Be fair to flows!

A fair network is attractive and trustworthy and far more than just adequate.

Jim Roberts
IRT-SystemX, France

Revisiting congestion control, active queue management (AQM) and packet scheduling

- new interest is arising from perceived problems in home networks (bufferbloat) and data center interconnects
- a new IETF working group will define an AQM that
  - "minimizes standing queues"
  - "helps sources control rates without loss (using ECN)"
  - "protects from aggressive flows"
  - "avoids global synchronization"
- AQM performance depends significantly on what congestion control is implemented in end-systems
  - high speed TCP, low priority TCP
  - new congestion control protocols for the data center (DCTCP,...)
  - new algorithms at application layer (QUIC,...)

AQM + congestion control is not the answer!

- how can a network continue to rely on end-systems implementing the right congestion control?
  - why should users comply?
  - or course, they don't!
- instead, impose per-flow fairness in the core...
  - this is scalable, feasible and sufficient
- and enhanced flow-aware scheduling at the edge
  - since a fair share may not be sufficient
  - priority to a video stream, priority to Dad!

Outline

- perceived problems, proposed solutions
  - bufferbloat
  - the data center interconnect
- the case for fair flow queuing
  - traffic at flow level and scalability
  - fairness is all we need in the network
  - something else in the last/first mile

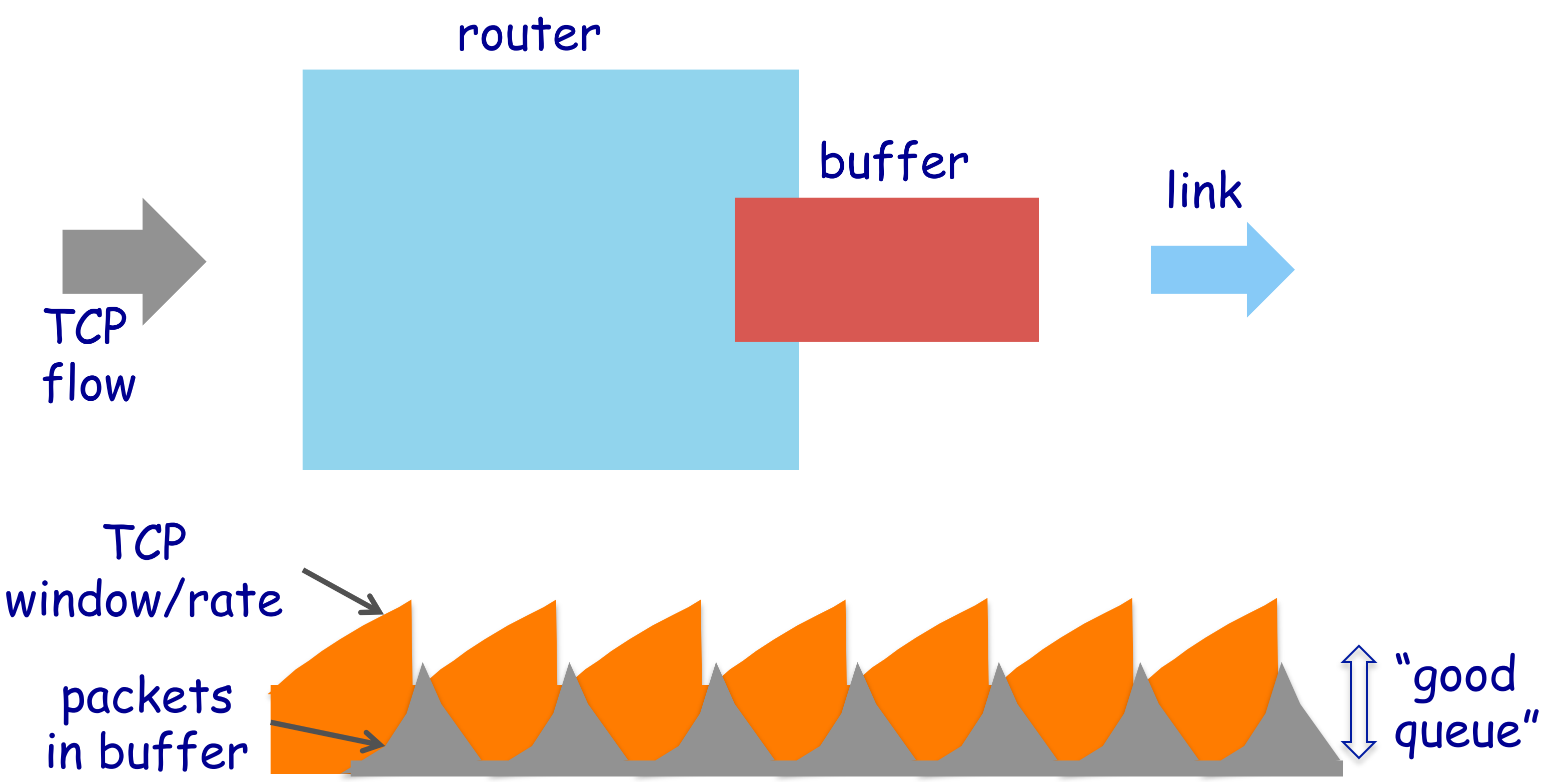
“Bufferbloat”

- the problem: too large buffers, notably in home routers, get filled by TCP leading to excessive packet latency

Bloat, the puffer-fish
© Pixar

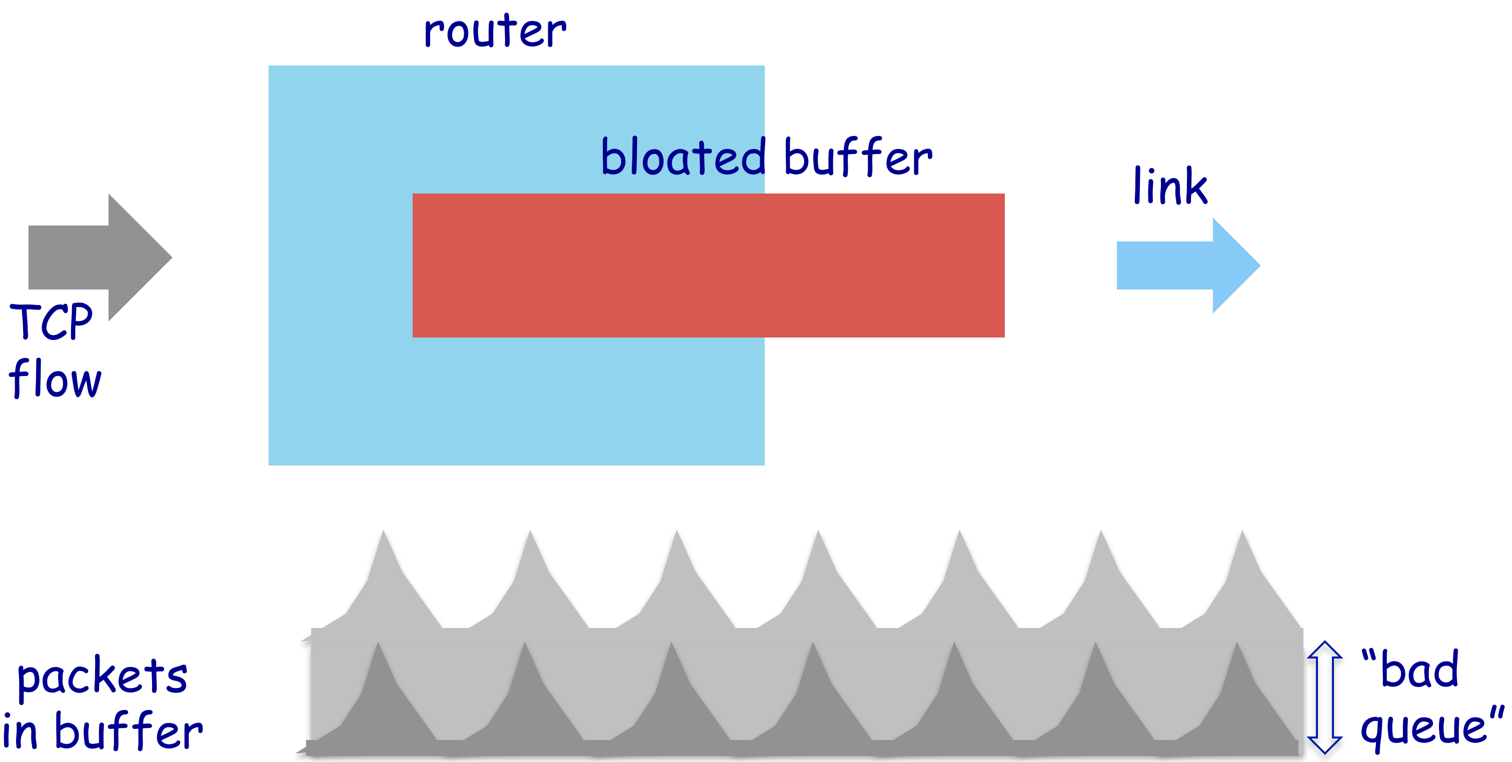
Bufferbloat

- how TCP should use the buffer



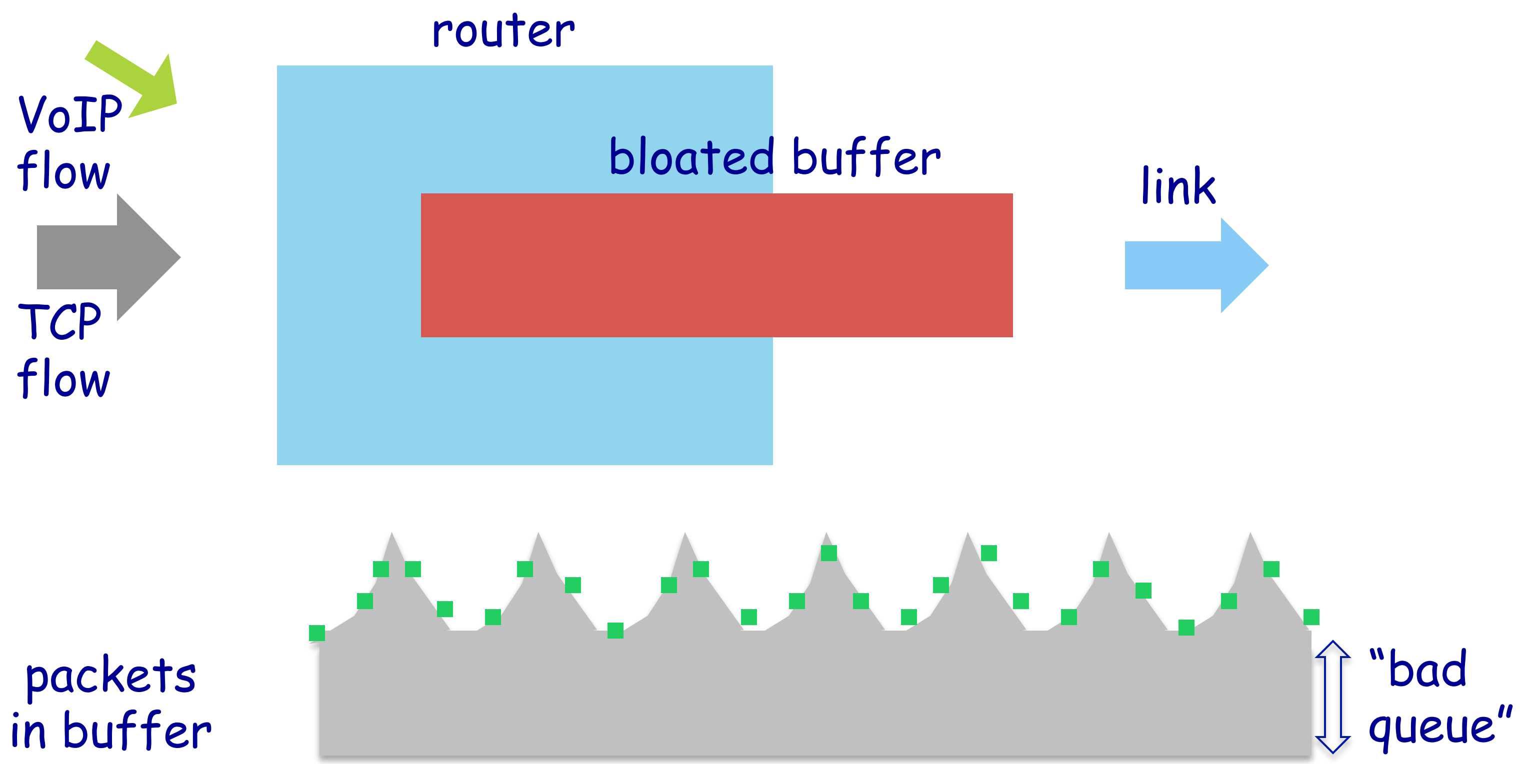
Bufferbloat

- impact of a bloated buffer: longer delays, same throughput



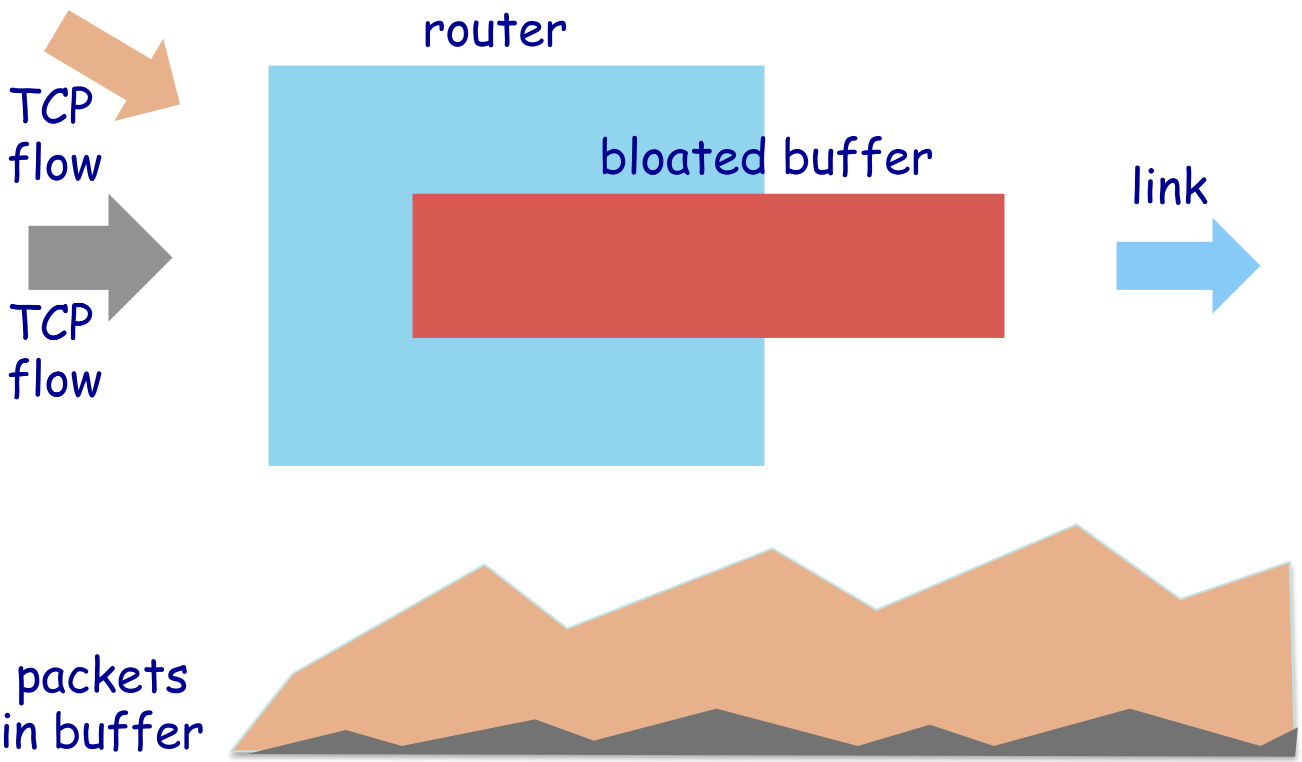
Bufferbloat

- impact of a bloated buffer: high latency for real time flows



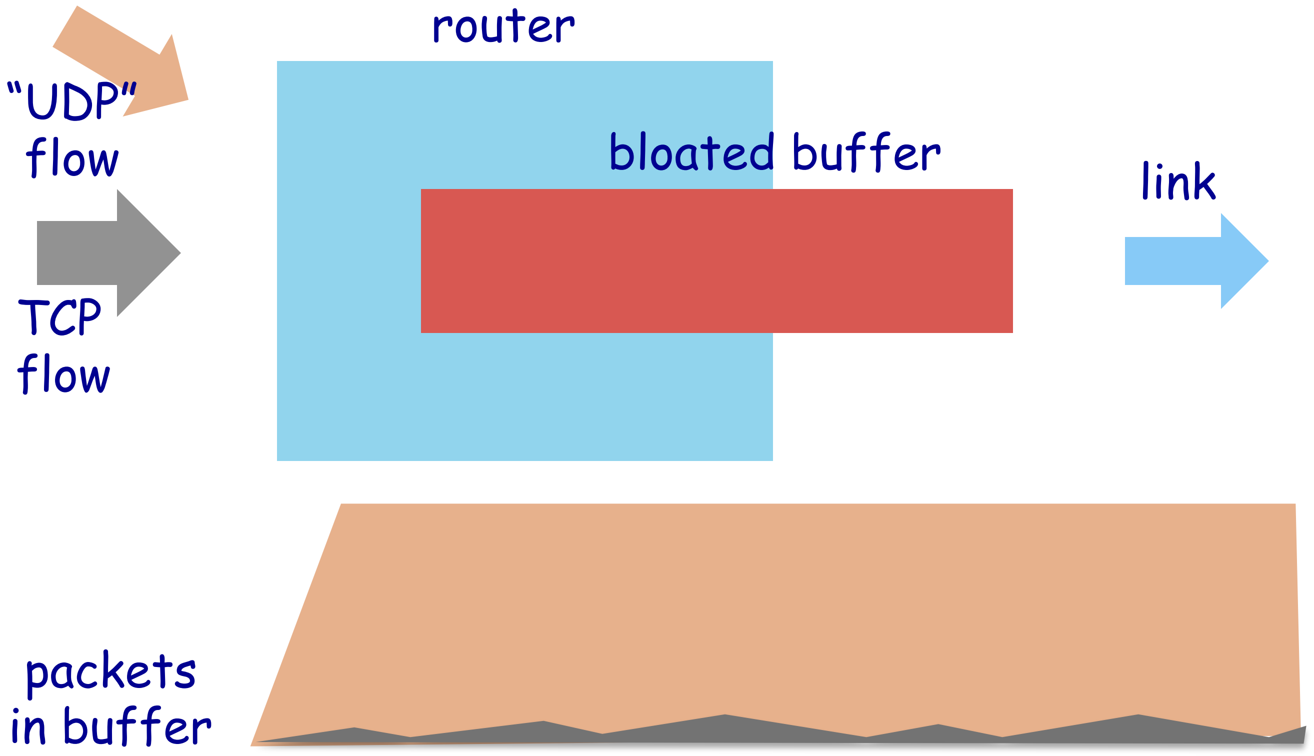
Bufferbloat

- impact of drop tail: unfair bandwidth sharing



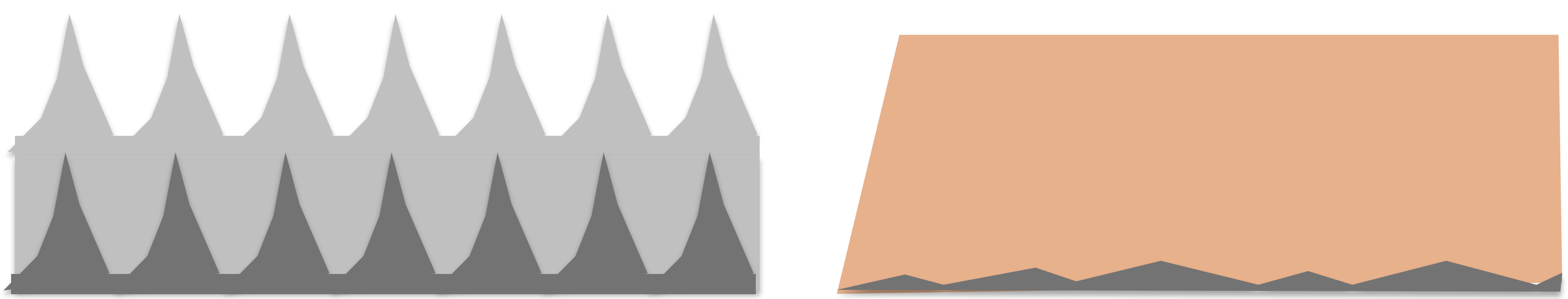
Bufferbloat

- impact of drop tail: unfair bandwidth sharing



AQM to combat bufferbloat

- CoDel (Controlled Delay) [Nichols & Jacobson, 2012]
  - measure packet sojourn time
  - drop packets to keep minimum delay near target
- PIE (Proportional Integral controller Enhanced) [Pan et al, 2013]
  - drop probability updated based on queue length & departure rate
- both rely on TCP in end-system, neither ensures fairness

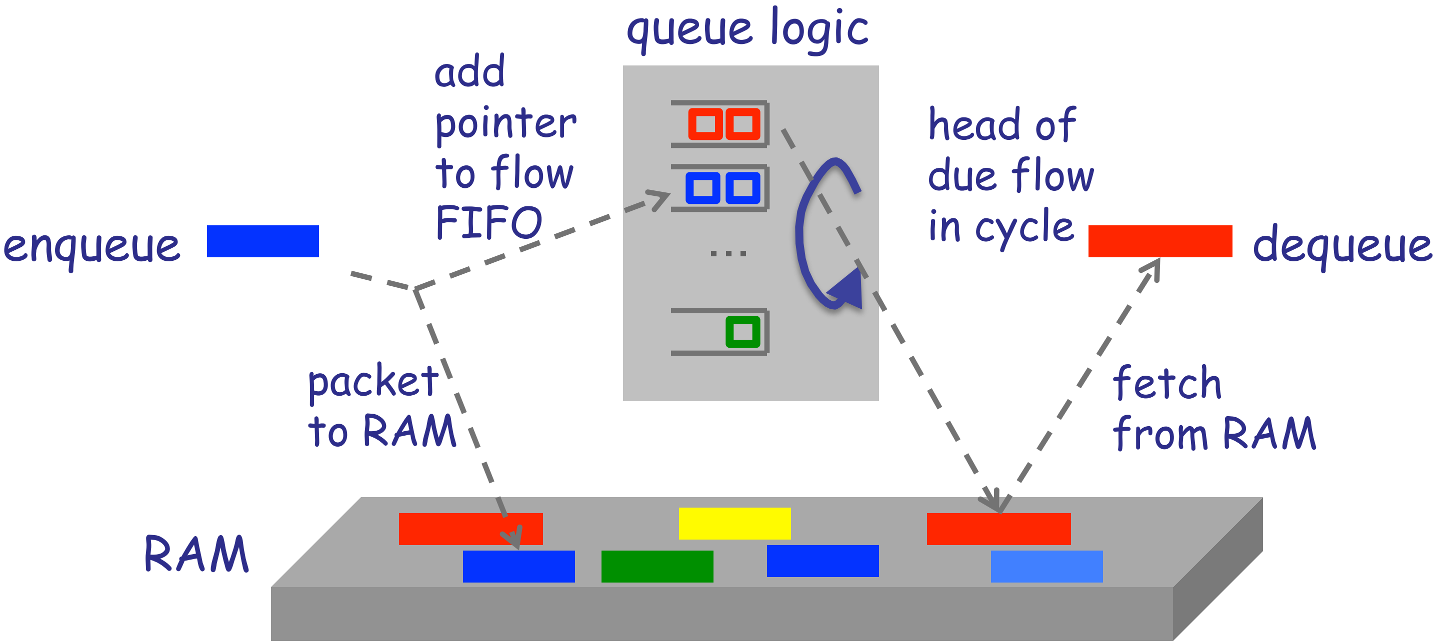


AQM and scheduling: fq_codel

- fq_codel combines SFQ (stochastic fairness queuing) and CoDel
  - hash flow ID to one of ~1000 queues (buckets)
  - deficit round robin scheduling over queues
  - control latency in each queue using CoDel
- with some enhancements
  - priority to packets of "new" flows
  - drop from queue head rather than tail
- V. Jacobson (quoted by D. Täht):
  - "If we're sticking code into boxes to deploy CoDel, don't do that. Deploy fq_codel. It's just an across the board win"

Realizing fair queuing

- a shared pool of RAM
- enqueue and dequeue logic implementing deficit round robin (DRR)
- complexity and performance depend on number of active flows (flows that have 1 or more packets in buffer)

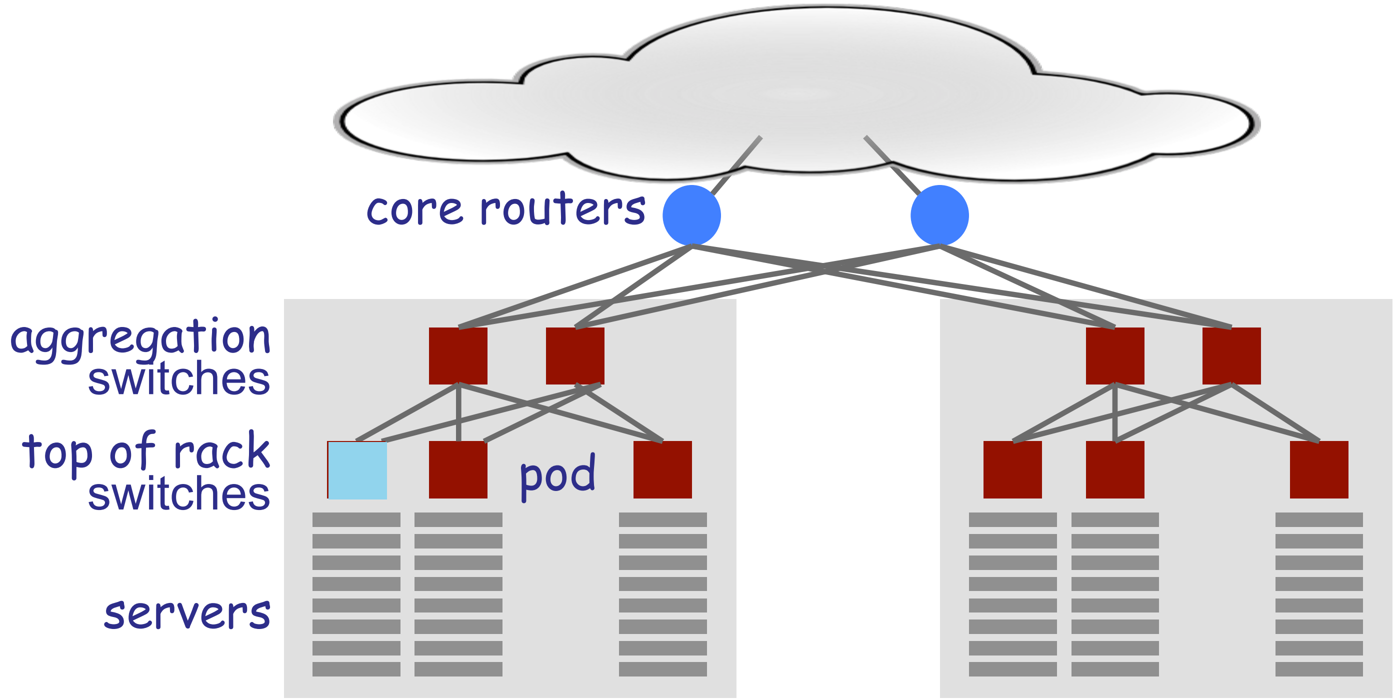


Outline

- perceived problems, proposed solutions
  - bufferbloat
  - the data center interconnect
- the case for fair flow queuing
  - traffic at flow level and scalability
  - fairness is all we need in the network
  - something else in the last/first mile

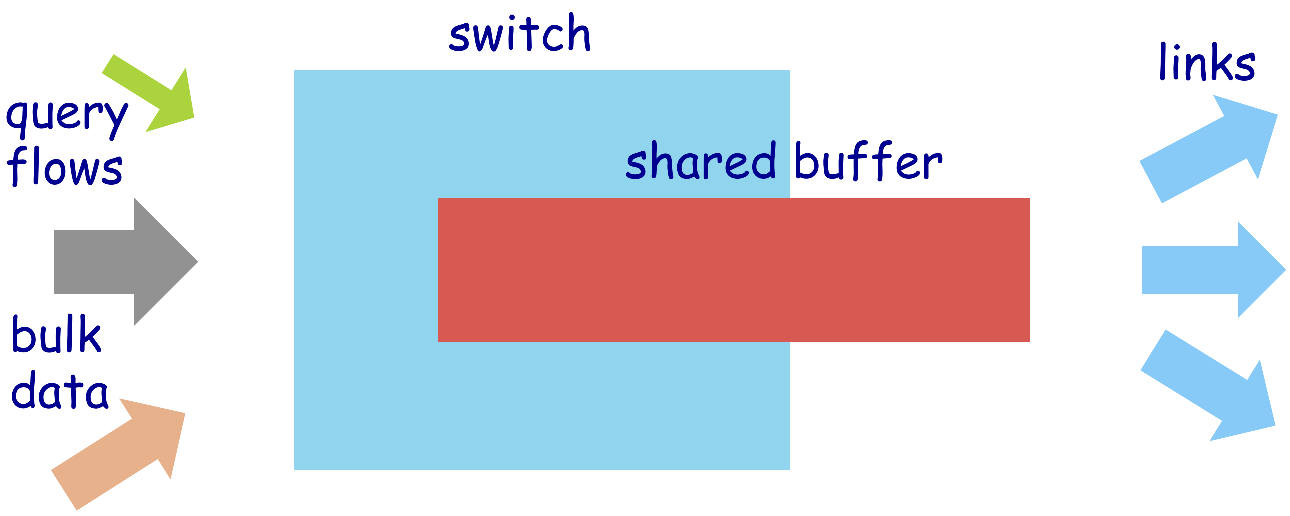
Congestion in the interconnect

- 1000s of servers connected by commodity switches and routers
- mixture of bulk data transfers requiring high throughput...
- ... and query flows requiring low latency



Congestion in the interconnect

- 1000s of servers connected by commodity switches and routers
- mixture of bulk data transfers requiring high throughput...
- ... and query flows requiring low latency
- a practical observation
  - regular TCP does not ensure high throughput and low latency

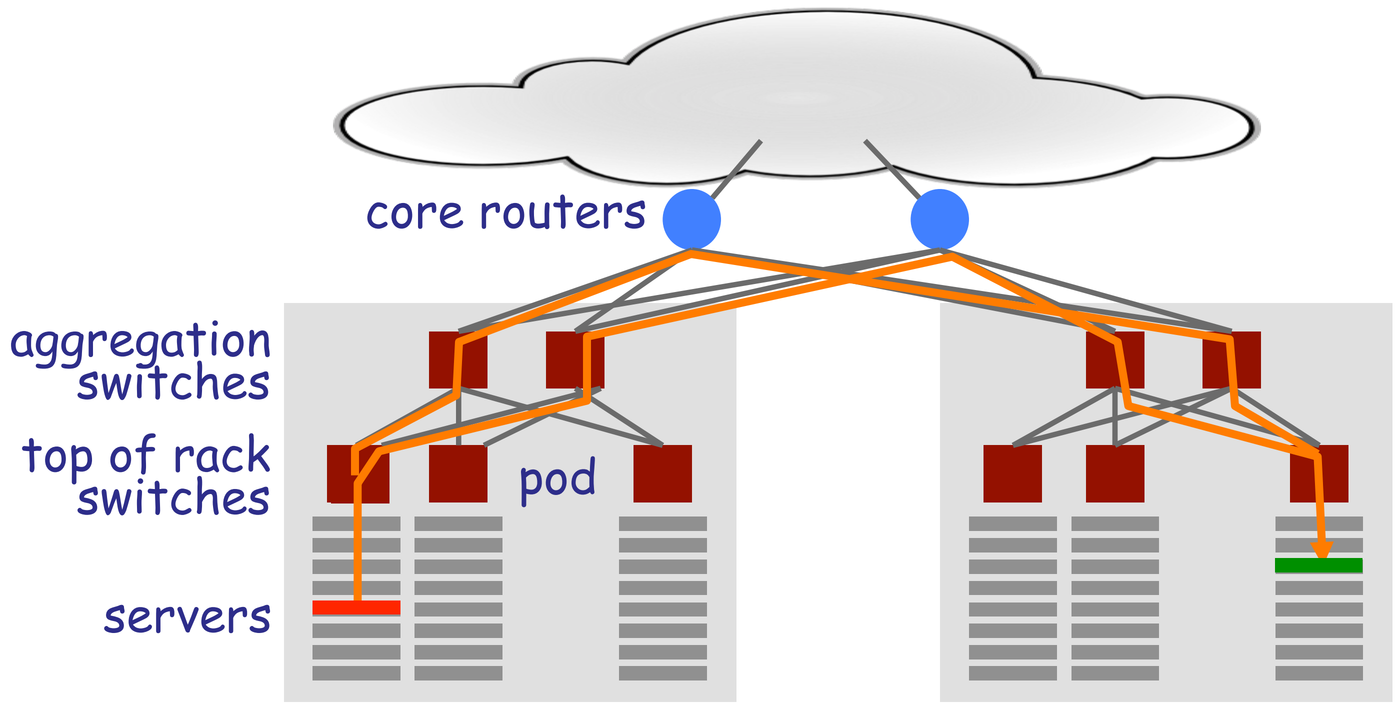


Many new congestion control protocols

- DCTCP [Alizadeh 2010]
  - limits delays by refined ECN scheme to smooth rate variations
- D^3 [Wilson 2011]
  - "deadline driven delivery"
- D^2TCP [Vamanan 2012]
  - combines aspects of previous two
- PDQ [Hong 2012]
  - size-based pre-emptive scheduling
- HULL [Alizadeh 2012]
  - low delay by "phantom queues" and ECN
- evaluations assume all data center flows implement the recommended protocol

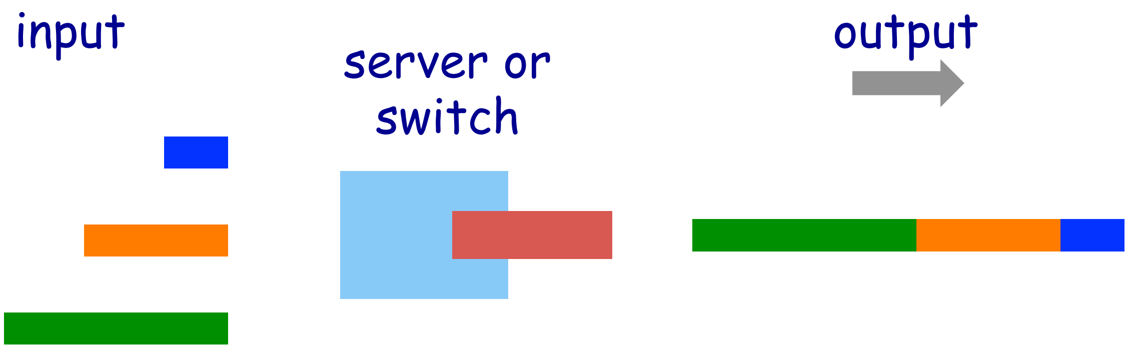
pFabric: "minimalist data center transport"

- instead of end-to-end congestion control, implement scheduling in switch and server buffers [Alizadeh 2013]
- "key insight: decouple flow schedule from rate control"



pFabric: "minimalist data center transport"

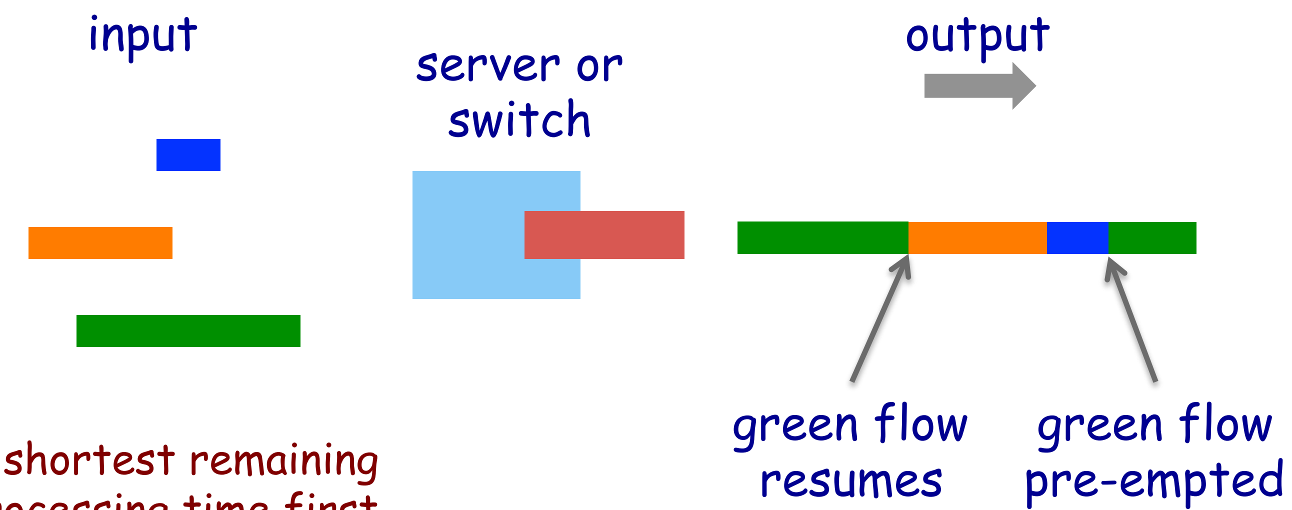
- instead of end-to-end congestion control, implement scheduling in switch and server buffers [Alizadeh 2013]
- "key insight: decouple flow schedule from rate control"
- SRPT* scheduling to minimize flow completion time



* SRPT = shortest remaining processing time first

pFabric: "minimalist data center transport"

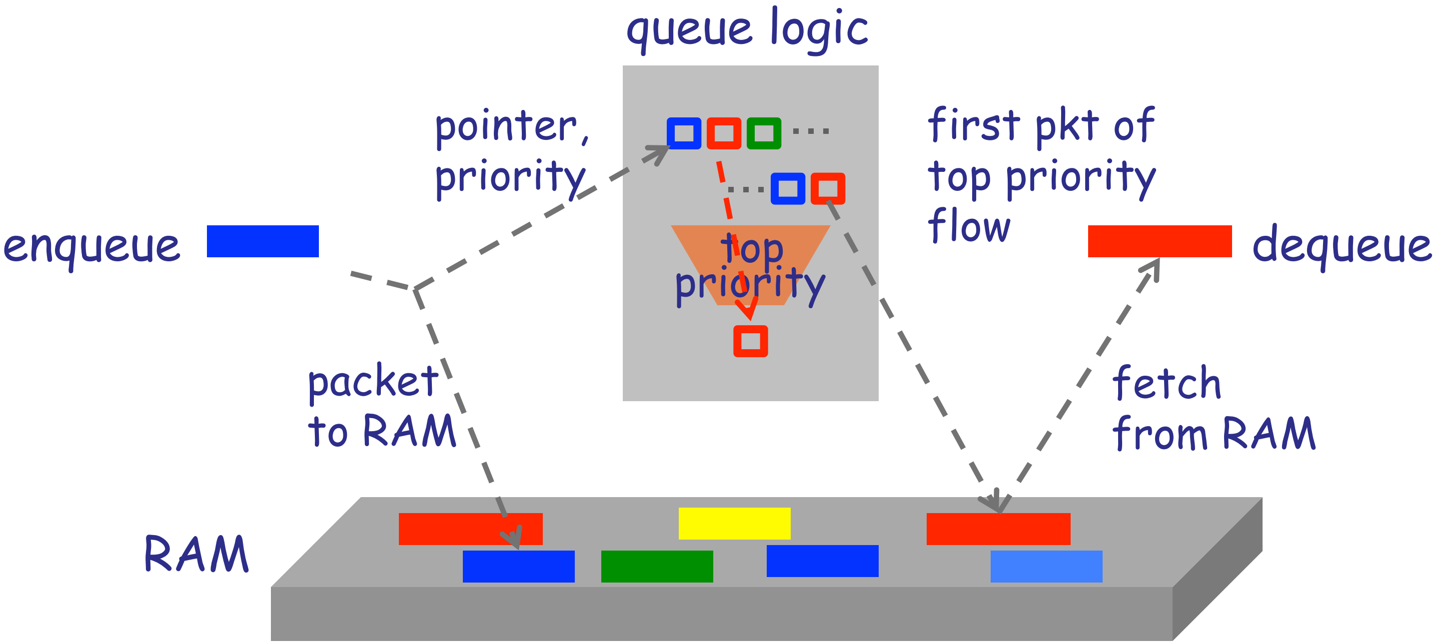
- instead of end-to-end congestion control, implement scheduling in switch and server buffers [Alizadeh 2013]
- "key insight: decouple flow schedule from rate control"
- SRPT* scheduling to minimize flow completion time
- also optimal for flows that arrive over time
- minimal rate control: eg, start at max rate, adjust using AIMD



* SRPT = shortest remaining processing time first

Realizing SRPT queuing

- a shared pool of RAM
- enqueue and dequeue logic implementing SRPT
  - priority ⇔ remaining flow size
- similar complexity to DRR

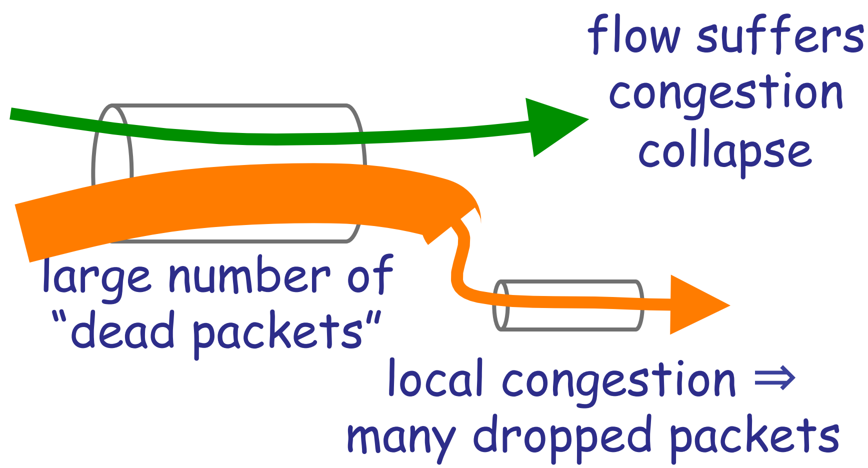


Outline

- perceived problems, proposed solutions
  - bufferbloat
  - the data center interconnect
- the case for fair flow queuing
  - traffic at flow level and scalability
  - fairness is all we need in the network
  - something else in the last/first mile

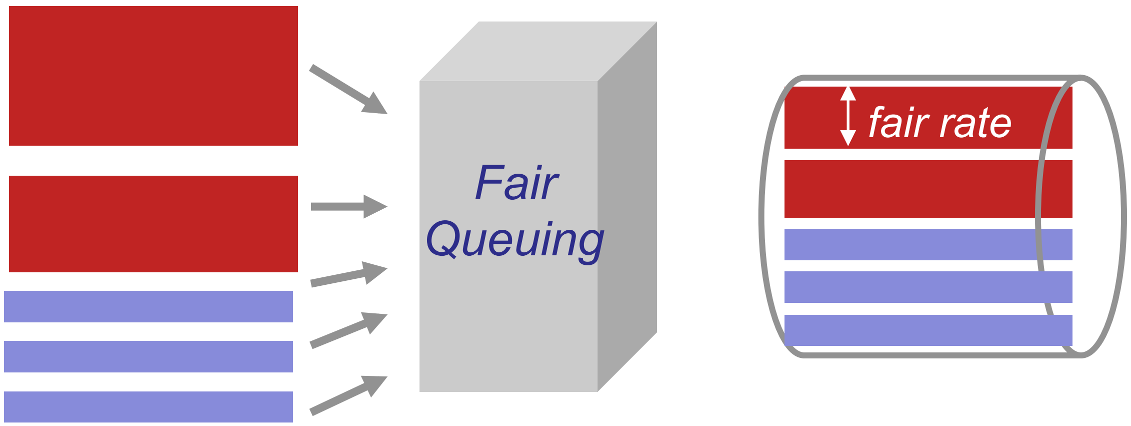
How can the Internet continue to rely on end-to-end congestion control ?

- "TCP saved the Internet from congestion collapse"
  - but there are easier ways to avoid the "dead packet" phenomenon
- new TCP versions are necessary for high speed links
  - but they are generally very unfair to legacy versions
- no incentive for applications to be "TCP friendly"
  - in particular, congestion pricing is unworkable
- better, like pFabric, to decouple scheduling and rate control



Decouple scheduling and rate control through per-flow fair queuing

- imposed fairness means no need to rely on TCP
  - flows use the congestion control they want (eg, high speed TCP)
  - no danger from “unresponsive flows”
- fairness realizes implicit service differentiation
  - since rate of streaming flows is generally less than fair rate
  - lower still latency by giving priority to “new” flows
- as proposed by [Nagle 1985], [Kumar 1998], [Kortebi 2004],...
  - with longest queue drop as AQM

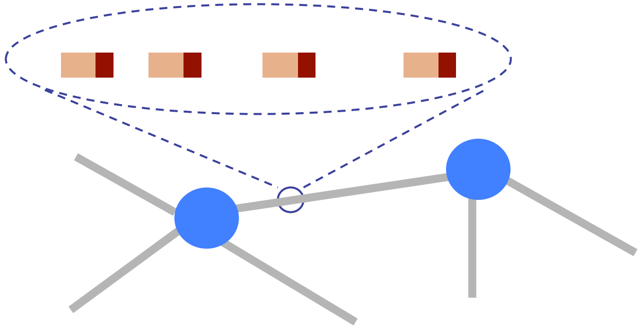


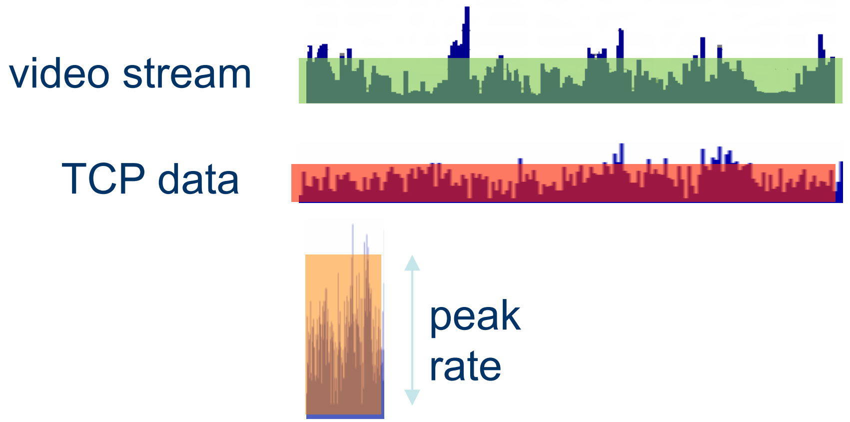
Outline

- perceived problems, proposed solutions
  - bufferbloat
  - the data center interconnect
- the case for fair flow queuing
  - traffic at flow level and scalability
  - fairness is all we need in the network
  - something else in the last/first mile

Understanding traffic at flow level

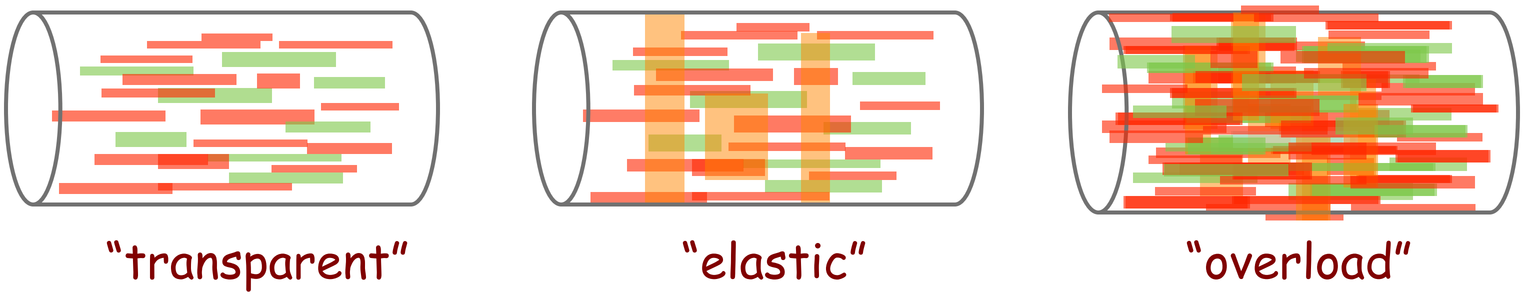
- rather than modelling traffic at packet level, recognize that packets belong to a flow (file transfer, voice signal,...)
  - a set of packets with like header fields, local in space and time
- traffic is a process of flows of different types
  - conversational, streaming, interactive data, background
  - with different traffic characteristics – rates, volumes,...
  - and different requirements for latency, integrity, throughput
- characterized by size and "peak rate"





Sharing regimes and the meaning of congestion

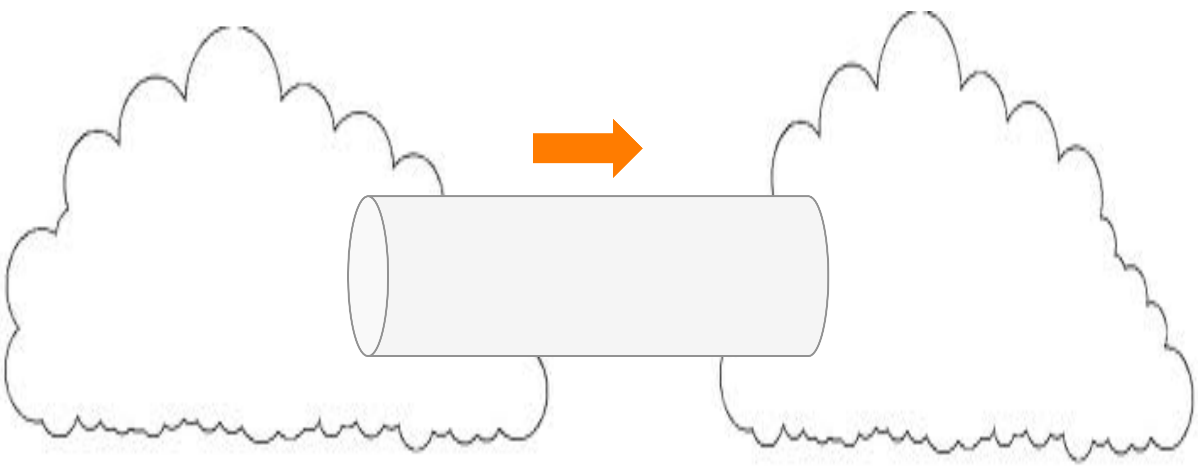
- three bandwidth sharing regimes



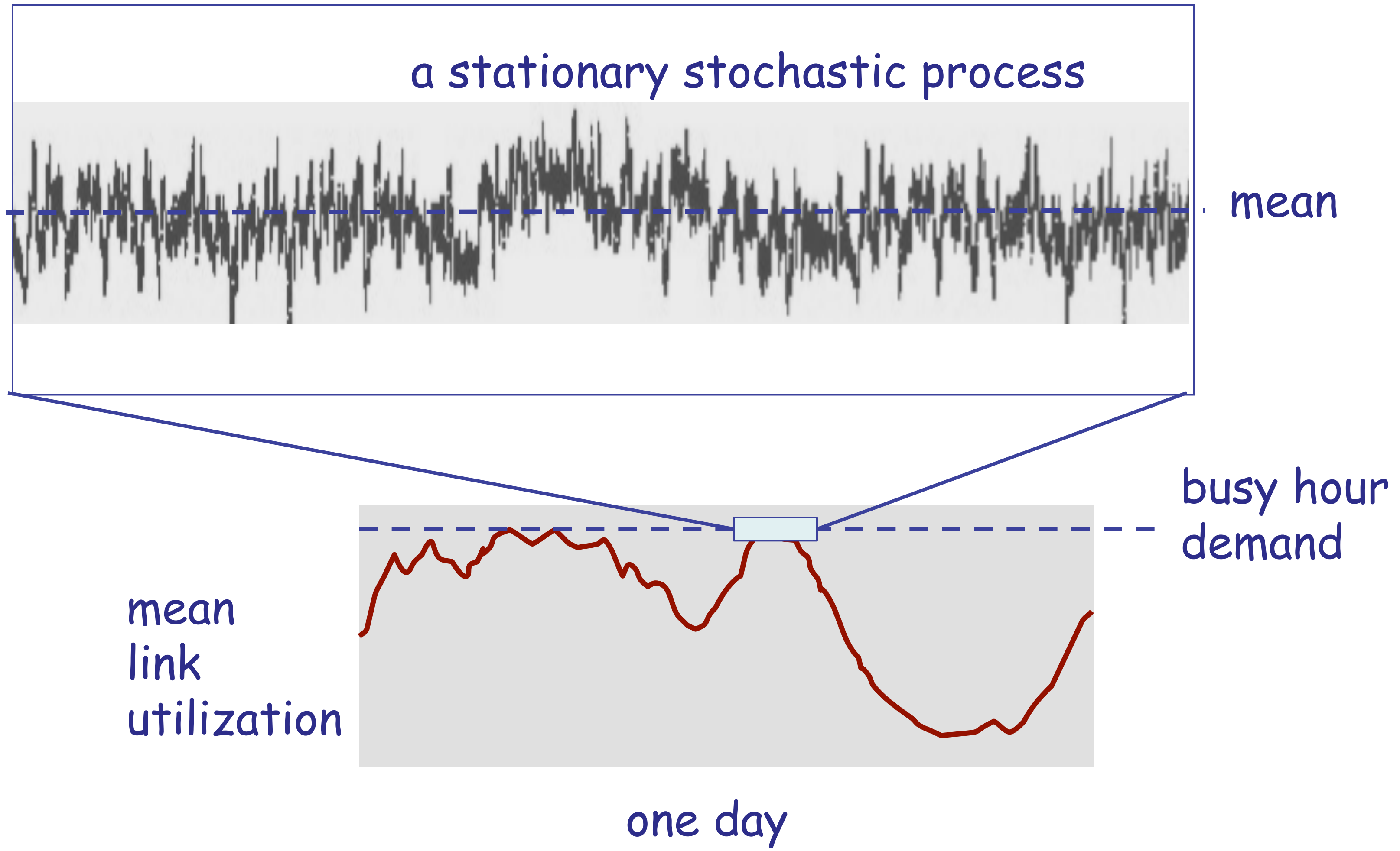
- transparent regime:
  - all flows suffer negligible loss, no throughput degradation
- elastic regime:
  - some high rate flows can saturate residual bandwidth; without control these can degrade quality for all other flows
- overload regime:
  - traffic load is greater than capacity; all flows suffer from congestion unless they are handled with priority

Statistical bandwidth sharing

- ie, statistical multiplexing with elastic traffic
- consider a network link handling flows between users, servers, data centers,...
- define, link load = flow arrival rate x mean flow size / link rate
  = packet arrival rate x mean packet size / link rate
  = mean link utilization



Traffic variations and stationarity

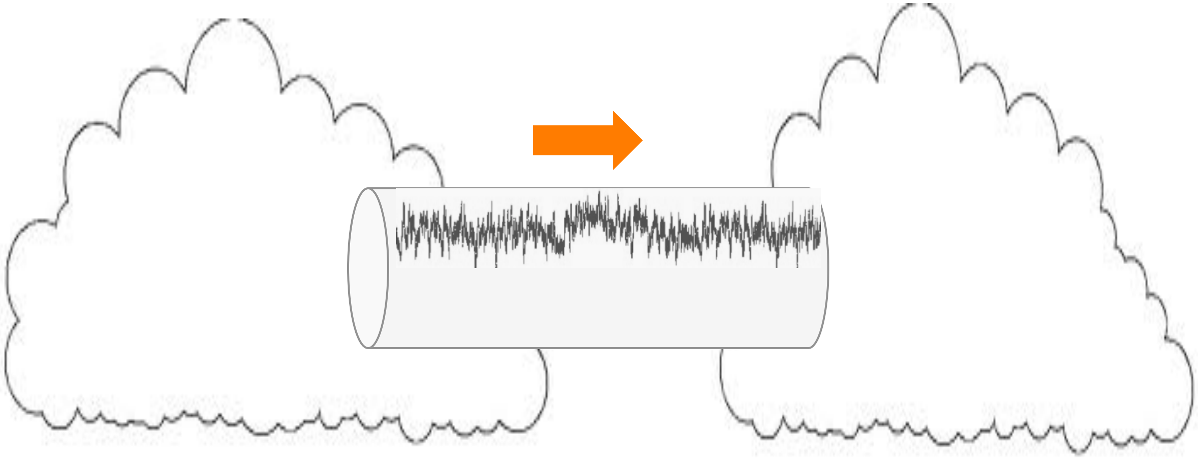


Statistical bandwidth sharing

- ie, statistical multiplexing with elastic traffic
- consider a network link handling flows between users, servers, data centers,...
- define, link load = flow arrival rate x mean flow size / link rate

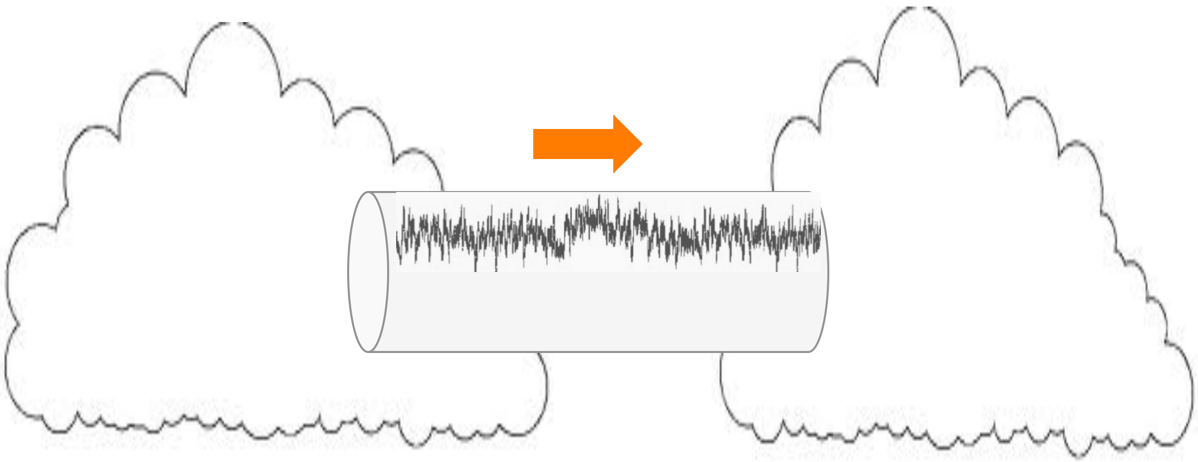
  = packet arrival rate x mean packet size / link rate

  = mean link utilization

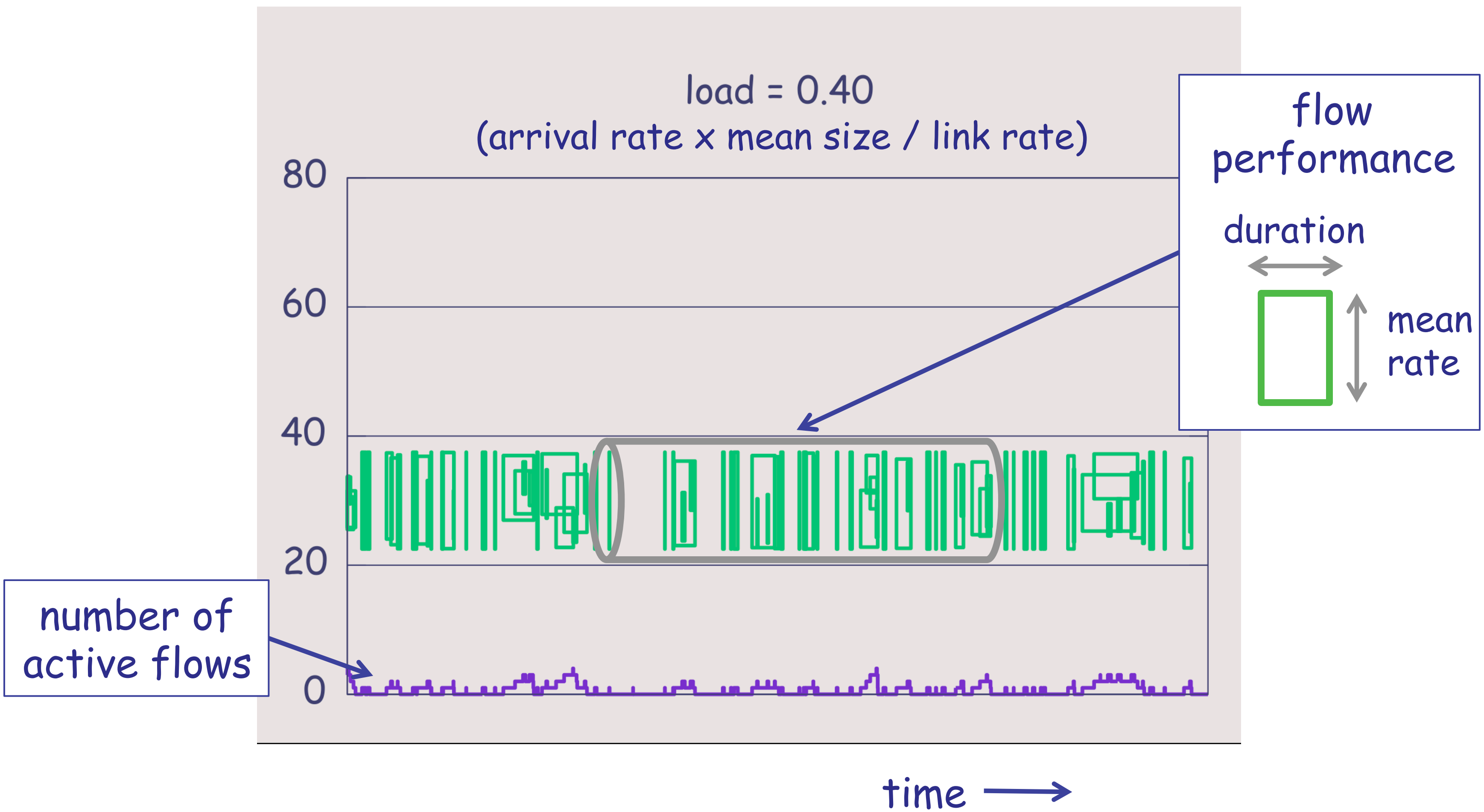


Bandwidth sharing performance

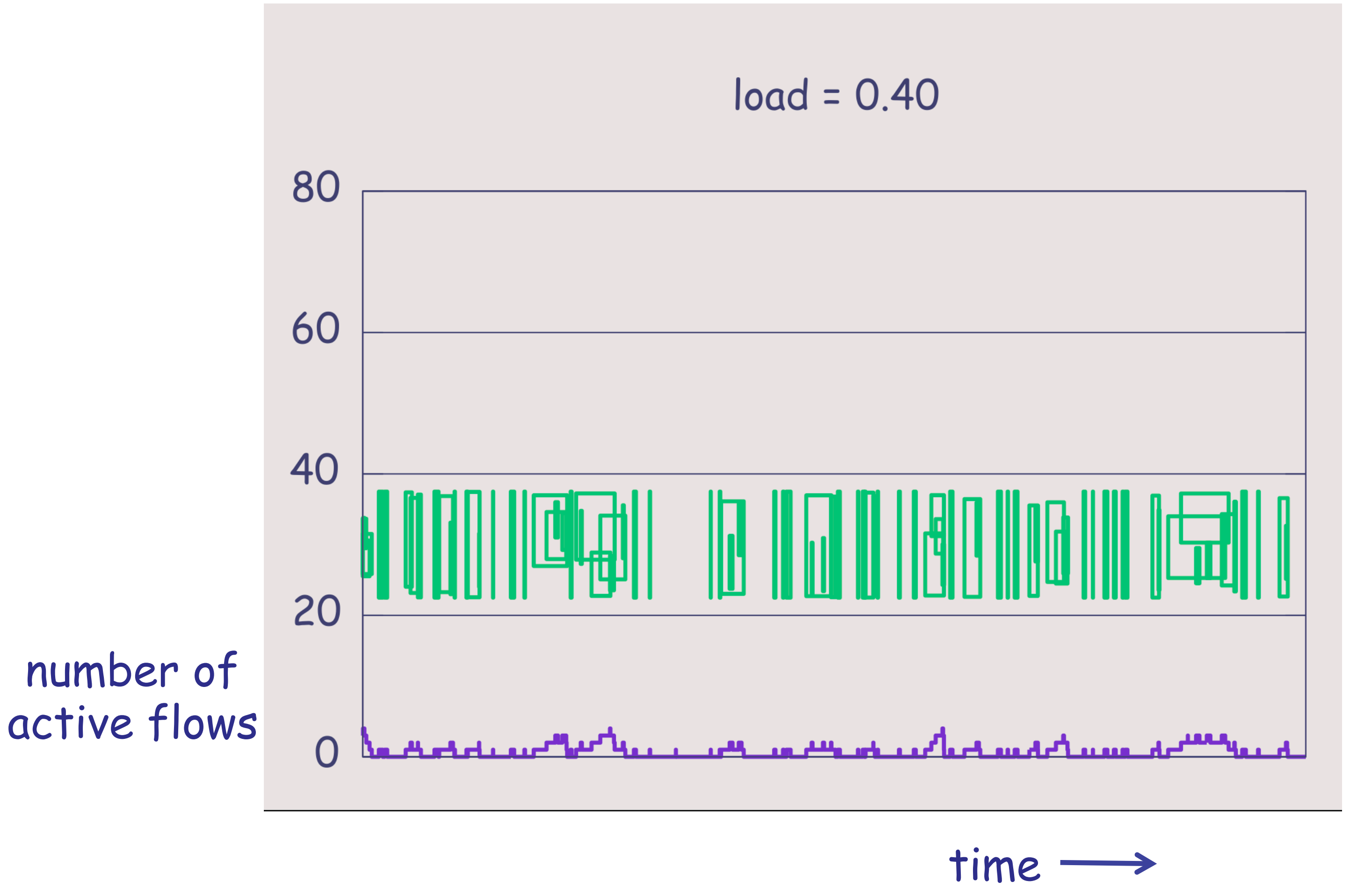
- in the following simulation experiments, assume flows
  - arrive as a Poisson process
  - have exponential size distribution
  - instantaneously share link bandwidth fairly
- results apply more generally thanks to insensitivity



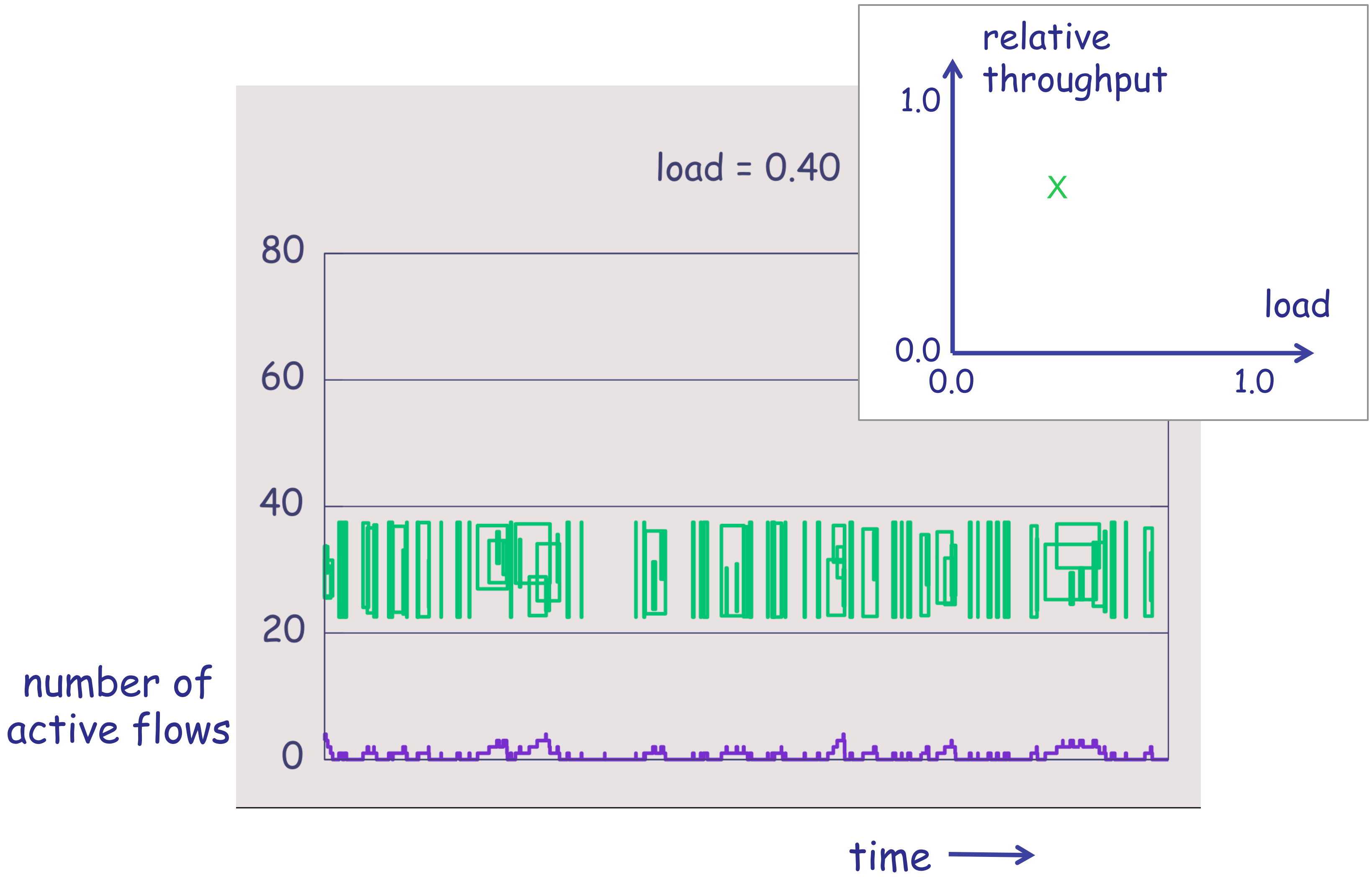
Performance of fair shared link



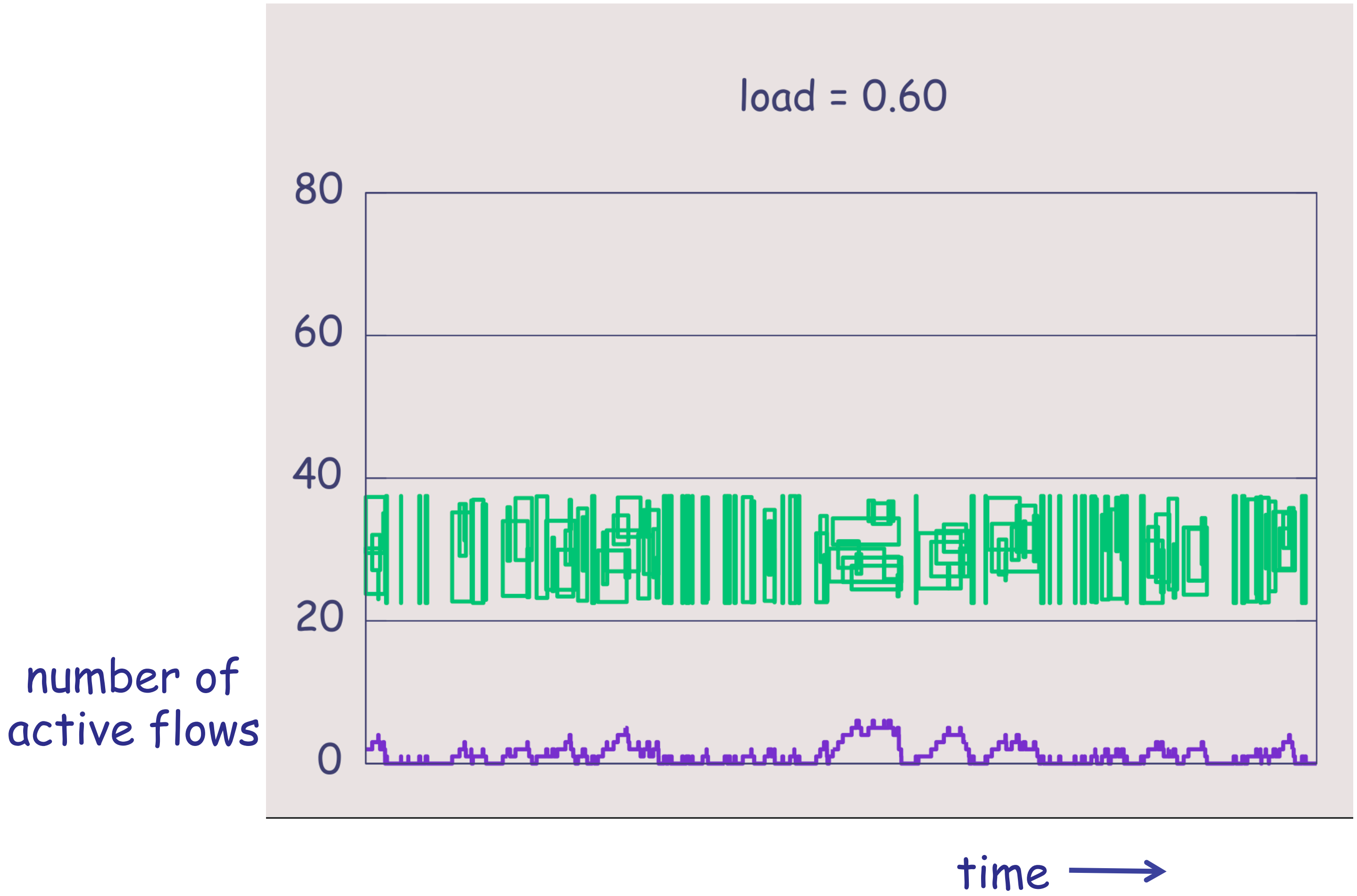
Performance of fair shared link



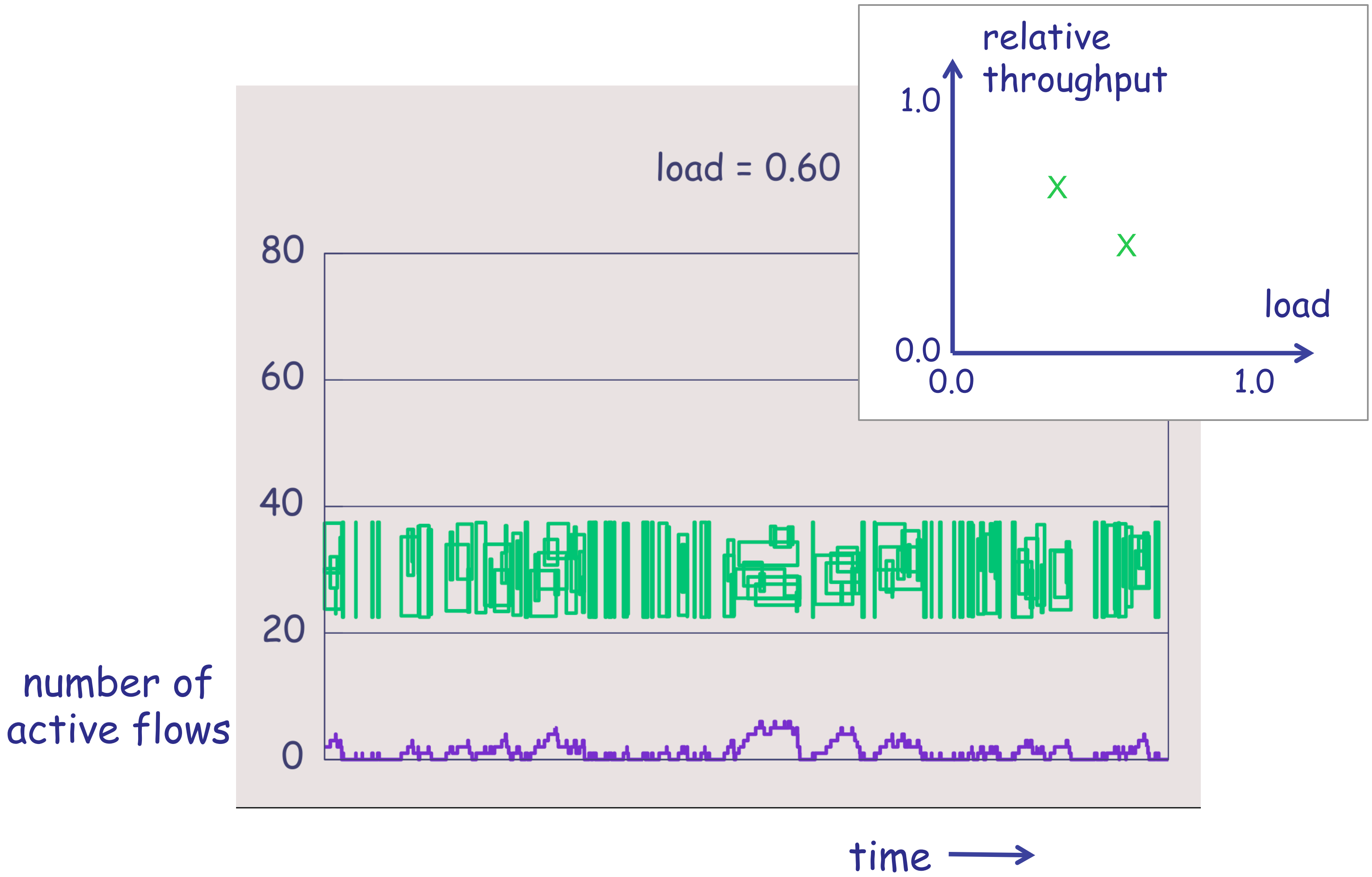
Performance of fair shared link



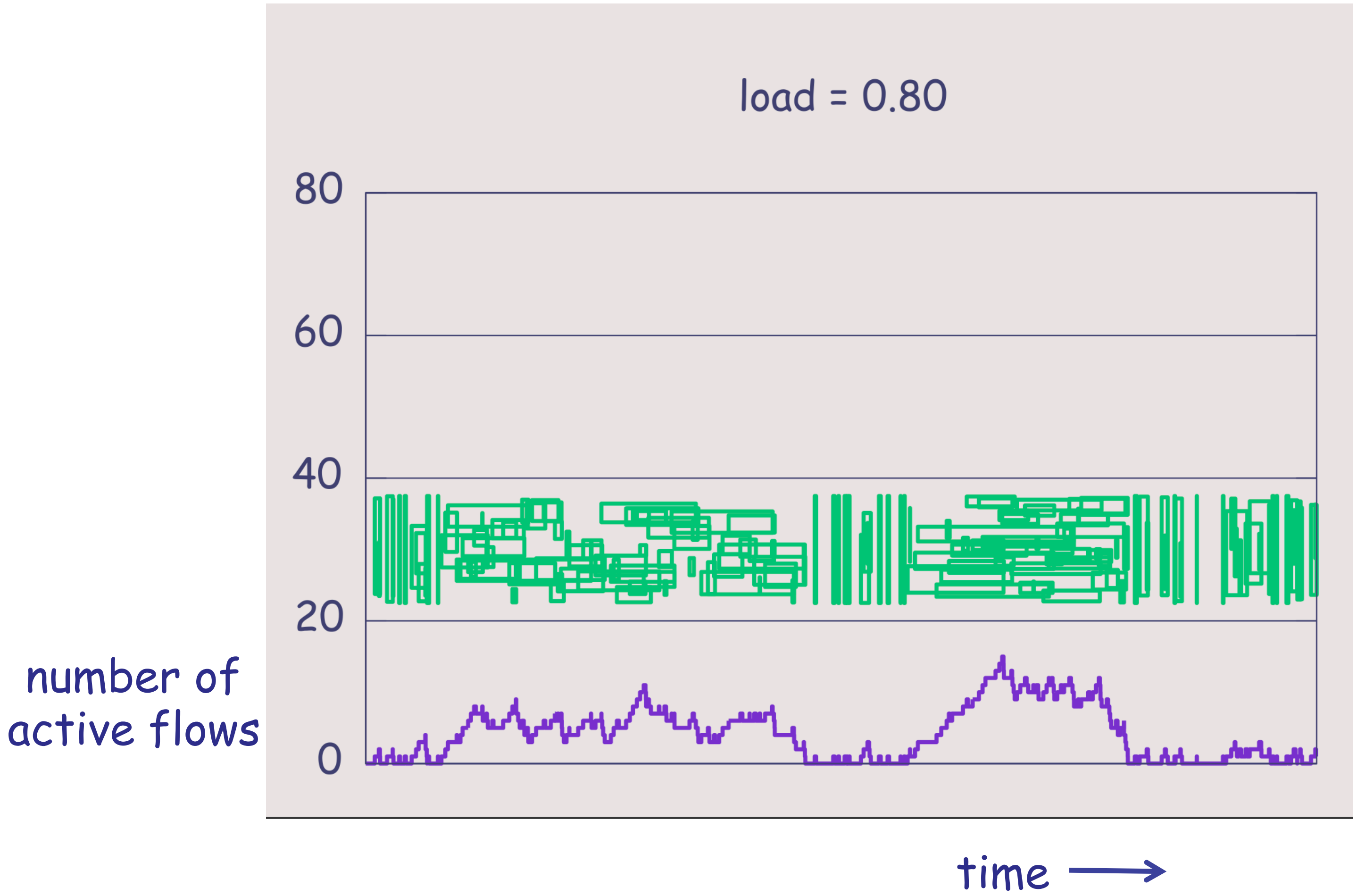
Performance of fair shared link



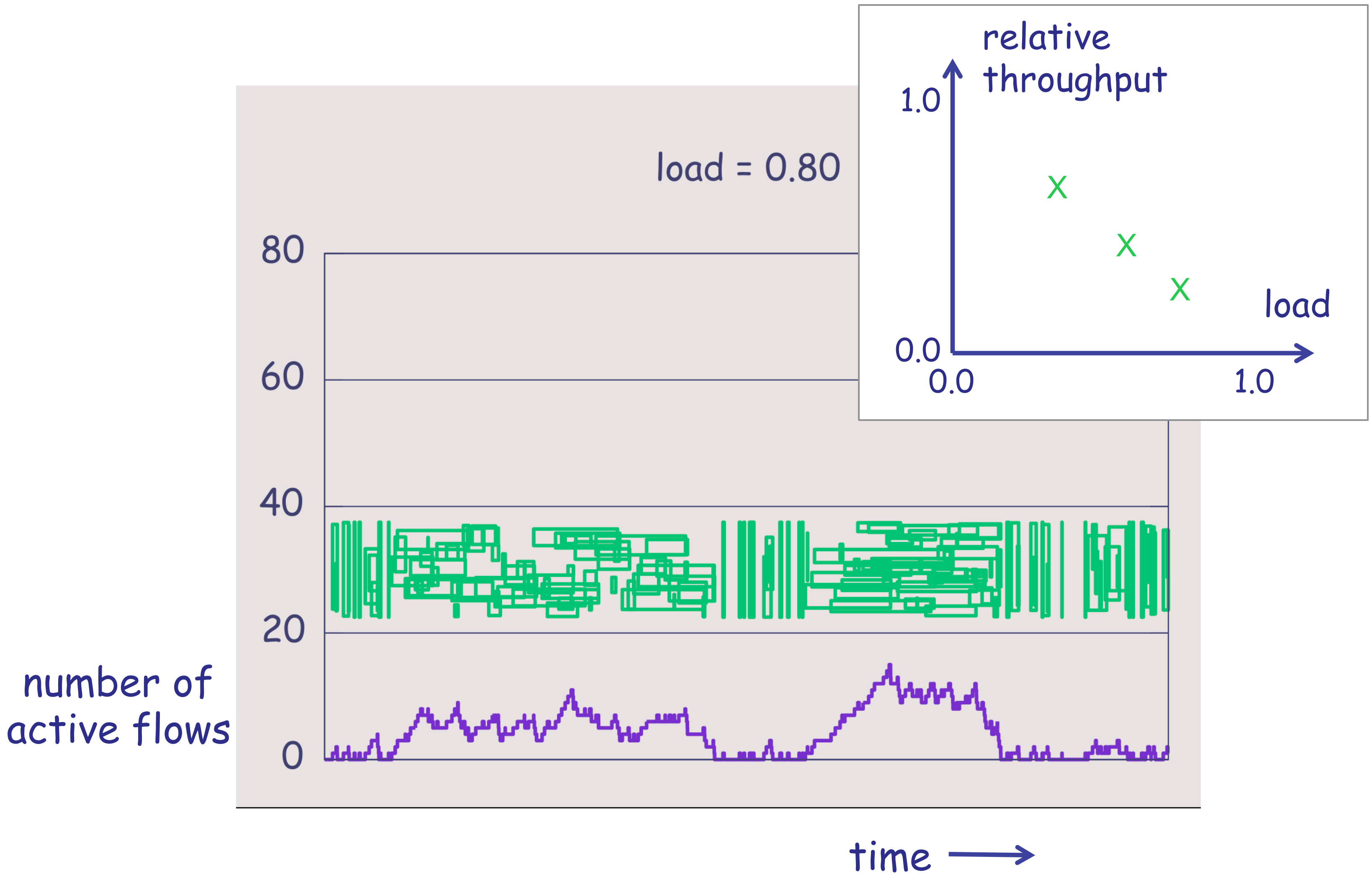
Performance of fair shared link



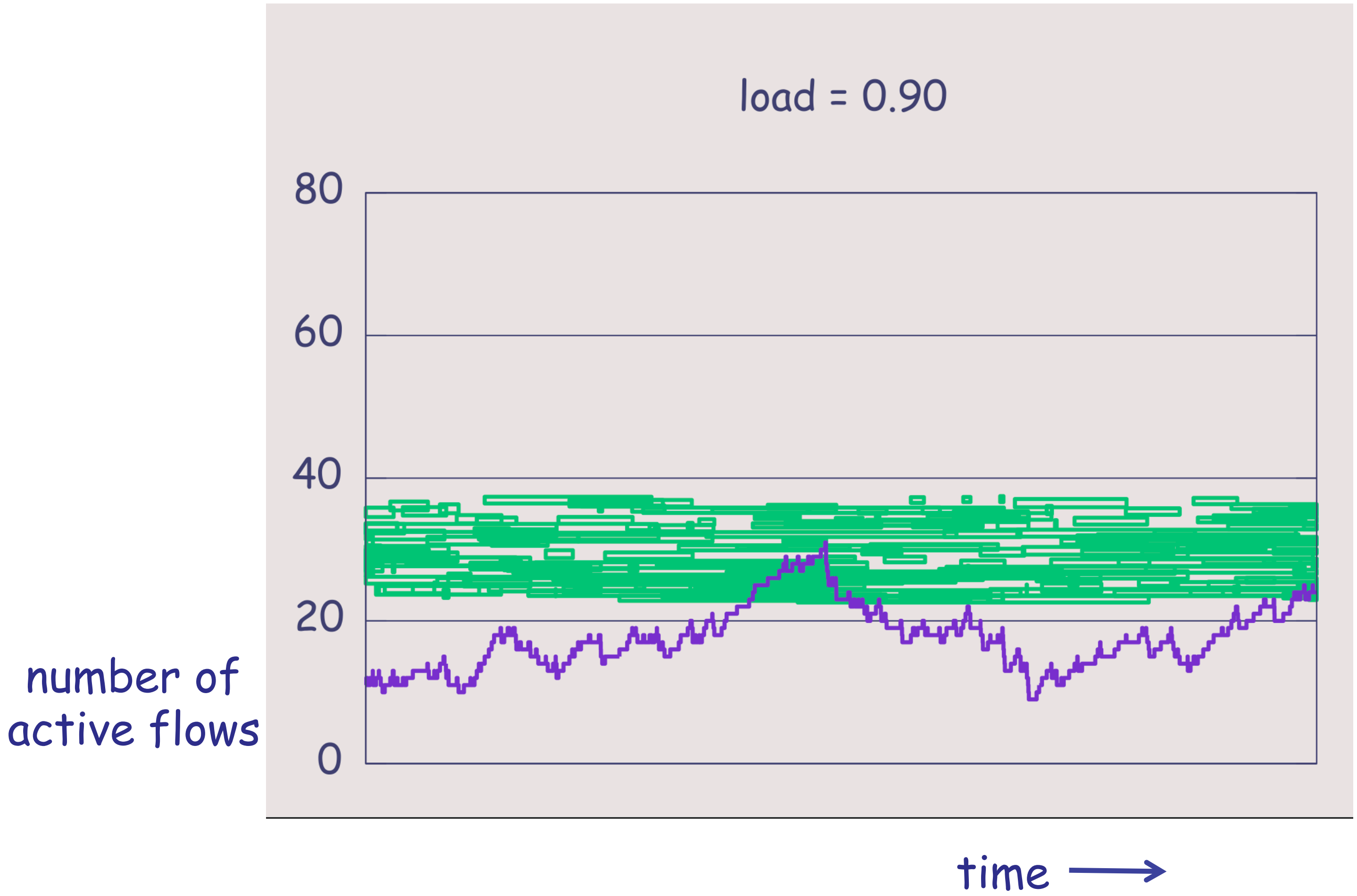
Performance of fair shared link



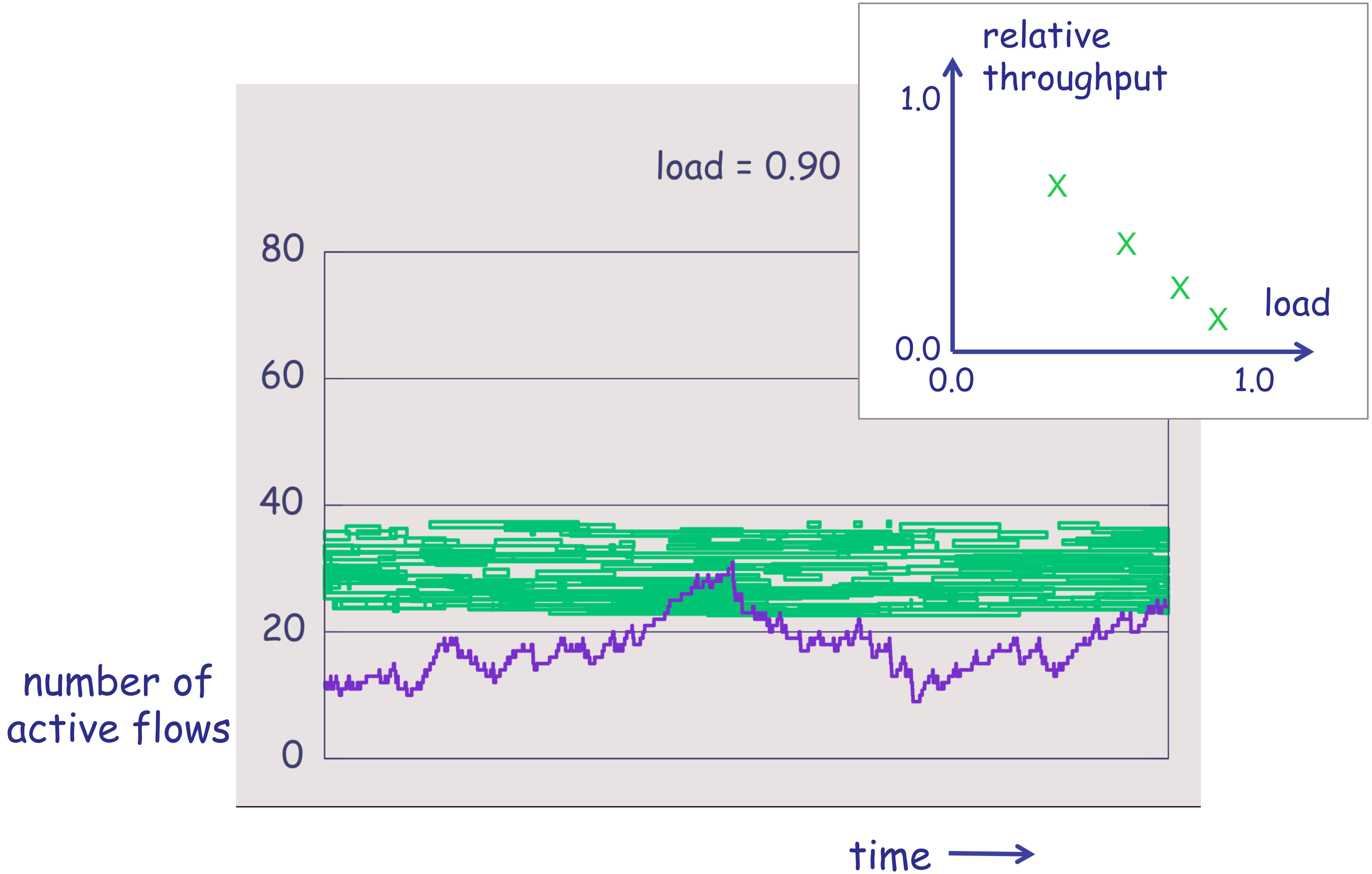
Performance of fair shared link



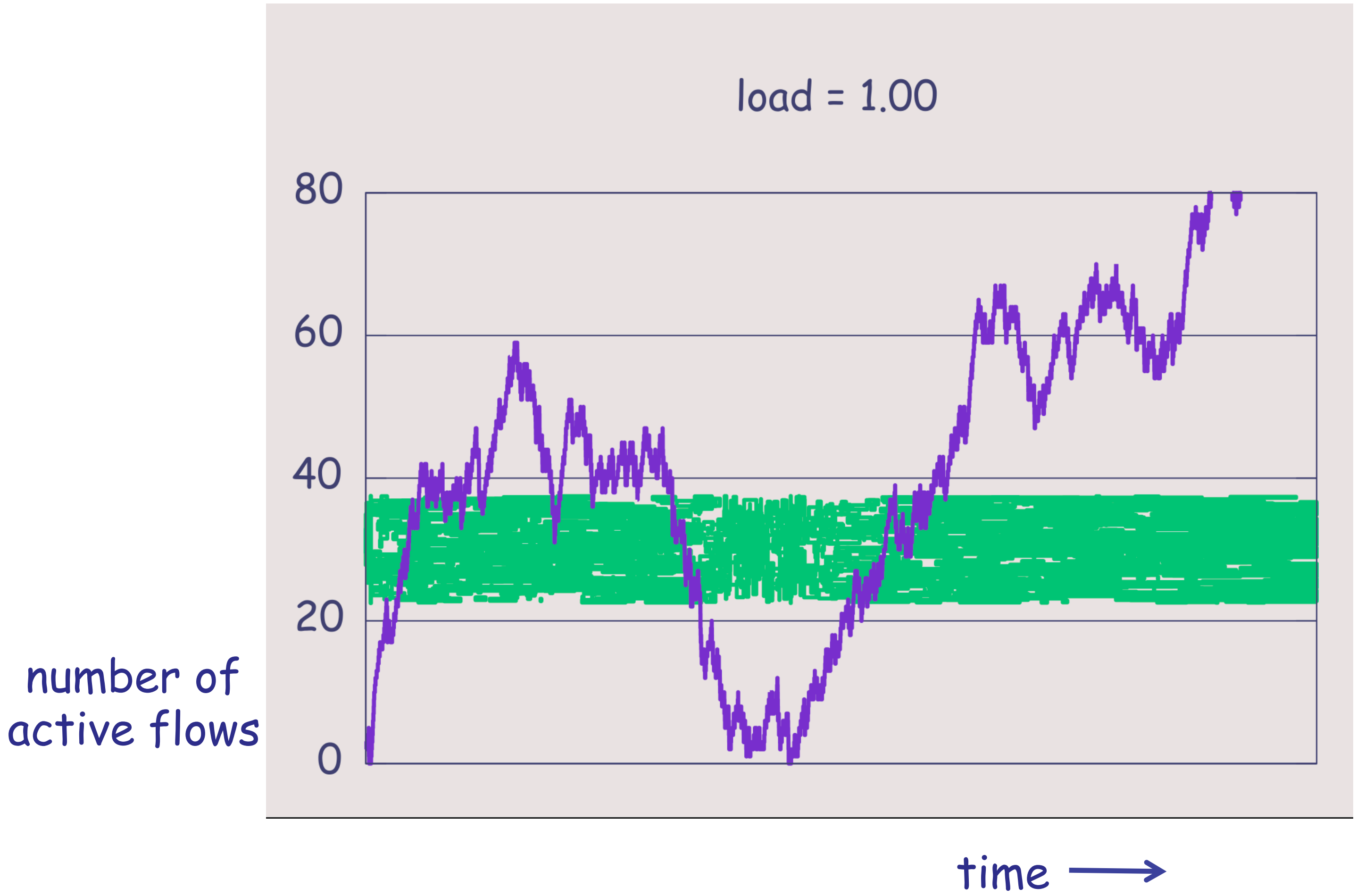
Performance of fair shared link



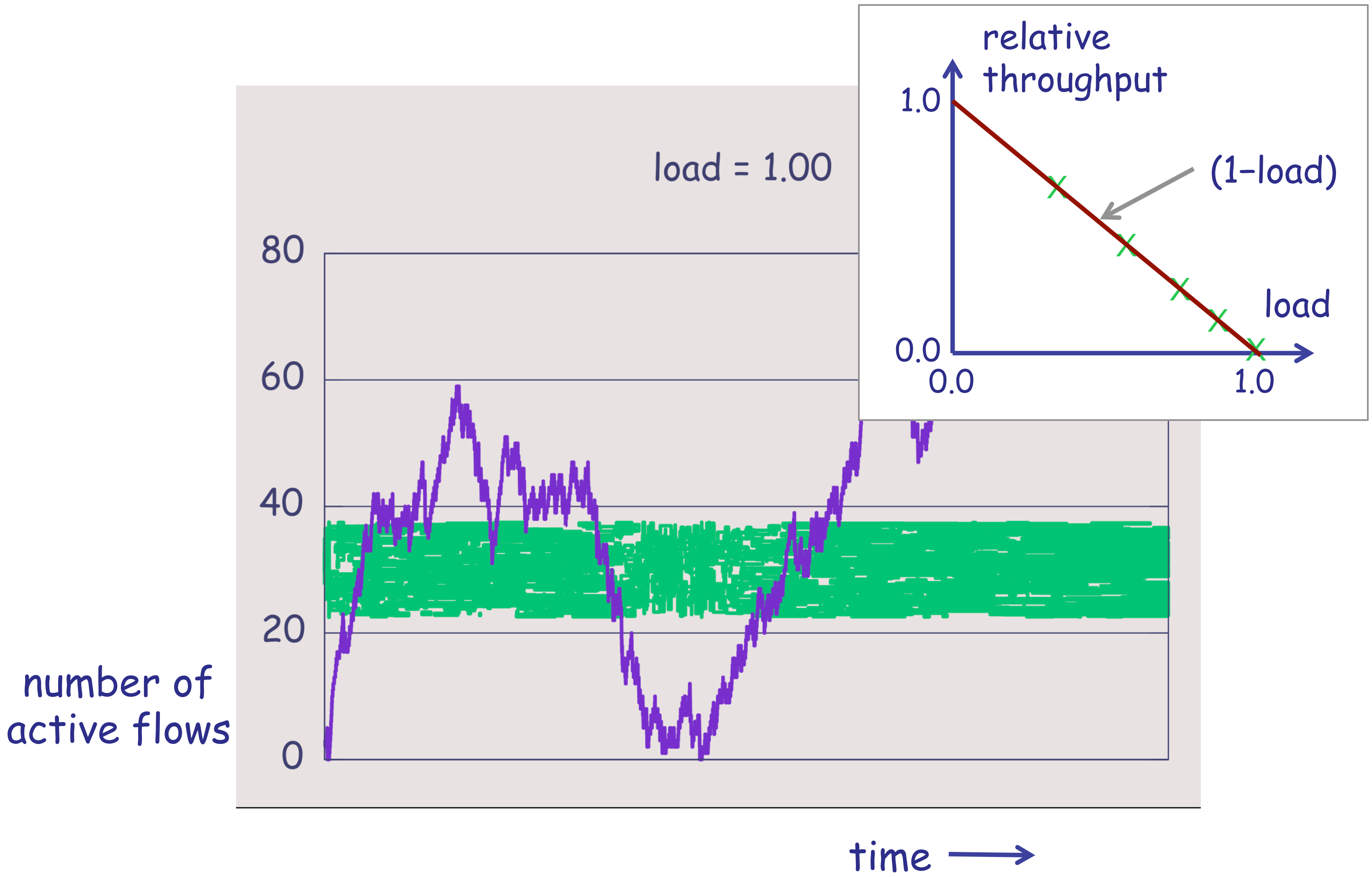
Performance of fair shared link



Performance of fair shared link

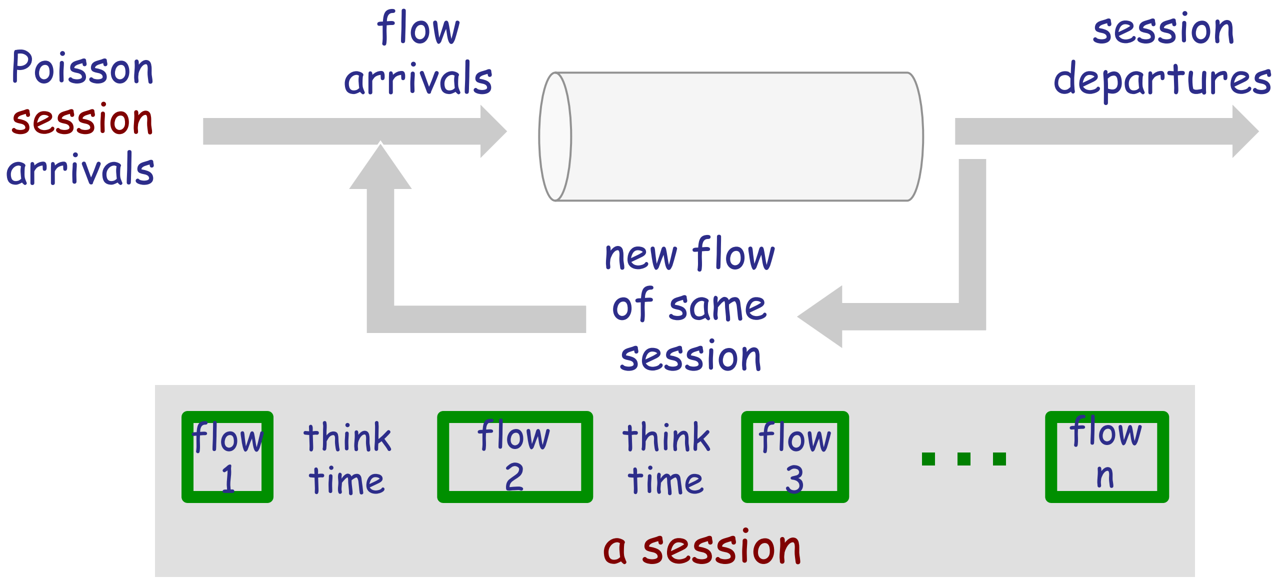


Performance of fair shared link



Observations

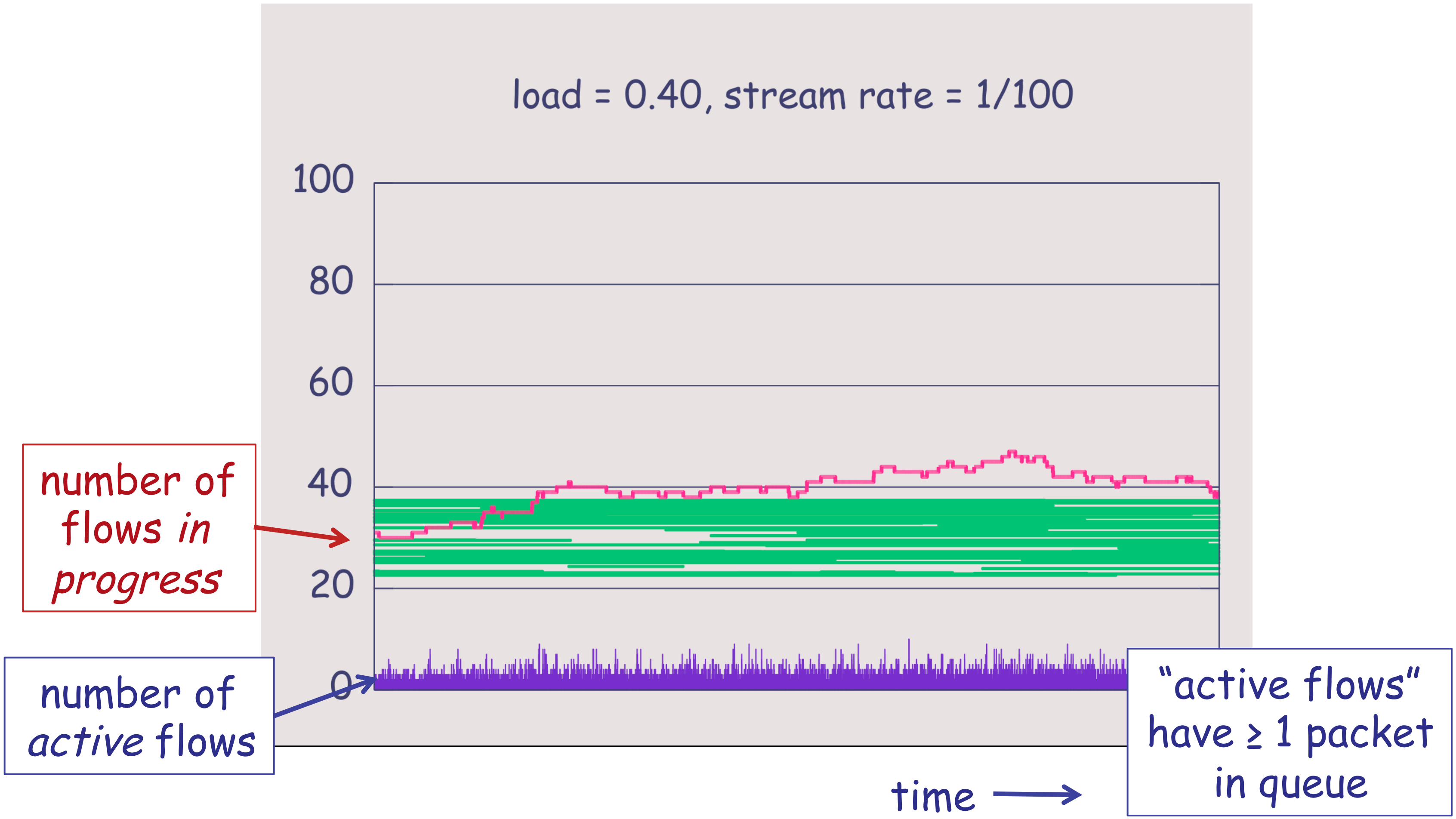
- the number of flows using a fairly shared link is small until load approaches 100% (*for any link capacity*)
- therefore, fair queuing schedulers are feasible and scalable
- our simulations make Markovian assumptions but the results for the number of active flows are true for much more general traffic [Ben-Fredj 2001]



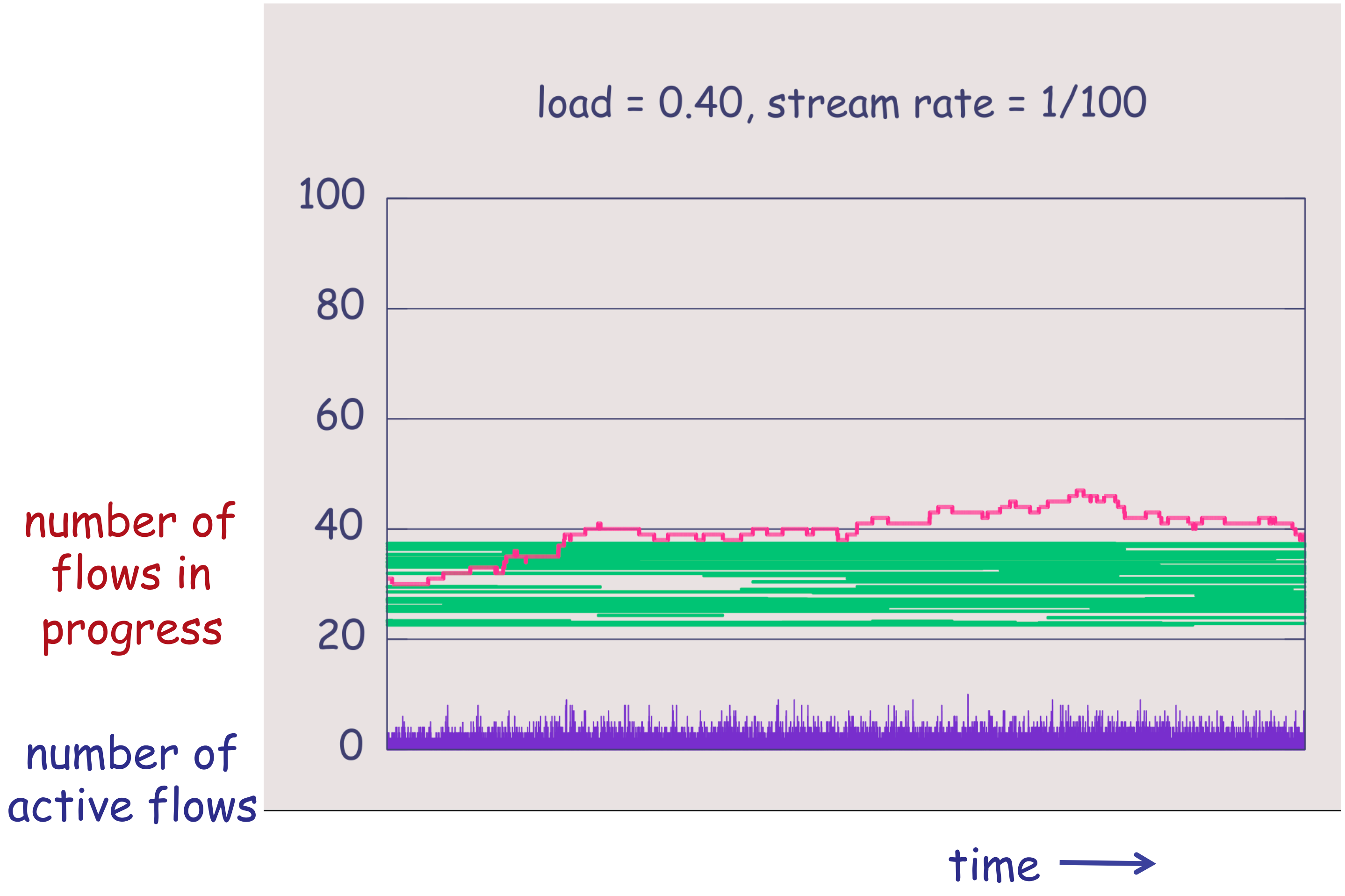
More simulations

- on Internet core links (≥ 10 Gbps), the vast majority of flows cannot use all available capacity; their rate is constrained elsewhere on their path (eg, ≤ 10 Mbps)
- consider a link shared by flows whose maximum rate is only 1% of the link rate
  - conservatively assume these flows emit packets as a Poisson process at rate proportional to the number of flows in progress

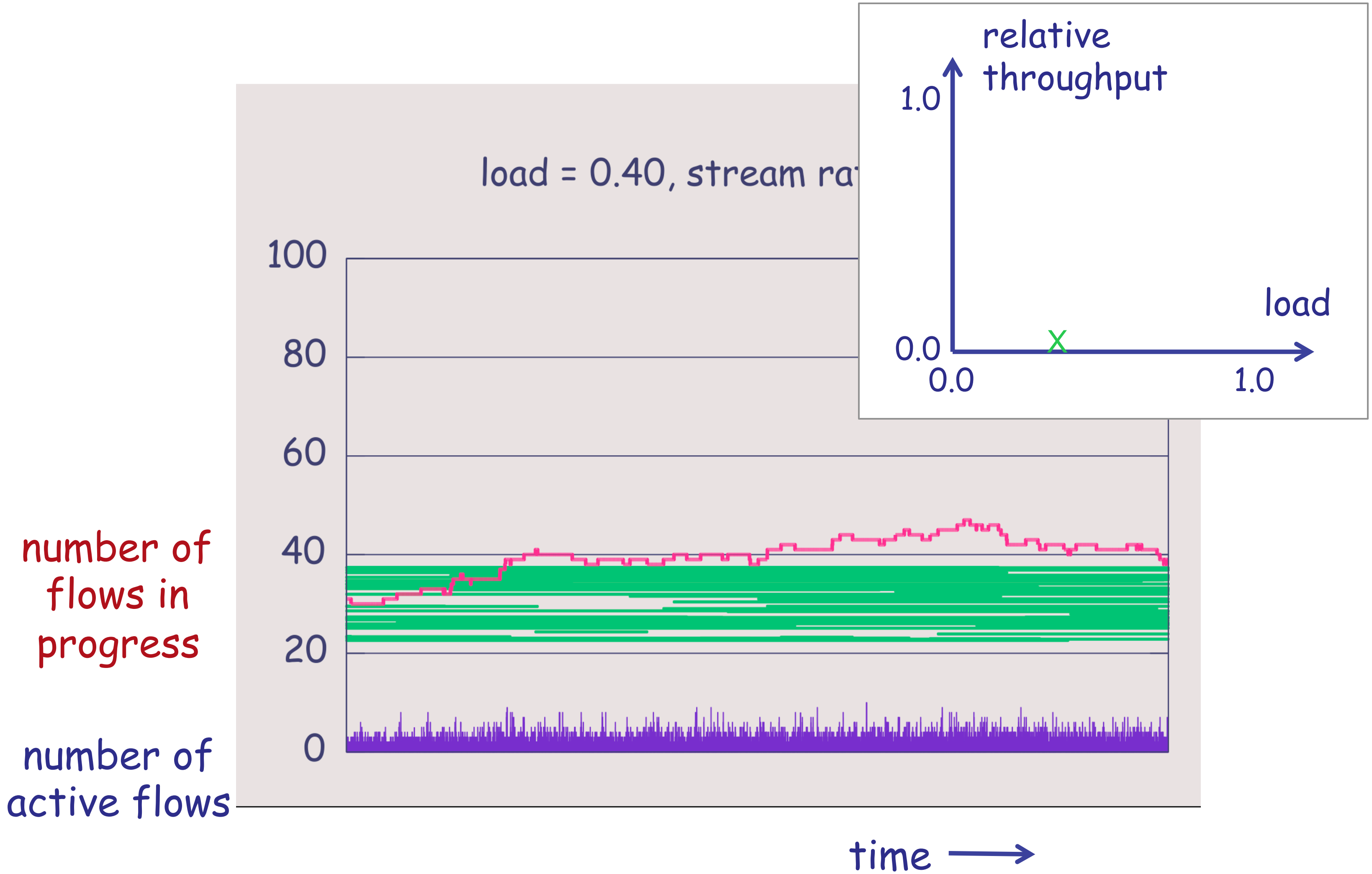
Performance with rate limited flows



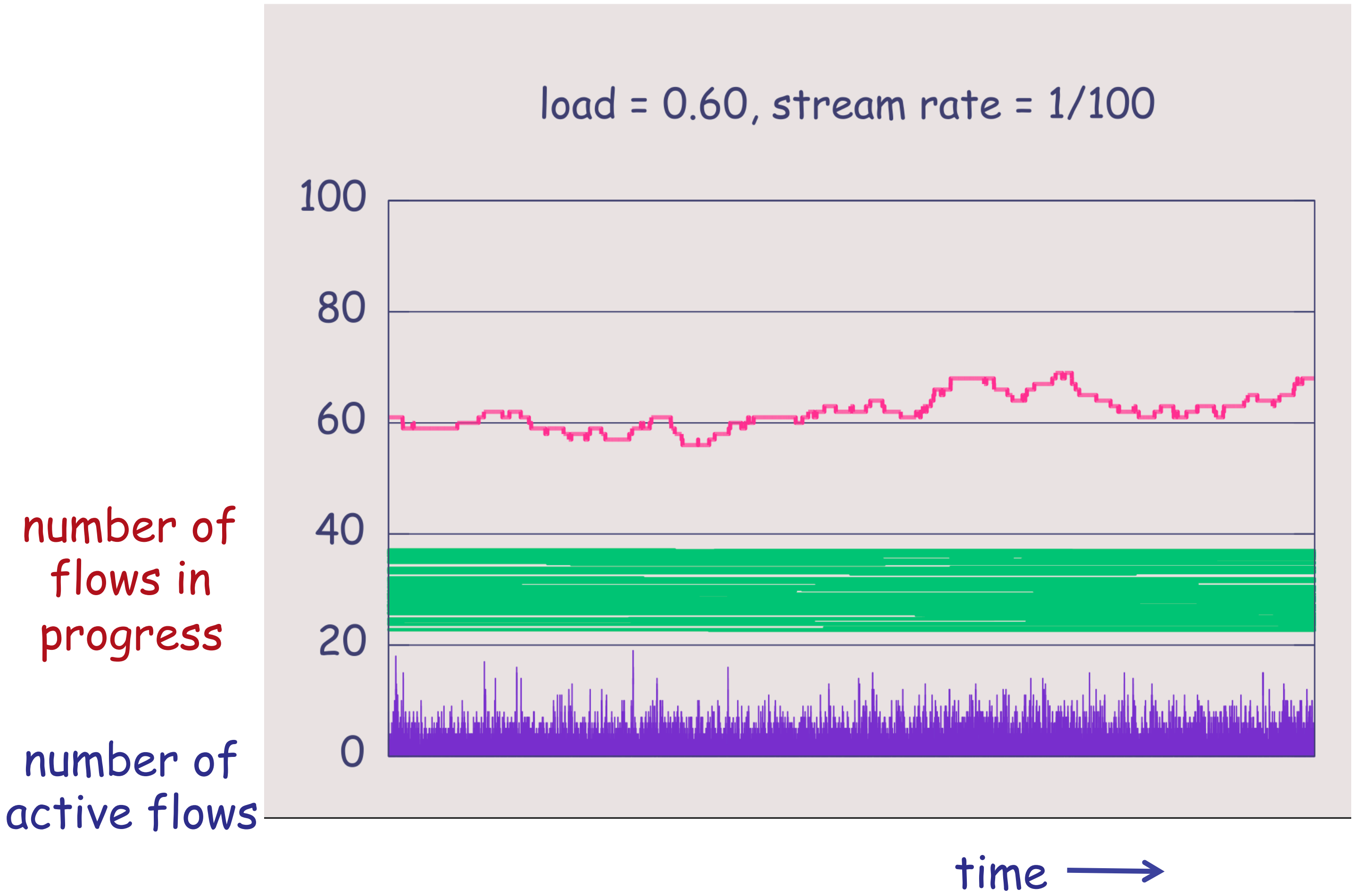
Performance with rate limited flows



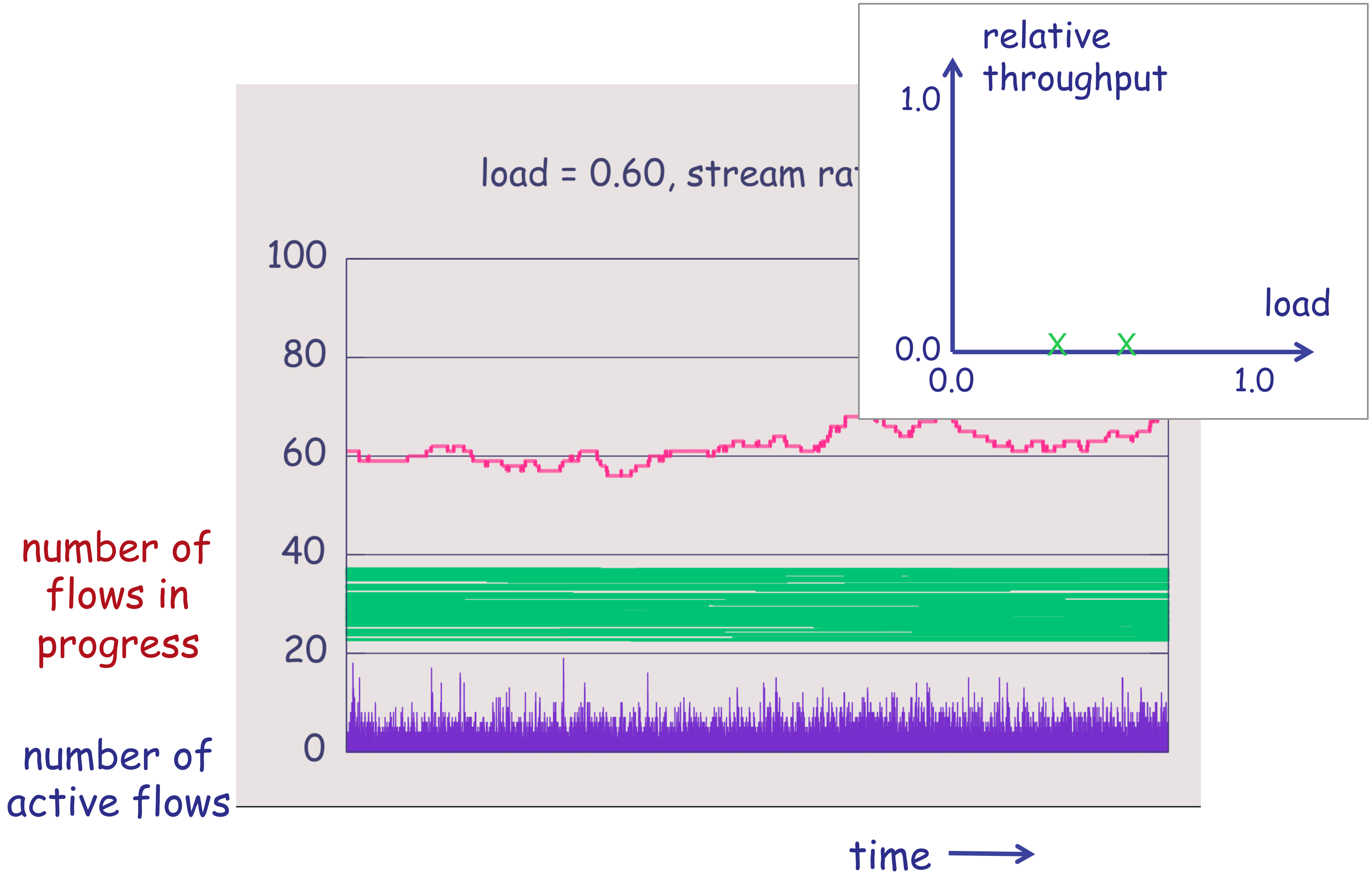
Performance with rate limited flows



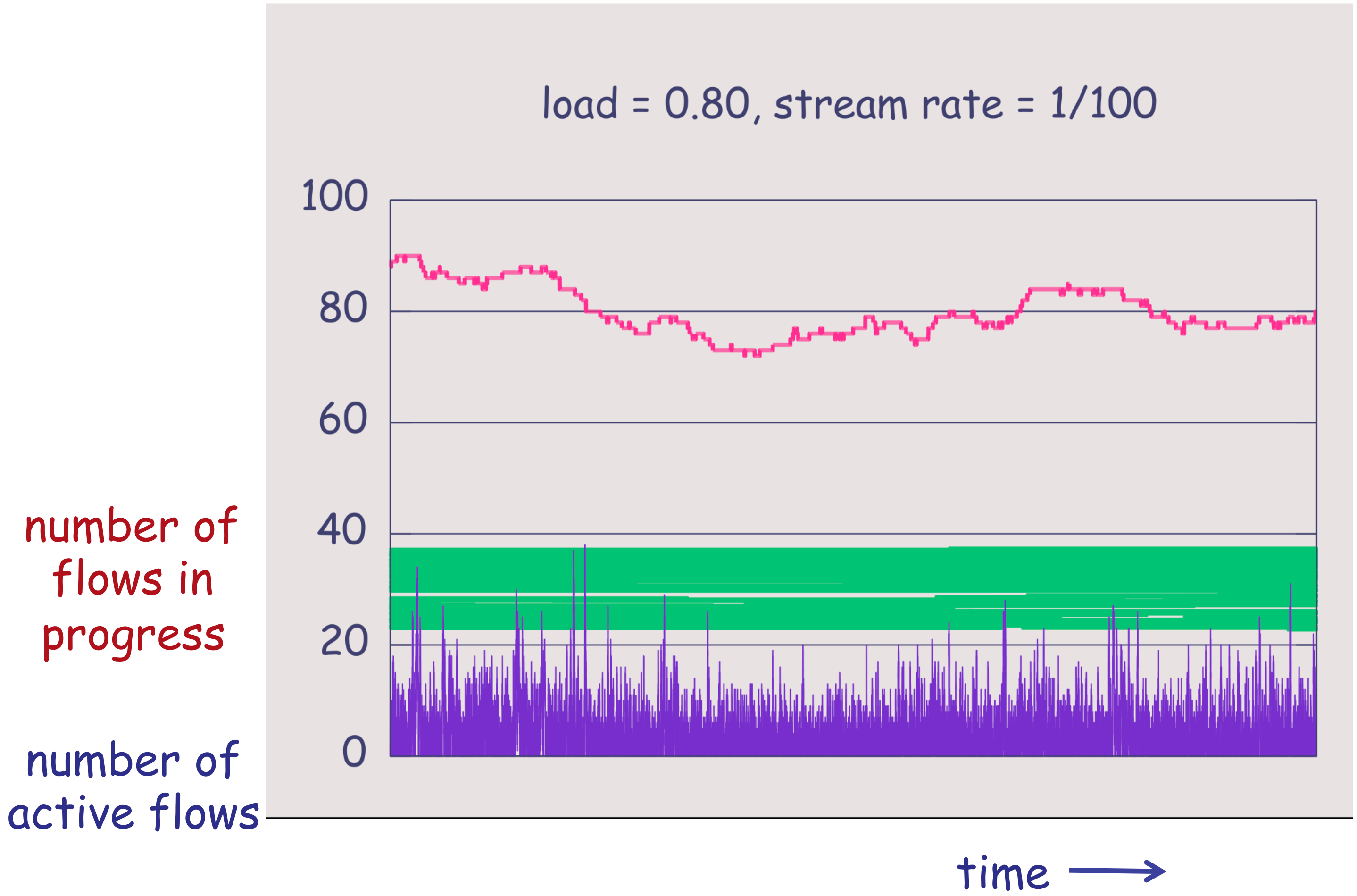
Performance with rate limited flows



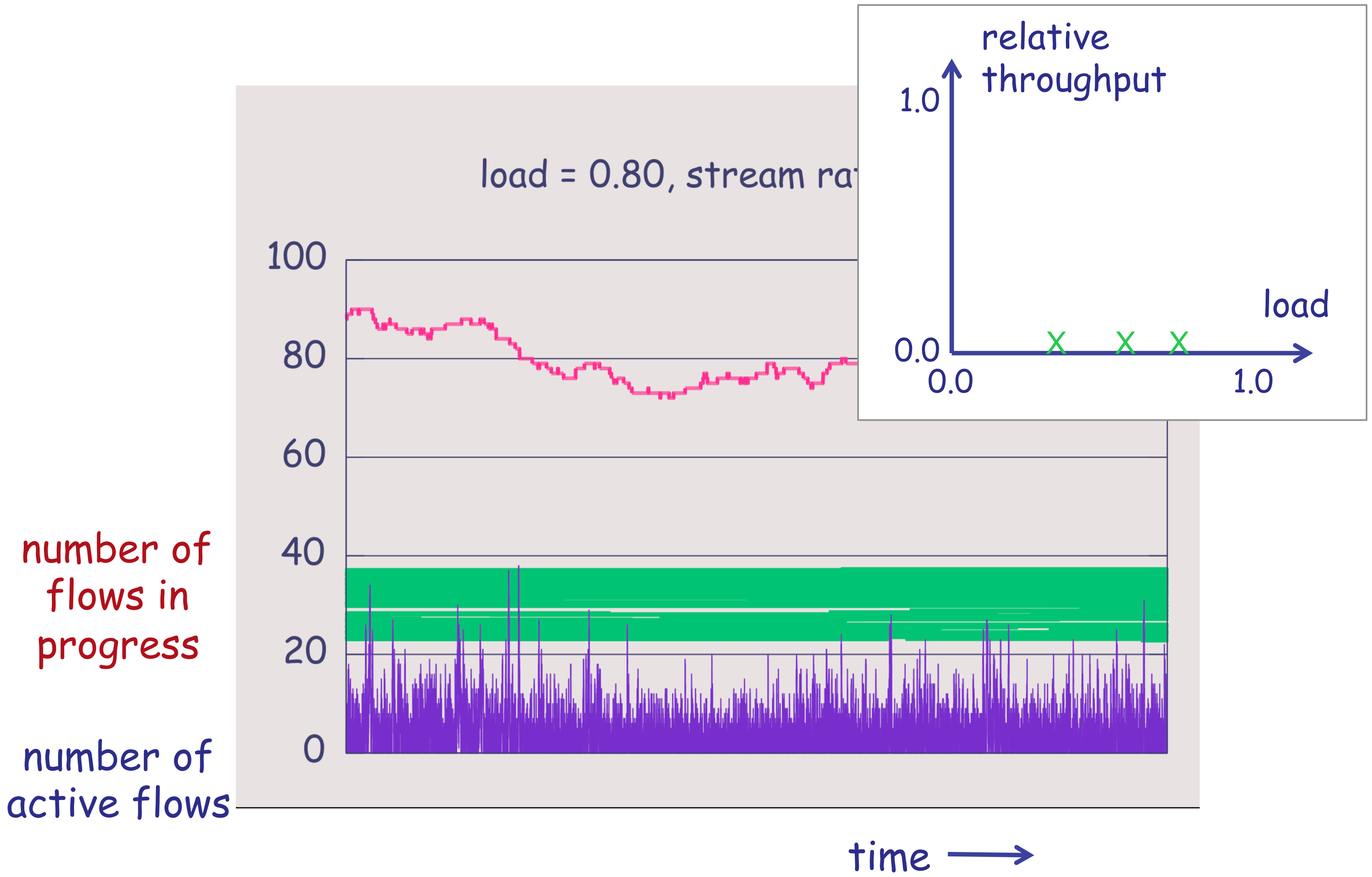
Performance with rate limited flows



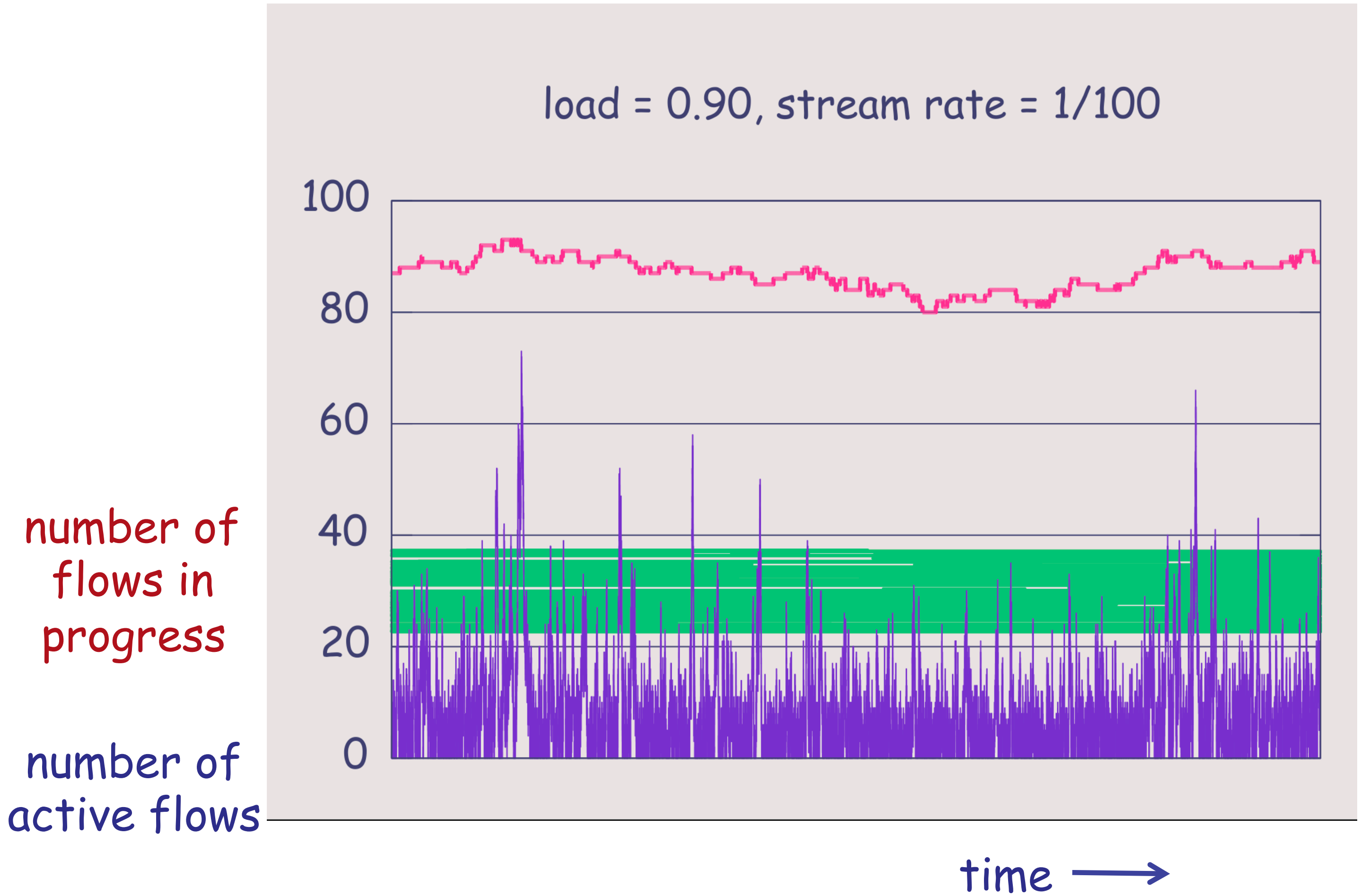
Performance with rate limited flows



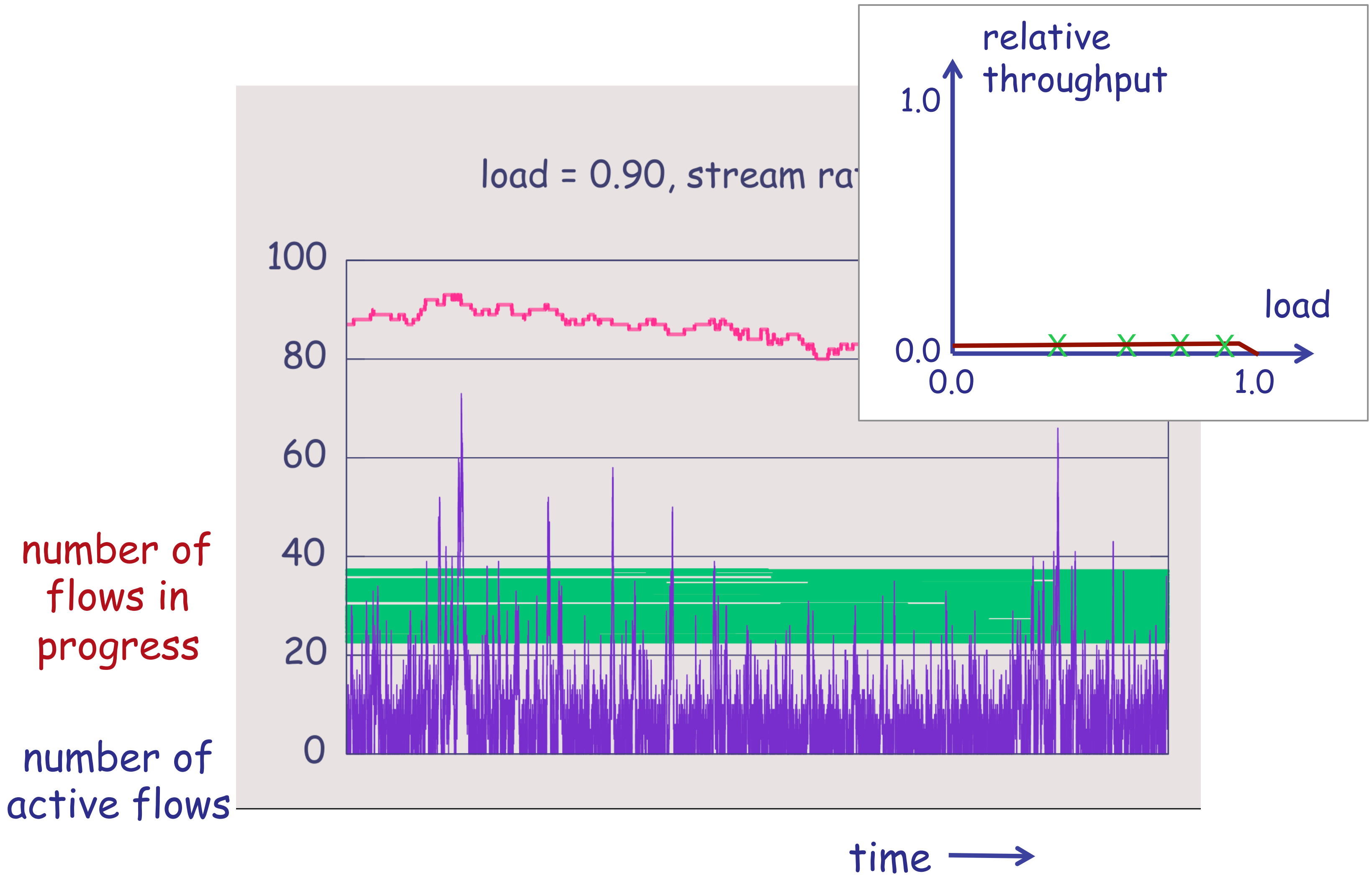
Performance with rate limited flows



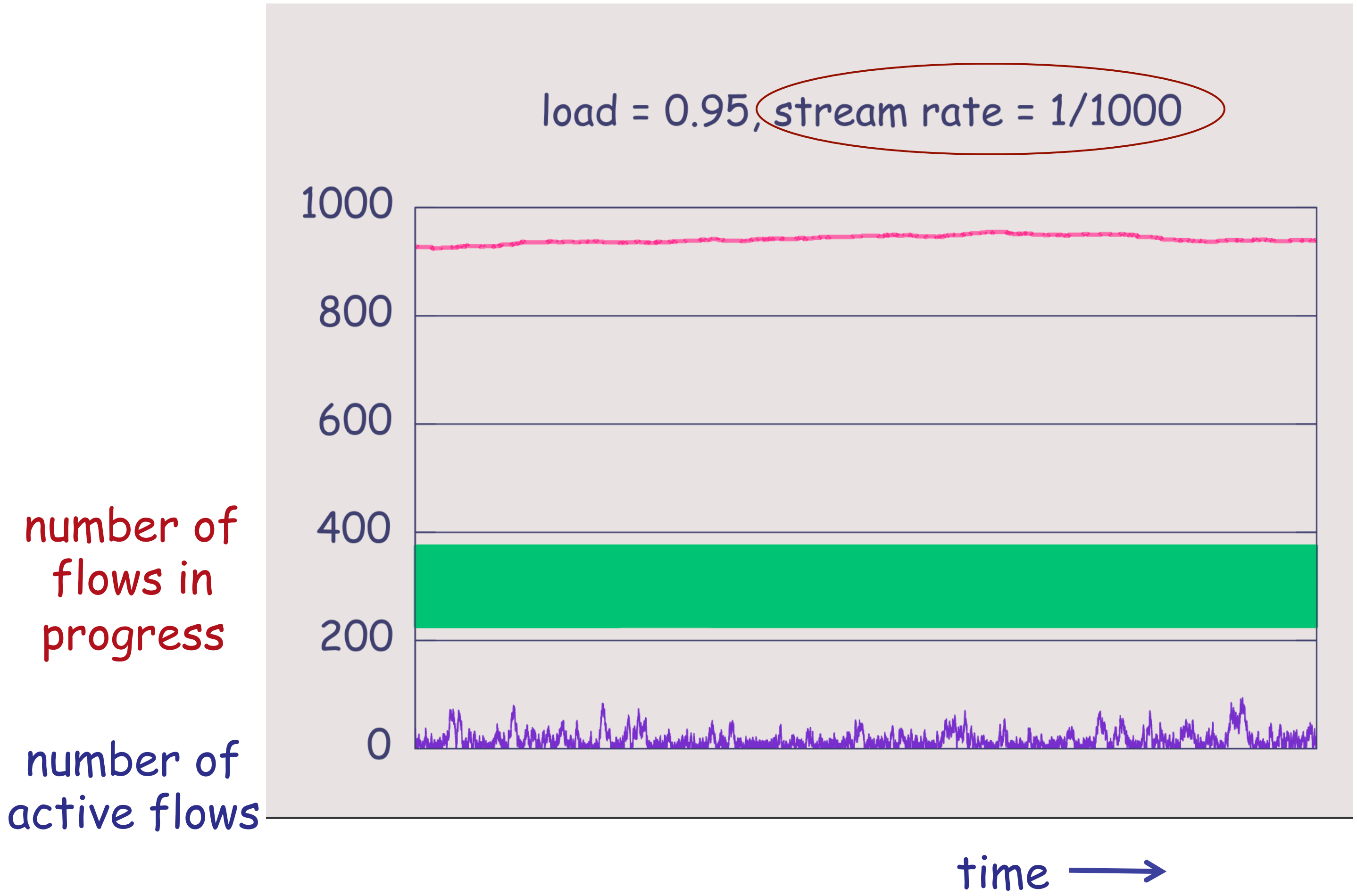
Performance with rate limited flows



Performance with rate limited flows



Performance with rate limited flows



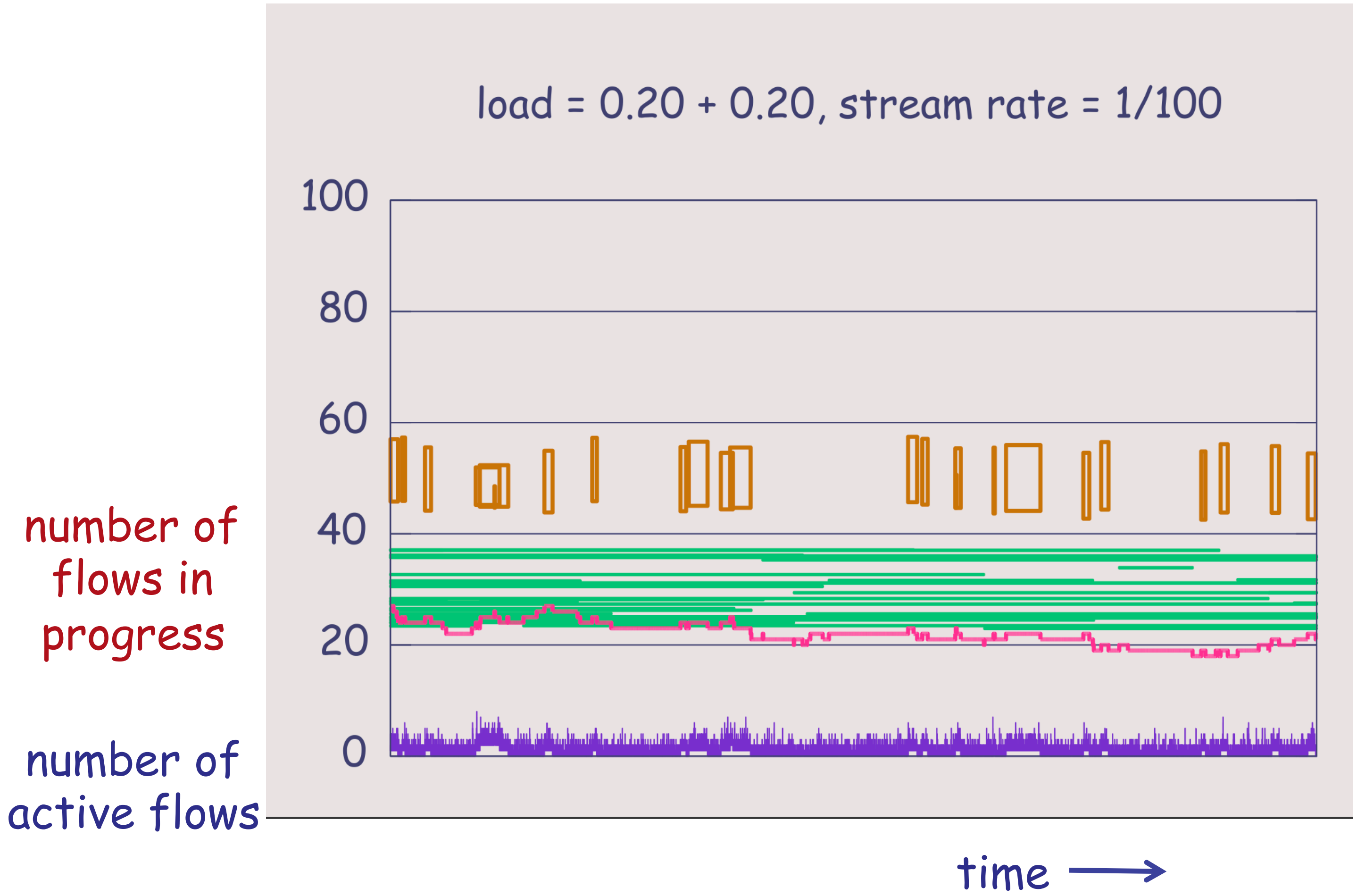
Observations 2

- most flows are not elastic and emit packets at their peak rate
- these flows are “active”, and need to be scheduled, only when they have a packet in the queue
- the number of *active* flows is small until load approaches 100%
- fair queuing is feasible and scalable, even when the number of flows *in progress* is very large

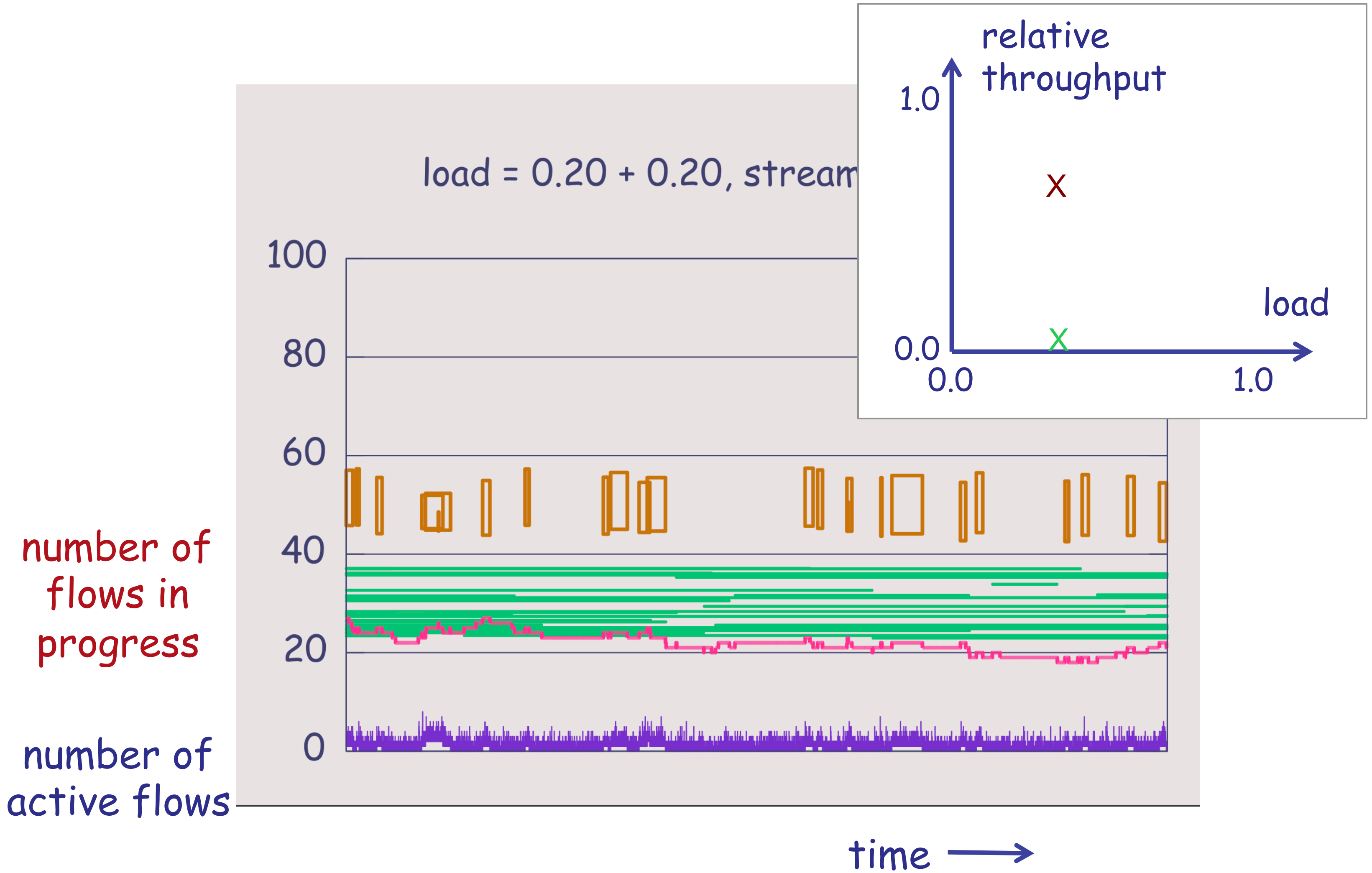
Yet more simulations

- links may be shared by many rate limited flows and a few elastic flows
- consider a link shared by 50% of traffic from flows whose peak rate is 1% of link rate and 50% elastic traffic

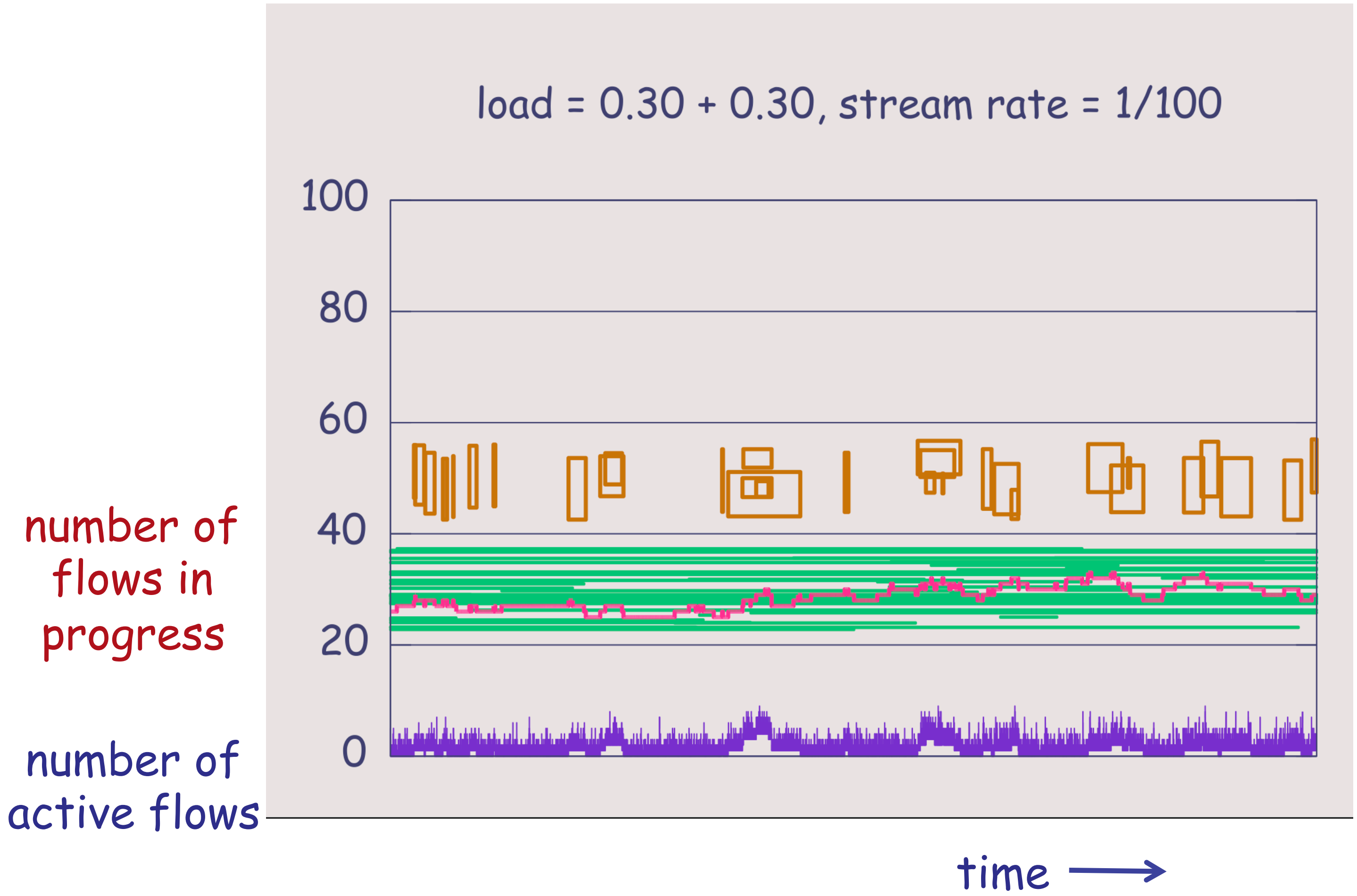
Performance of link with elastic and rate limited flows



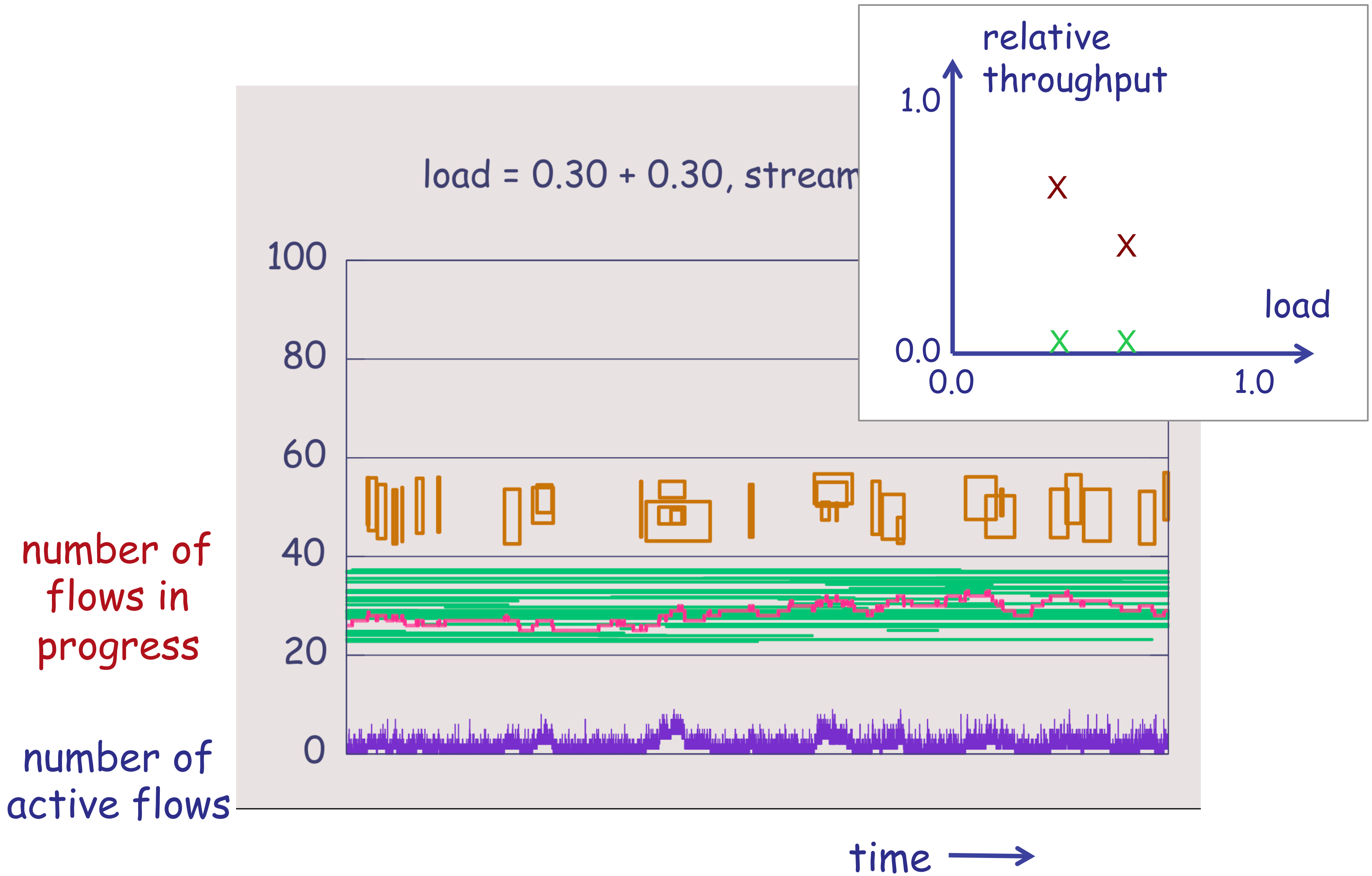
Performance of link with elastic and rate limited flows



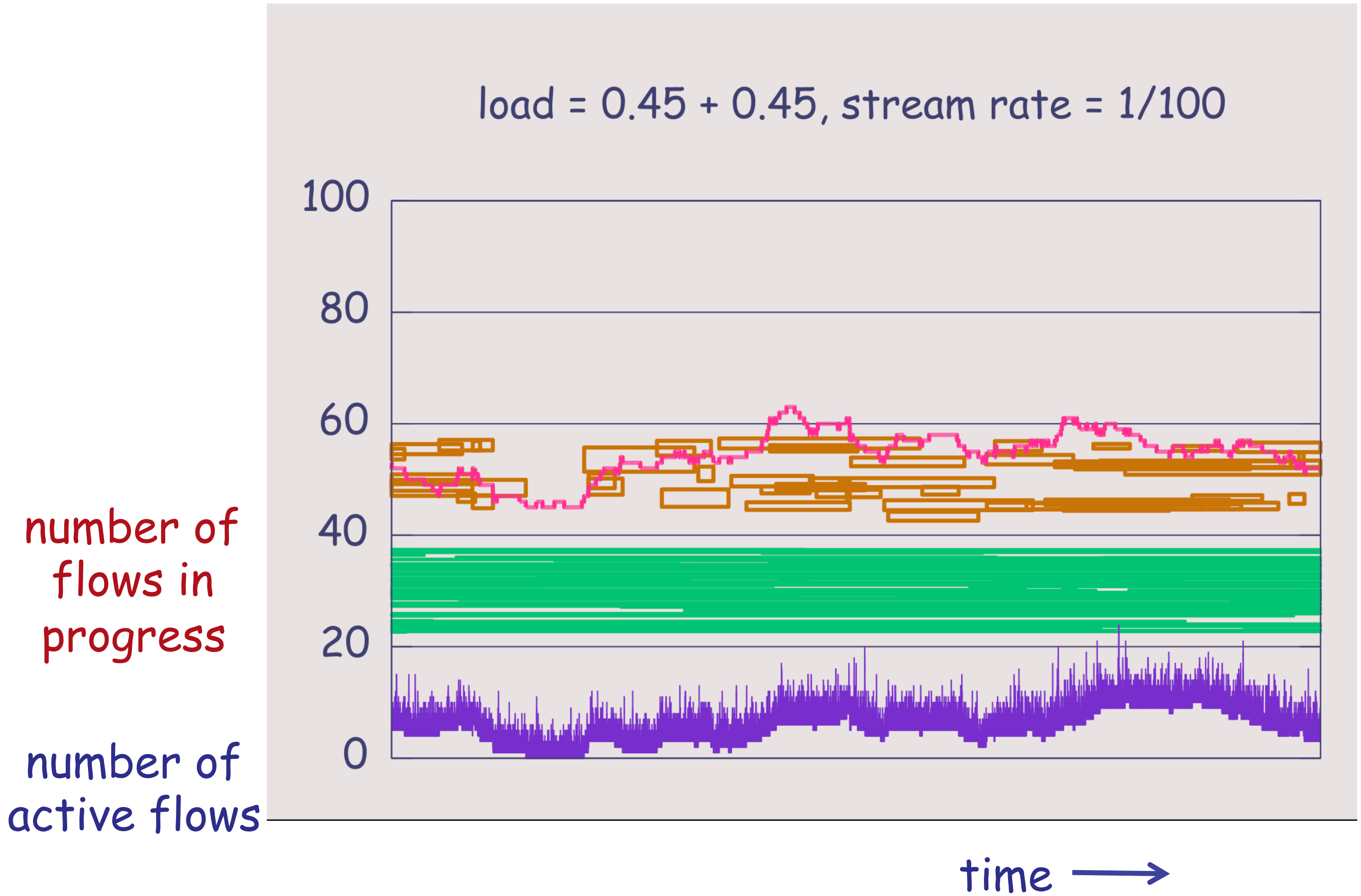
Performance of link with elastic and rate limited flows



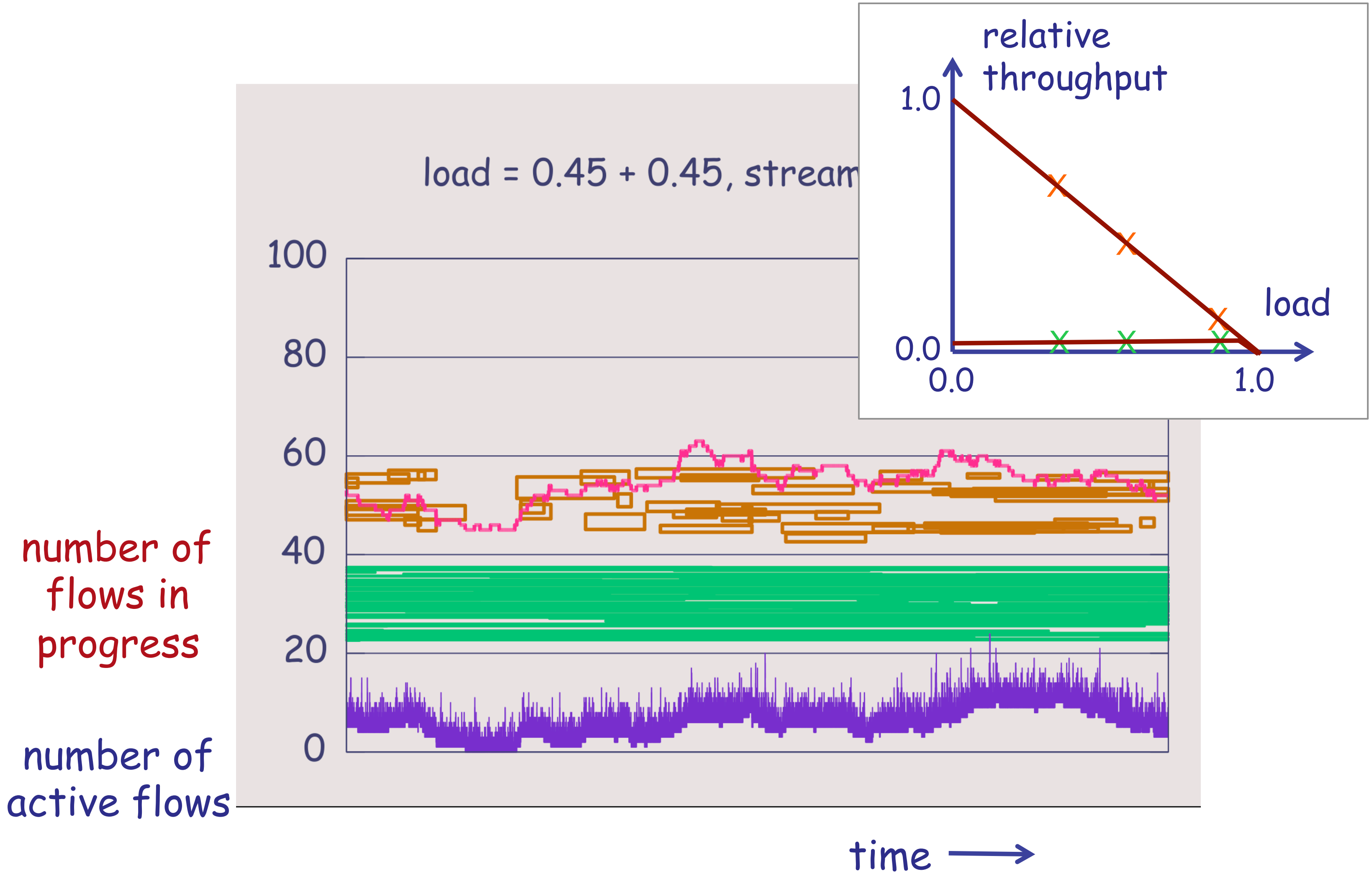
Performance of link with elastic and rate limited flows



Performance of link with elastic and rate limited flows



Performance of link with elastic and rate limited flows



Observations 3

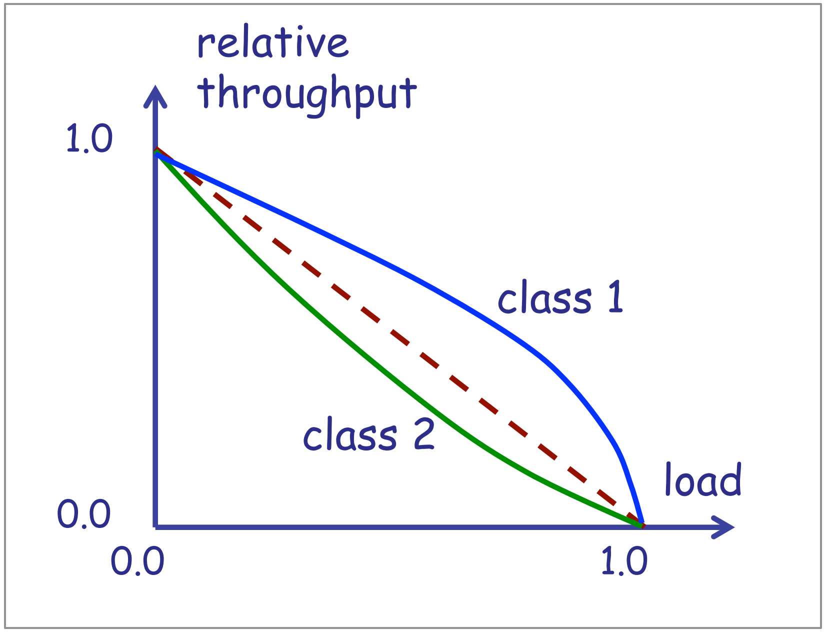
- the number of *active* flows is small (<100) with high probability until load approaches 100%
- therefore, fair queuing is feasible and scalable
- fair queuing means packets of limited peak rate flows see negligible delay:
  - they are delayed by at most 1 round robin cycle
  - this realizes implicit service differentiation since conversational and streaming flows are in the low rate category
  - a scheduler like DRR considers packets as belonging to new flows; we can therefore identify them and give them priority (cf. fq-codel)

Outline

- perceived problems, proposed solutions
  - bufferbloat
  - the data center interconnect
- the case for fair flow queuing
  - traffic at flow level scalability
  - fairness is all we need in the network
  - something else in the last/first mile

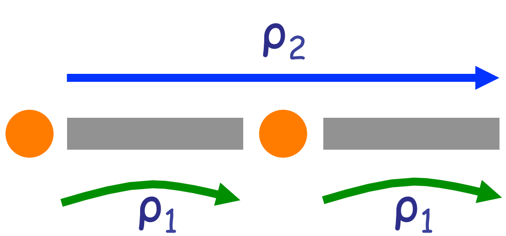
Fairness, not weighted fairness

- eg. class 1 flows get 10 times rate of class 2, equal traffic
  - limited gain for class 1, class 2 hardly suffers until load close to 1
  - from [Bonald and Masoulié, 2001]
- little advantage from weights for added cost of flow state
- little disadvantage when sharing is not perfectly fair

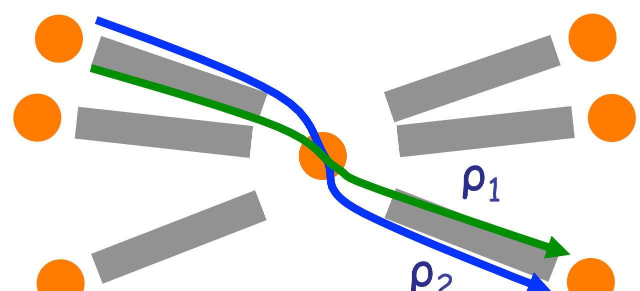


Fairness, not size-based scheduling

- SRPT makes it very easy to cheat by segmenting long flows
  - pFabric assumes cooperative sources in a private data center
- size-based scheduling in a network brings capacity loss
  - eg, for linear network, [Verloop et al, 2003] "*confirm the tendency for users with long routes and large service requirements to experience severe performance degradation*"
  - pFabric applies to a star-network – what capacity of SRPT ?



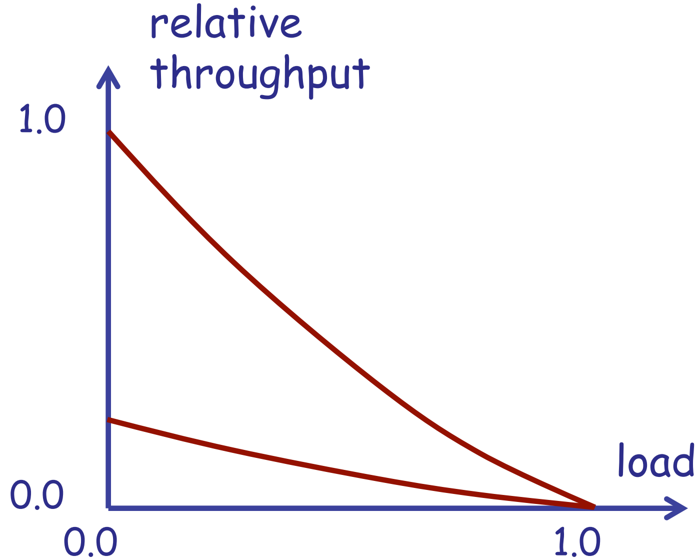
for strict priority:
only stable if $\rho_2 < (1-\rho_1)^2$



for strict priority:
only stable if $\rho_2 < (1-\rho_1)^2$

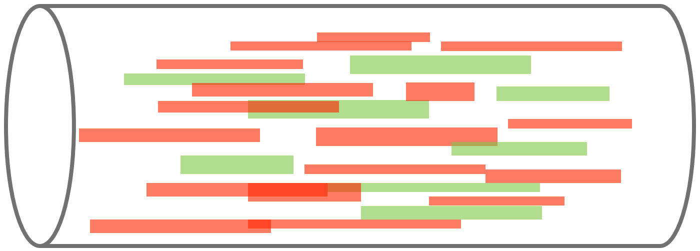
Fairness is stable if $\rho_l < 1$ for all links l

- as proved by [Paganini et al., 2009] for α-fair sharing and general flow size distribution
  - as occurs with reasonable congestion control
- it is stable if routers implement fair queuing, even if users do not use congestion control [Bonald et al, 2009]
  - though performance can suffer !



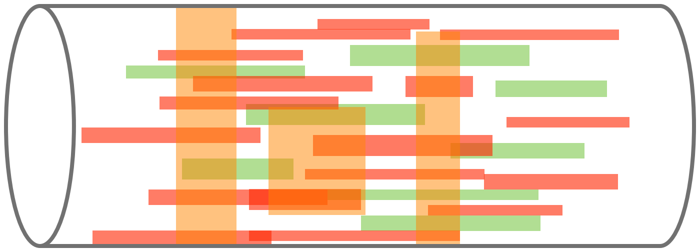
Fairness for predictable performance

- suppose a flow mix $\{(a_1,c_1),\ldots,(a_m,c_m)\}$ where a_i is load and c_i is (integer) peak rate of class i; link of (integer) capacity C; N is sum of peak rates of flows in progress
- in a loss system, the rate distribution, $f(n) = Pr[N=n]$, is insensitive and satisfies
  - $f(n) = 1/n \sum a_i f(n-c_i)$ for $0 \le i \le C$



Fairness for predictable performance

- suppose a flow mix $\{(a_1,c_1),\ldots,(a_m,c_m)\}$ where a_i is load and c_i is (integer) peak rate of class i; link of (integer) capacity C; N is sum of peak rates of flows in progress
- in a (balanced) fair system, the rate distribution, $f(n) = Pr[N=n]$, is insensitive and satisfies
  - $f(n) = 1/n \sum a_i f(n-c_i)$ for $0 \le n \le C$
    $1/C \sum a_i f(n-c_i)$ for $C < n$
- in a (maxmin) fair system, the congestion probability $P_c = Pr[\text{fair rate} < c]$ satisfies
  - $P_c < Erlang_{delay}(A/c, C/c)$ where $A = \sum a_i$ is overall demand
  - see [Bonald 2012] for justification and significance



Recommendation for traffic control on shared network links

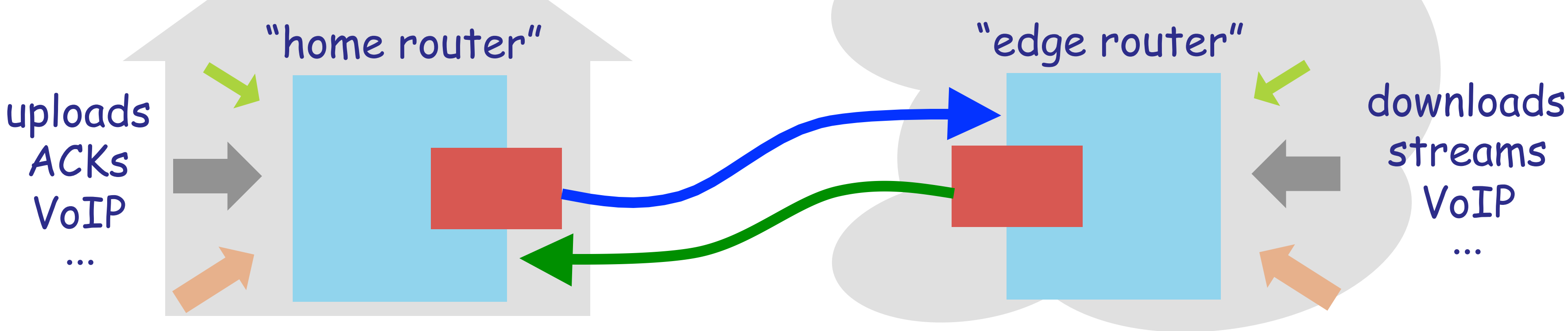
- implement per-flow fair queuing in router queues with longest queue drop
  - avoids relying on end-system congestion control
  - and realizes implicit service differentiation
  - and is scalable and feasible
- view fairness as an *expedient* not a socio-economic objective
  - yielding predictable performance: an "Internet Erlang formula"
- apply traffic engineering to ensure load is not too close to 100% and implement overload controls in case this fails
- this works for the Internet and data center interconnects but access networks need more than fair sharing

Outline

- perceived problems, proposed solutions
  - bufferbloat
  - the data center interconnect
- the case for fair flow queuing
  - traffic at flow level and scalability
  - fairness is all we need in the network
  - something else in the last/first mile

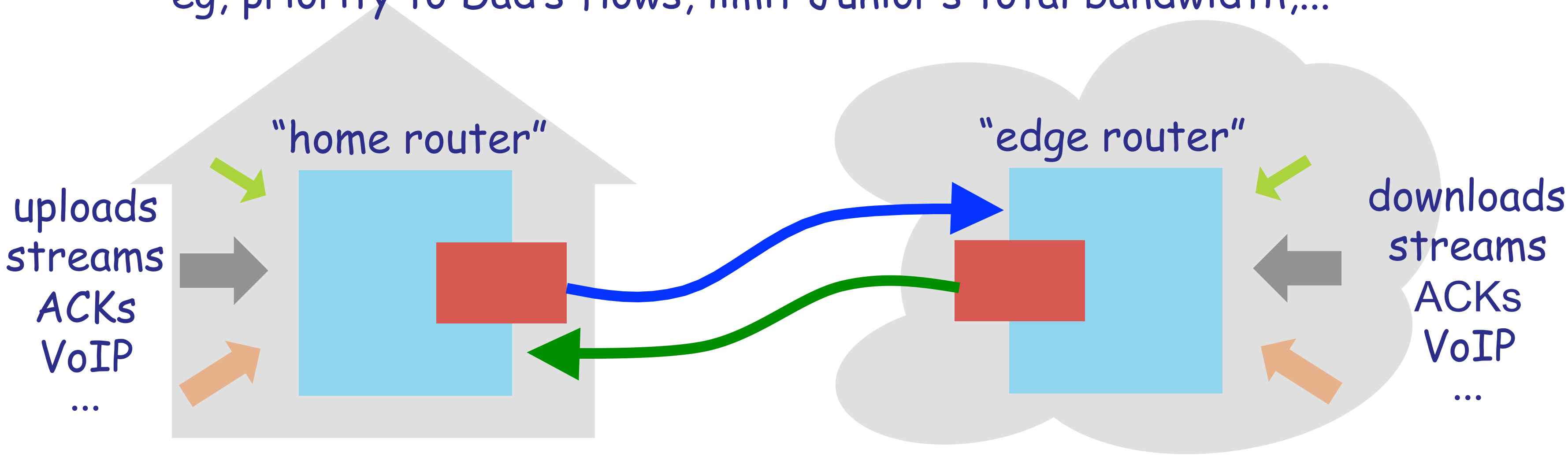
Sharing the first/last mile

- this is the usual bottleneck for Internet flows
  - for DSL, cable, wireless, fiber
- AQM and congestion control or flow-aware scheduling?
  - for upstream and downstream
- flow fairness is not enough
  - eg, for a video streamed at more than the fair rate
- who rules? who decides priorities?
  - surely the user, not the network operator



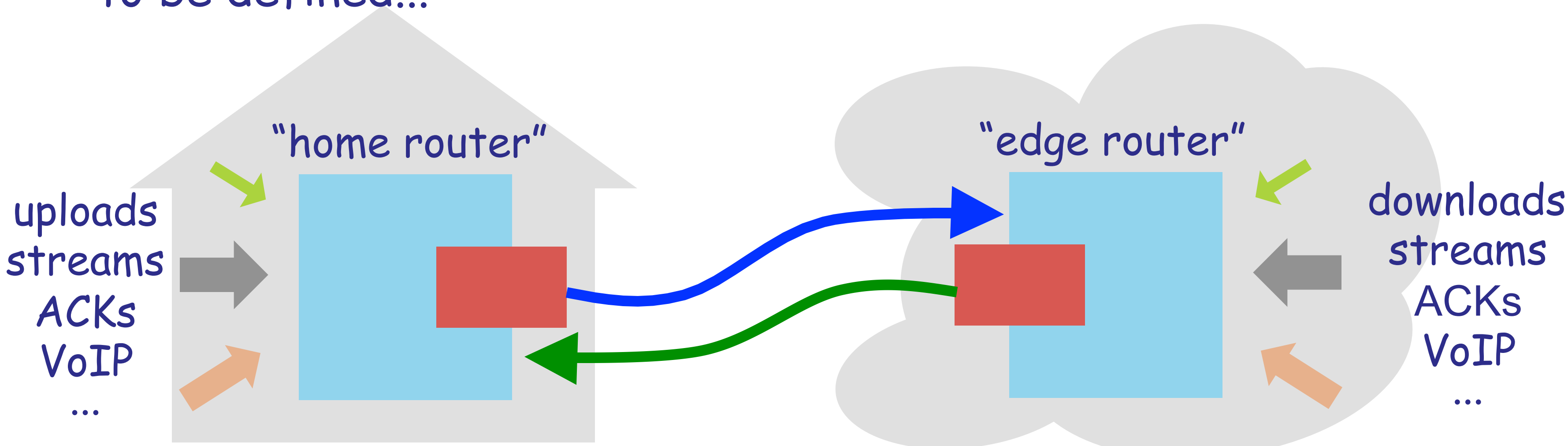
Sharing the first/~~last~~ mile

- CoDel, PIE, fq-codel do not distinguish classes of service
  - eg, though users have priorities (eg, background uploads)
- also, difficult coexistence of AQM and low priority congestion ctrl
  - eg, CoDel, FQ,... give same share to LEDBAT & TCP [Gong 2013]
- prefer explicit, per-flow scheduling to realize user's fairness and priority objectives
  - eg, priority to Dad's flows, limit Junior's total bandwidth,...



Sharing the ~~first~~/last mile

- operators differentiate managed and over-the-top services
  - though users may want other priorities (eg, priority to Skype, background downloads)
- user control would be a better alternative, requiring:
  - a flow scheduler in the router trading off complexity and flexibility
  - a signalling protocol allowing the user to dictate priorities
- to be defined...



Conclusions

- new AQM and congestion controls are a poor, short term fix
  - to perceived congestion in the home network (bufferbloat) and the data center interconnect
- fair flow queuing, proposed since 1985, is the preferred solution for high capacity shared network links
  - performance robust to end-system behaviour
  - provably scalable and therefore feasible
  - yields an "Erlang formula" for the Internet
- first/last mile sharing needs more complex per-flow scheduling
  - enabling "priority to video", "priority to Dad",...
  - both upstream and downstream, under user control

References

- Alizadeh, et al., "Data center TCP (DCTCP)" SIGCOMM, 2010.
- Alizadeh, et al., "Less is more: trading a little bandwidth for ultra-low latency in the data center", NSDI, 2012.
- Alizadeh et al. "pFabric: minimal near-optimal datacenter transport", SIGCOMM, 2013.
- Ballani et al., "Towards predictable datacenter networks", SIGCOMM, 2011.
- Ben-Fredj et al., "Statistical bandwidth sharing: a study of congestion at flow level". SIGCOMM, 2001.
- Bonald and Massoulié, "Impact of fairness on Internet performance", SIGMETRICS, 2001.
- Bonald and Proutière, "Insensitive bandwidth sharing in data networks". Queueing Systems, 2003
- Bonald and Roberts, "Internet and the Erlang formula", SIGCOMM Comput. Commun. Rev., 2012.
- Floyd and Fall, "Promoting the use of end-to-end congestion control", IEEE ToN, 1999.
- Gong et al. "Fighting the bufferbloat: on the coexisitence of AQM and low priority congestion controlé, TMA, 2013.

References (2)

- Hong et al., "Finishing flows quickly with preemptive scheduling", SIGCOMM, 2012.
- Kortebi et al. "Cross-protect: implicit service differentiation and admission control", HPSR, 2004.
- Kumar et al., "Beyond best efort: router architectures for the differentiated services of tomorrow's Internet", IEEE Comm Mag., 1998.
- Nagle, "On packet switches with infinite storage", RFC 970, 1985.
- Nichols and Jacobson, "Controlling queue delay", Comm. ACM, 2012.
- Paganini et al. "Stability of networks under general file size distribution with alpha fair rate allocation", Allerton, 2009.
- Pan et al. "PIE: a lightweight control scheme to address the bufferbloat problem", HPSR, 2013.
- Vamanan et al., "Deadline-aware datacenter TCP (D2TCP)", SIGCOMM, 2012.
- Verloop et al., "Stability of size-based scheduling disciplines in resource sharing networks", Perfom. Eval., 2005.
- Wilson et al., "Better never than late: meeting deadlines in datacenter networks", SIGCOMM, 2011.