues
  - Comparable flow state to AFD (state for minority high rate flows)
  - Time priority to small and well behaved flows
  - Better fairness than FRED, CSFQ

Approach	State Info	Queue Nr.	Impl. Chall.	Fairness
PFFQ	Stateful	One per flow	Dynamic queue nr	Excellent
RED	Stateless	One FIFO queue	Parameter config	Good – hom.t. Bad – het.t.
FRED	Local partial state	One FIFO queue	Parameter config	Very good
CSFQ	Stateless router, stateful pkt	One non-FIFO q.	Dynamic pkt state	UDP- Excellent TCP – Very Good
AFpFT	Local partial state	One non-FIFO q.	?	Excellent