THE ROYAL SOCIETY

Queueing theory, control theory, & buffer sizing

Damon Wischik

www.wischik.com/damon

DARPA grant W911NF05-1-0254

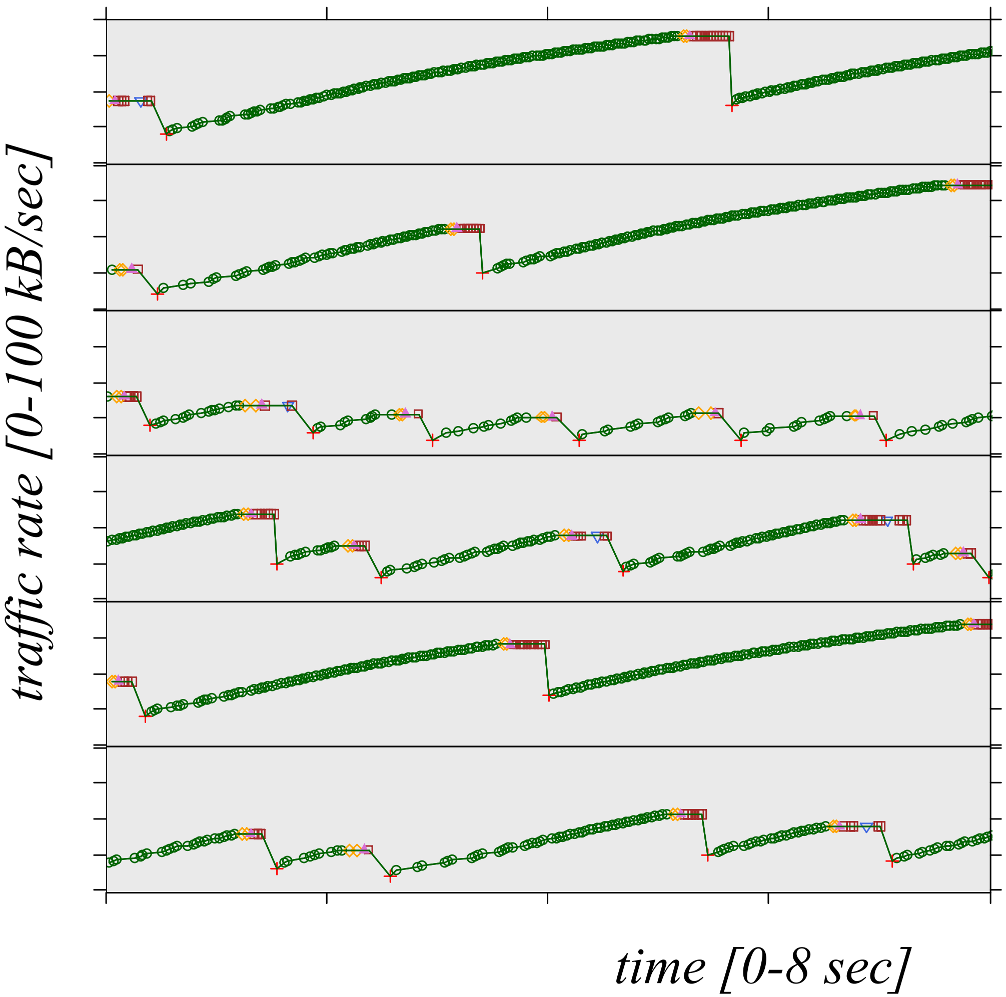
My interest

- Internet routers have buffers,
  - to accomodate bursts in traffic
  - to keep the link fully utilized

- How big do the buffers need to be, to accommodate TCP traffic?
  - 3 GByte? Rule of thumb says buffer = bandwidth×delay
  - 300 MByte? buffer = bandwidth×delay/√#flows
    [Appenzeller, Keslassy, McKeown, 2004]
  - 30 kByte? constant buffer size, independent of line rate [Kelly, Key, etc.]

- What is the role of probabilistic queueing theory?
  Is it all just fluid models & differential equations?

TCP

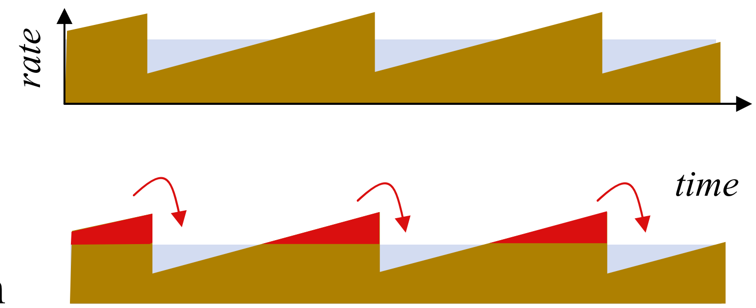



```
if (seqno > _last_acked) {
    if (!_in_fast_recovery) {
        _last_acked = seqno;
        _dupacks = 0;
        inflate_window();
        send_packets(now);
        _last_sent_time = now;
        return;
        }
    if (seqno < _recover) {
        uint32_t new_data = seqno - _last_acked;
        _last_acked = seqno;
        if (new_data < _cwnd) _cwnd -= new_data; else _cwnd=0;
        _cwnd += _mss;
        retransmit_packet(now);
        send_packets(now);
        return;
        }
    uint32_t flightsize = _highest_sent - seqno;
    _cwnd = min(_ssthresh, flightsize + _mss);
    _last_acked = seqno;
    _dupacks = 0;
    _in_fast_recovery = false;
    send_packets(now);
    return;
    }
if (_in_fast_recovery) {
    _cwnd += _mss;
    send_packets(now);
    return;
    }
_dupacks++;
if (_dupacks!=3) {
    send_packets(now);
    return;
    }
_ssthresh = max(_cwnd/2, (uint32_t)(2 * _mss));
retransmit_packet(now);
_cwnd = _ssthresh + 3 * _mss;
_in_fast_recovery = true;
_recover = _highest_sent;
}
```

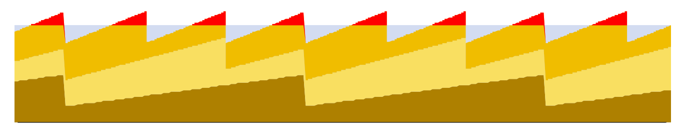
TCP sawtooth & buffer size

- The traffic rate produced by a TCP flow follows a ‘sawtooth’

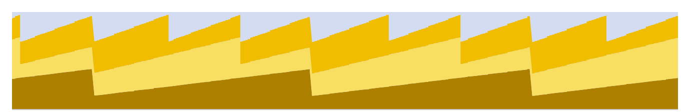
- To prevent the link’s going idle, the buffer must be big enough to smooth out a sawtooth
  - buffer = bandwidth×delay



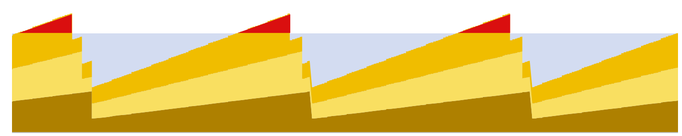
- When there are many TCP flows, the sawteeth should average out, so a smaller buffer is sufficient
  - buffer = bandwidth×delay/√#flows
    [Appenzeller, Keslassy, McKeown, 2004]



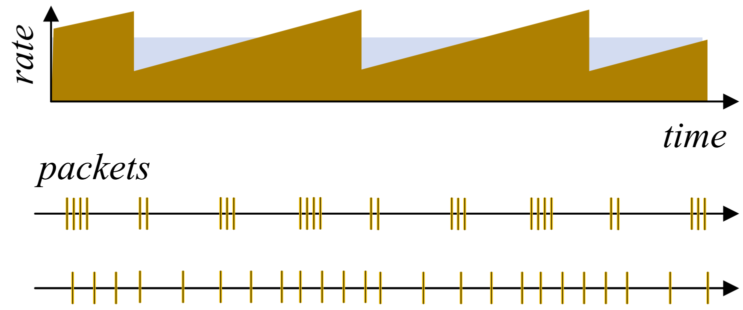
- If we could keep the traffic rate just a little bit lower, virtually no buffer would be needed...



- ...unless the flows are synchronized

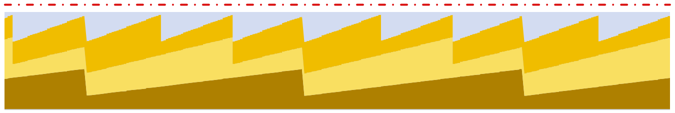


TCP packets & buffer size

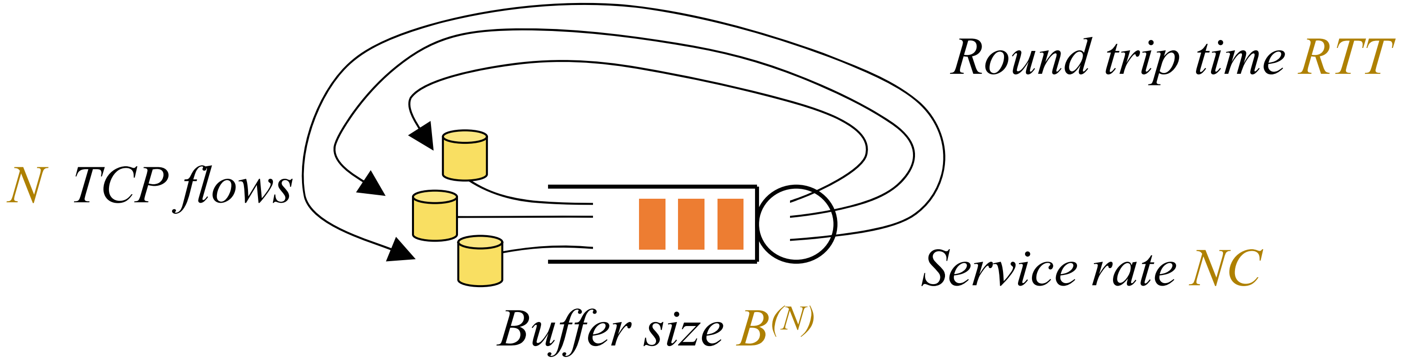


- TCP traffic is made up of packets
  - there may be packet clumps,
    if the access network is fast
  - or the packets may be spaced out

- Even if we manage to keep total data rate < service rate,
  chance alignment of packets will still lead to some queueing & loss,
  so we can’t dispense with buffers entirely



Formulate a maths question

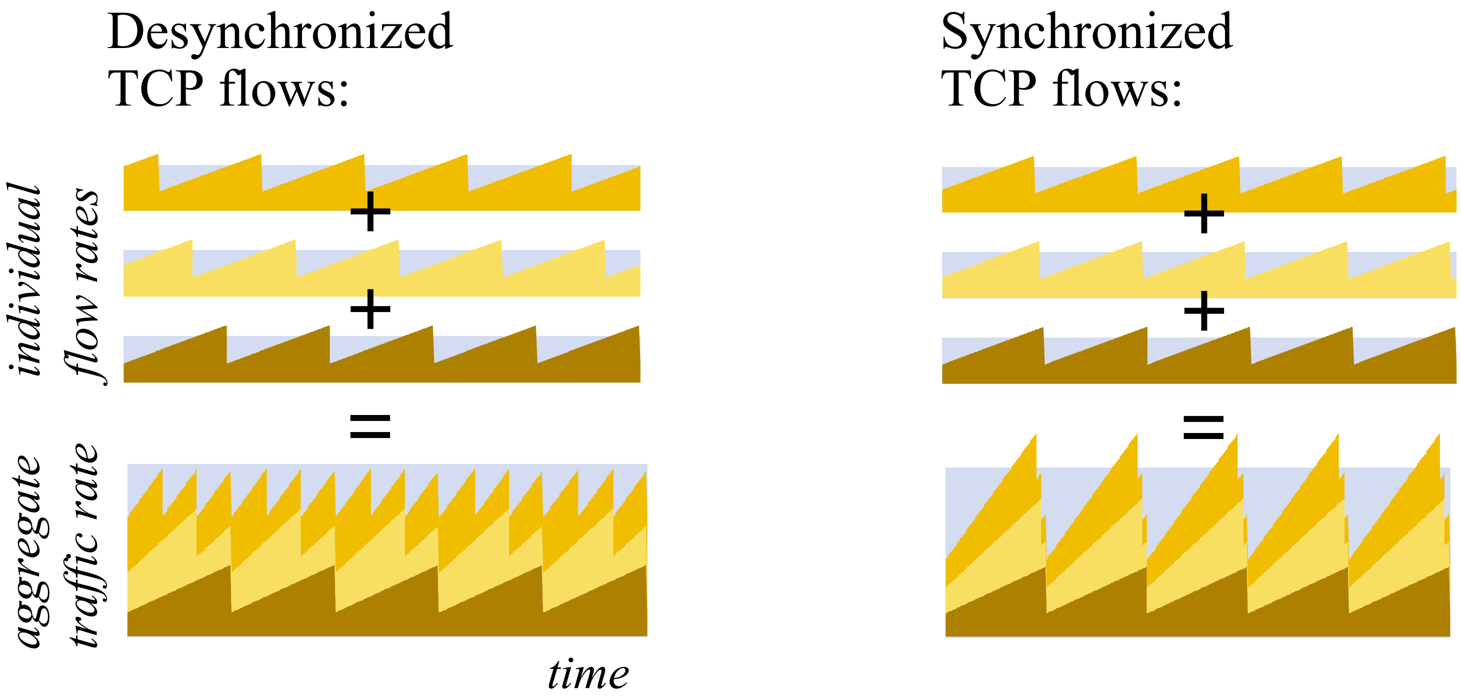


- What is the limiting queue-length process, as $N \rightarrow \infty$, in the three regimes
  - *large buffers* $B^{(N)}=BN$
  - *intermediate buffers* $B^{(N)}=B\sqrt{N}$
  - *small buffers* $B^{(N)}=B$

- *Why are these limiting regimes interesting? What are the alternatives?* [Bain, 2003]

TCP traffic model

- A single TCP flow follows a characteristic ‘sawtooth’

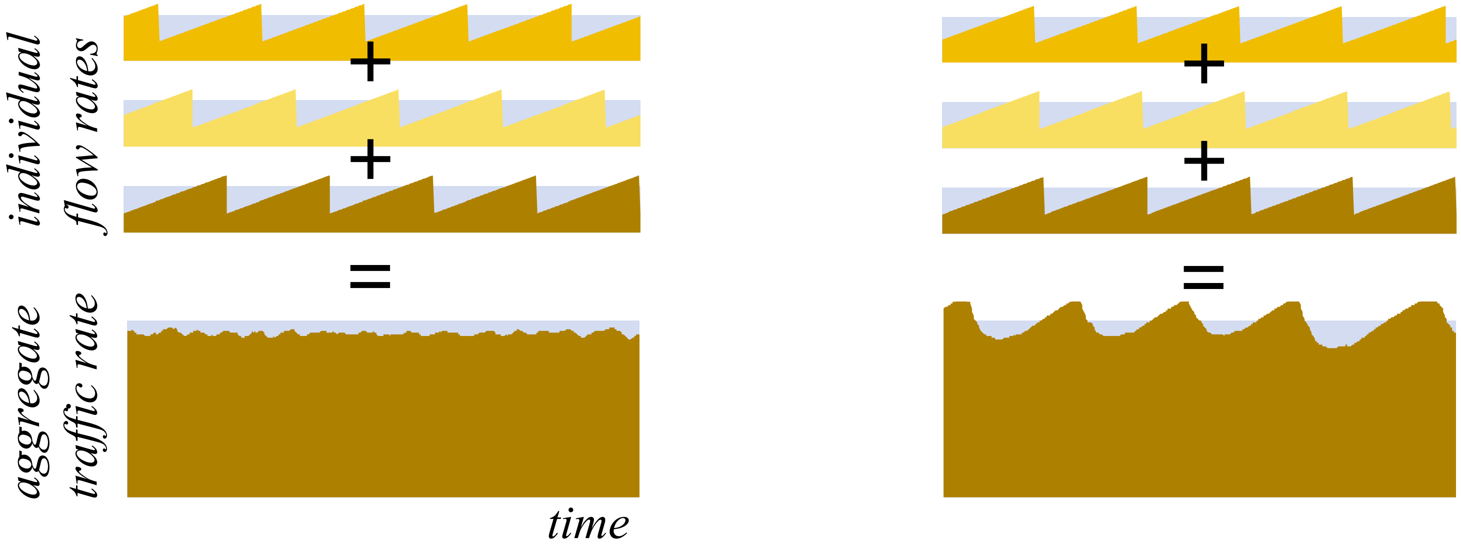


TCP traffic model

- A single TCP flow follows a characteristic 'sawtooth'
- Many TCP flows added together are smoother

Desynchronized
TCP flows:

Synchronized
TCP flows:



TCP traffic model

- When there are many TCP flows, the average traffic rate x_t varies smoothly, according to a delay differential equation [Misra, Gong, Towsley, 2000]

$$\frac{dx_t}{dt} = \frac{1}{RTT^2} - p_{t-RTT}\, x_{t-RTT} \frac{x_t}{2}$$

- The equation involves
  - p_t, the packet loss probability at time t
  - RTT, the average round trip time

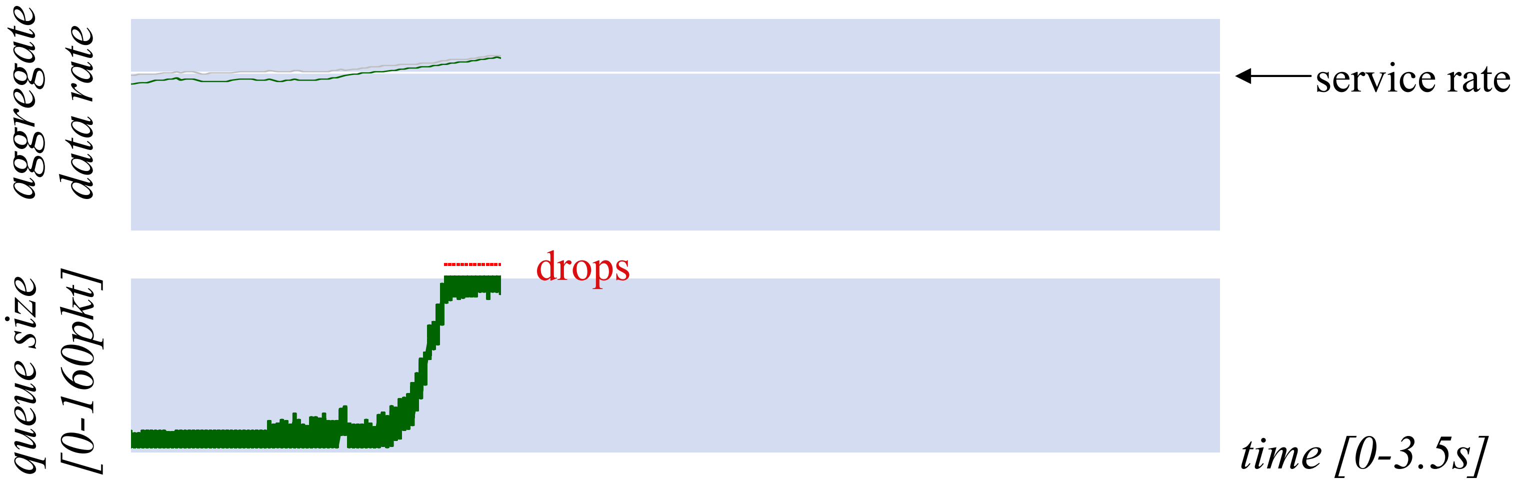
Queue model

- How does packet loss probability p_t depend on buffer size?
- The answer depends on the buffer size
  - *large buffers* $B^{(N)}=BN$
  - *intermediate buffers* $B^{(N)}=B\sqrt{N}$
  - *small buffers* $B^{(N)}=B$

Large buffer $B^{(N)} = BN$

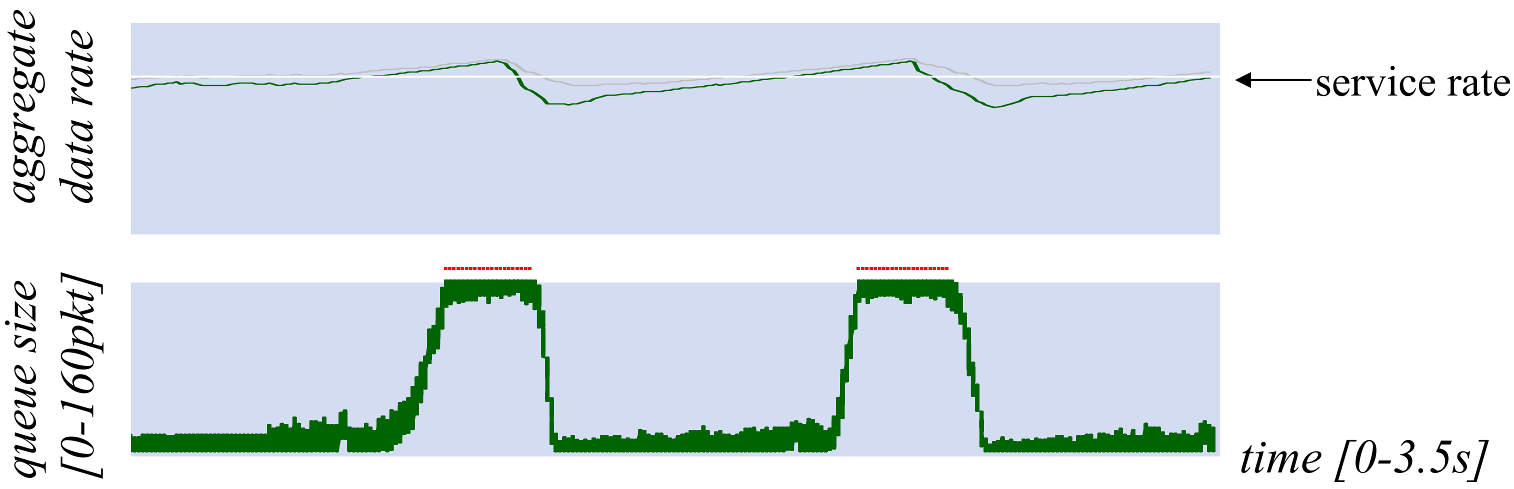
- When the aggregate data rate is less than the service rate, the queue stays small
- No packet drops, so TCPs increase their data rate

Large buffer $B^{(N)} = BN$



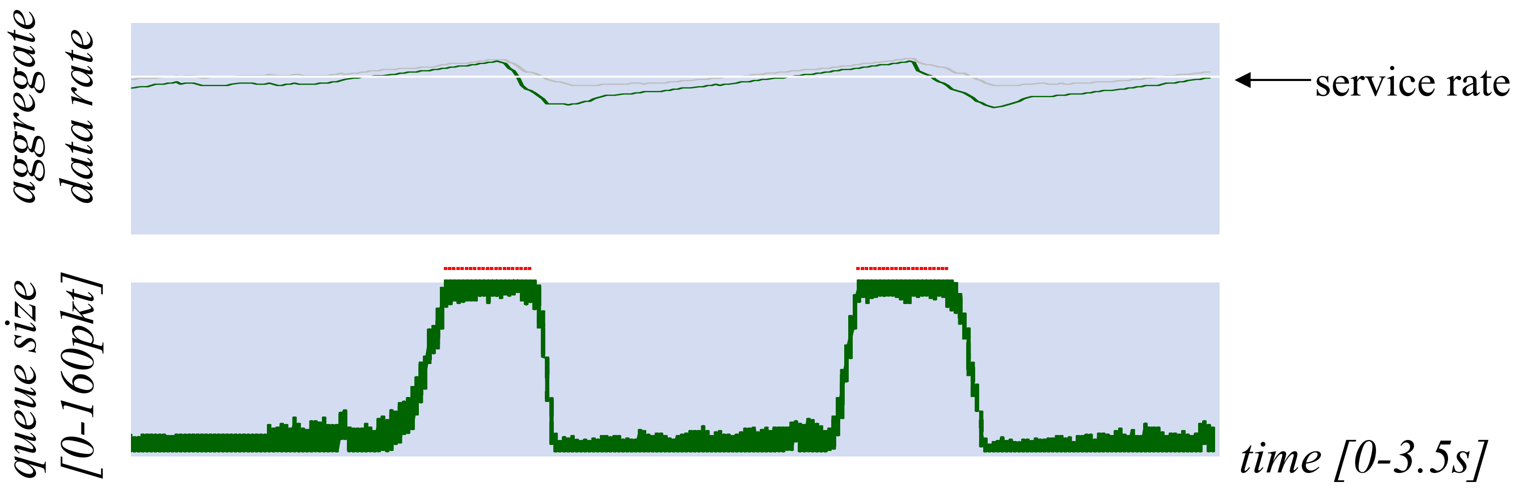
- When the aggregate data rate is less than the service rate, the queue stays small
- No packet drops, so TCPs increase their data rate
- Eventually the aggregate data rate exceeds the service rate, and a queue starts to build up
- When the queue is full, packets start to get dropped

Large buffer $B^{(N)} = BN$



- When the aggregate data rate is less than the service rate, the queue stays small
- No packet drops, so TCPs increase their data rate
- Eventually the aggregate data rate exceeds the service rate, and a queue starts to build up
- When the queue is full, packets start to get dropped
- One round trip time later, TCPs respond and cut back
  They may overreact, leading to synchronization *i.e. periodic fluctuations*

Large buffer $B^{(N)} = BN$



- Queue size & arrival rate vary on the same timescale
- The total queue size Nq_t satisfies the fluid model

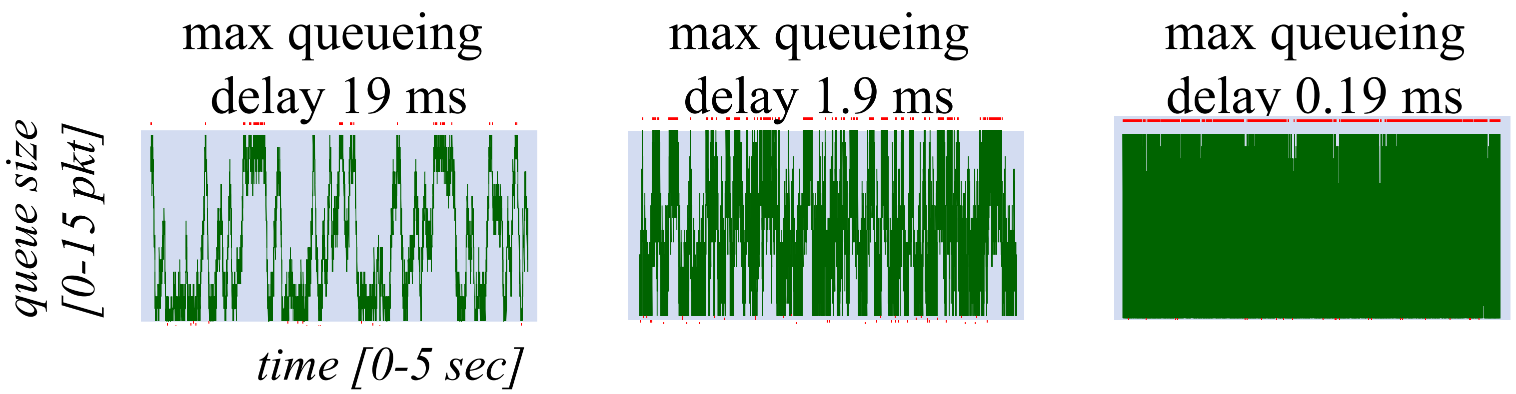
$$\frac{dq_t}{dt} = x_t(1 - p_t) - C$$

- When the queue is near full, Little's Law gives the drop probability

$$p_t = 1_{\{q_t = B\}}(x_t - C)/x_t$$

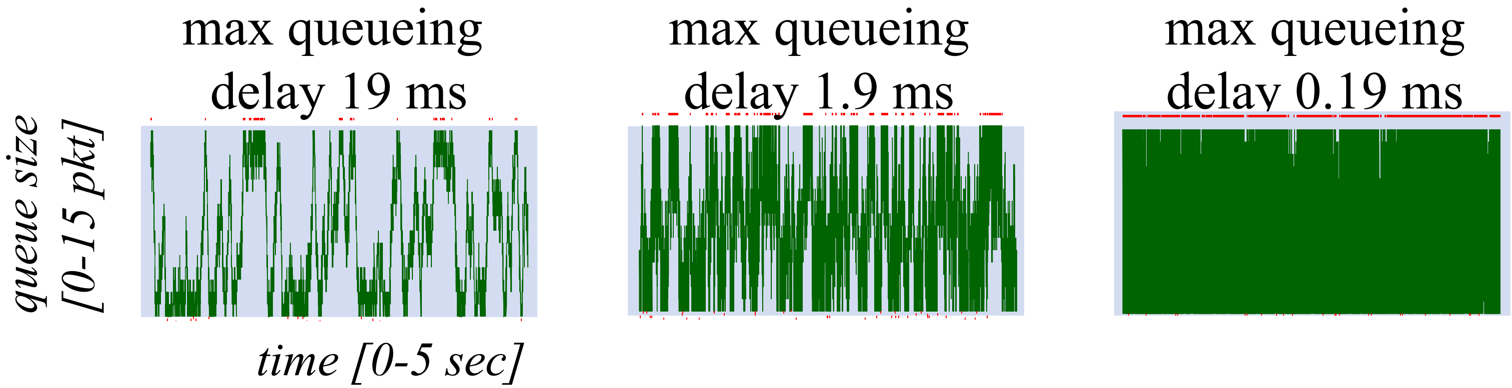
[cf McDonald, Reynier, 2003]

Small buffer $B^{(N)}=B$



- As the number of flows *N* and the capacity *NC* increase, we observe
  - queues aise because of chance alignments of packets
  - queue size fluctuates more and more rapidly, much more rapidly than variations in arrival rate
  - queue size distribution does not change
  - *like an M_{Nx} / M_{NC} / 1 / B queue*

Small buffer $B^{(N)}=B$

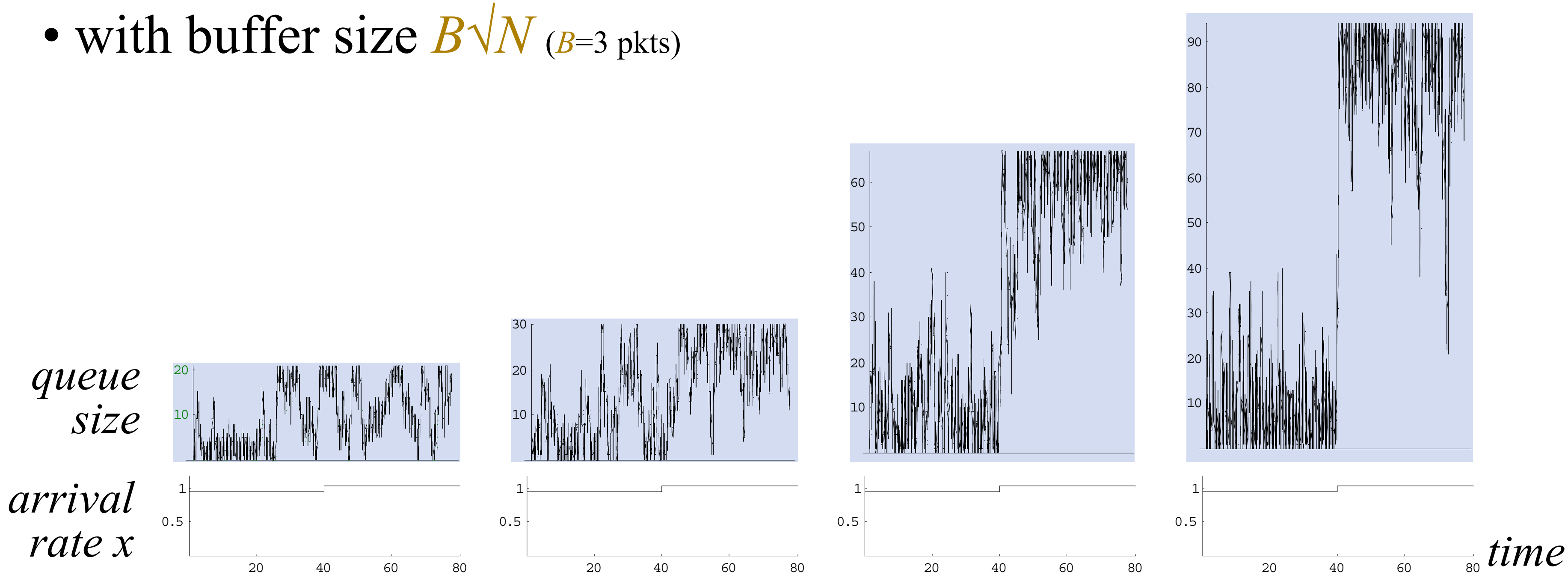


- We conjecture
  - a typical busy cycle lasts $O(1/N)$
  - packet arrivals over timescale $O(1/N)$ look like a Poisson process with constant arrival rate $x_t \approx x_{t+O(1/N)}$
  - drop probability converges to that for an M/D/1/B queue: $p_t \approx (x_t/C)^B$
- Evidence
  - In a queue with a small buffer, fed by arbitrary exogenous traffic, a typical busy cycle lasts $O(1/N)$, and queue size matches that in an M/D/1/B queue [Cao, Ramanan, 2002]
  - Over short timescales (<1ms), TCP traffic is approximately Poisson ["Internet traffic tends toward Poisson and independent as the load increases", Cao, Cleveland, Lin, Sun, 2002]

Intermediate buffers $B^{(N)}=B\sqrt{N}$

Consider a queue

- fed by N flows, each of rate x pkts/sec (x=0.95 then 1.05 pkts/sec)
- served at rate NC (C=1 pkt/sec)
- with buffer size $B\sqrt{N}$ (B=3 pkts)



N=50 N=100 N=500 N=1000

System summary

- x_t = average traffic rate at time t
  p_t = packet loss probability at time t
  C = capacity/flow B = buffer size RTT = round trip time N=# flows

- TCP traffic model

$$\frac{dx_t}{dt} = \frac{1}{RTT^2} - p_{t-RTT} x_{t-RTT} \frac{x_t}{2}$$

- Small buffer queueing model
  (though this is sensitive to traffic statistics)

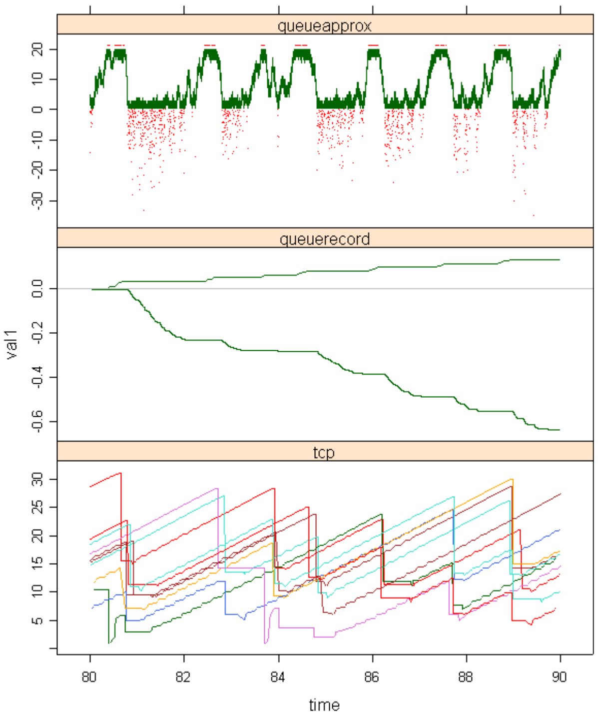
$$p_t \approx (x_t/C)^B$$

- Large buffer queueing model

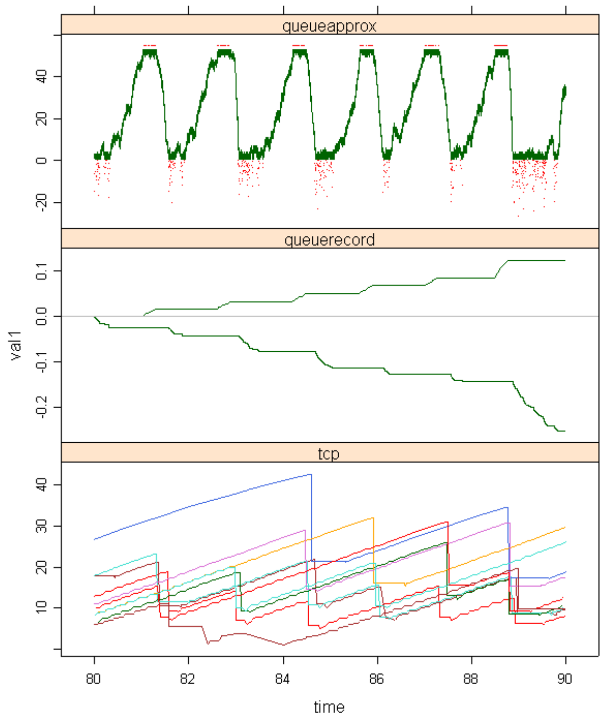
$$p_t = 1_{\{q_t = B\}} (x_t - C)/x_t$$
$$\frac{dq_t}{dt} = x_t(1-p_t) - C$$

Illustration 20 flows

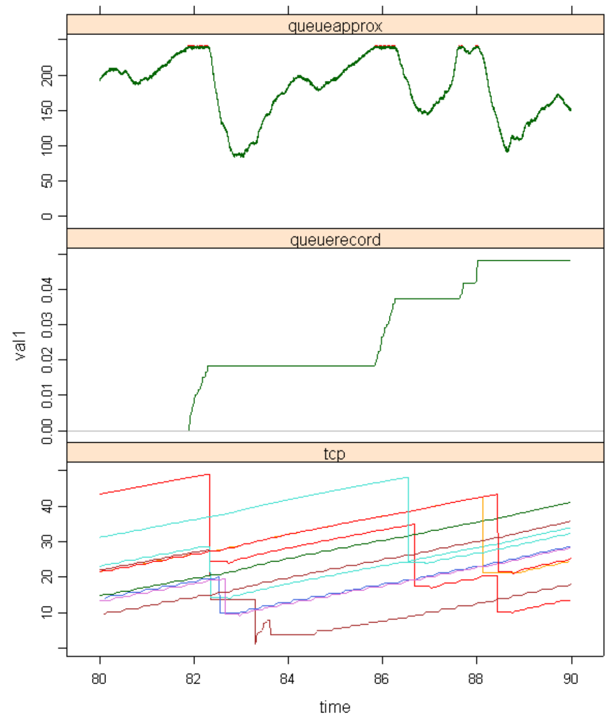
Standard TCP, single bottleneck link, no AQM
service C=1.2 kpkt/sec, RTT=200 ms, #flows N=20



B=20 pkts
(small buffer)



B=54 pkts
(intermediate buffer)

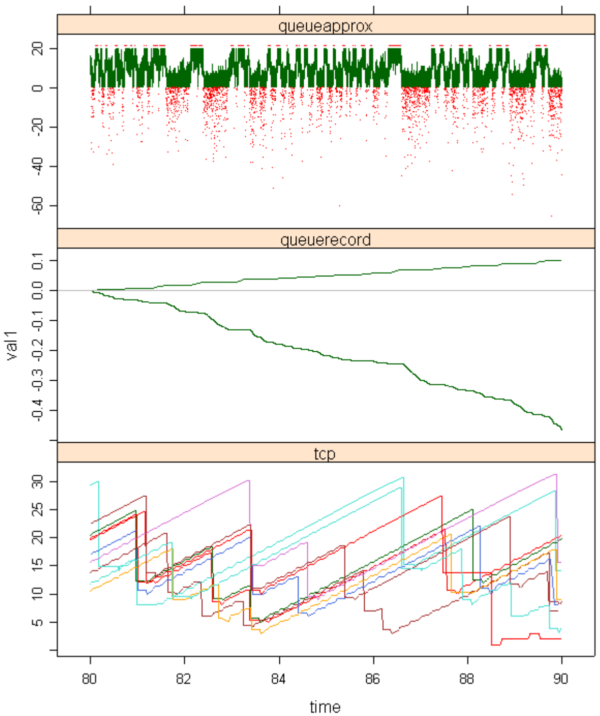


B=240 pkts
(large buffer)

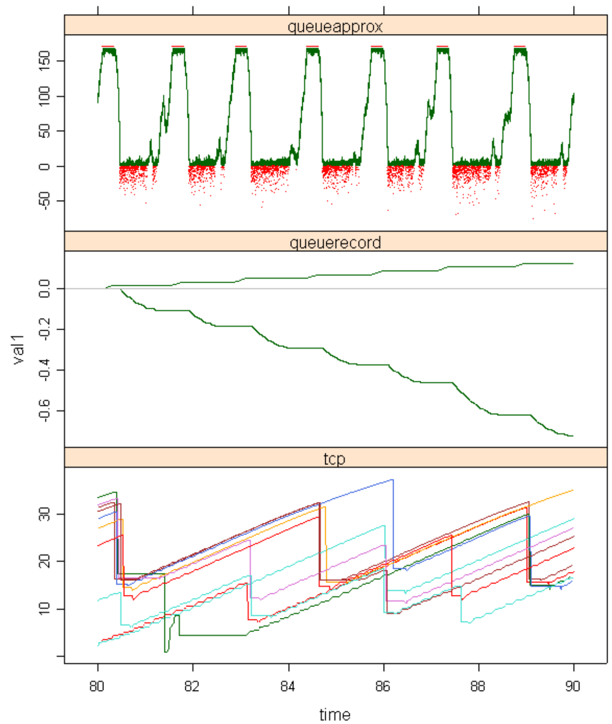
Illustration 200 flows

Standard TCP, single bottleneck link, no AQM

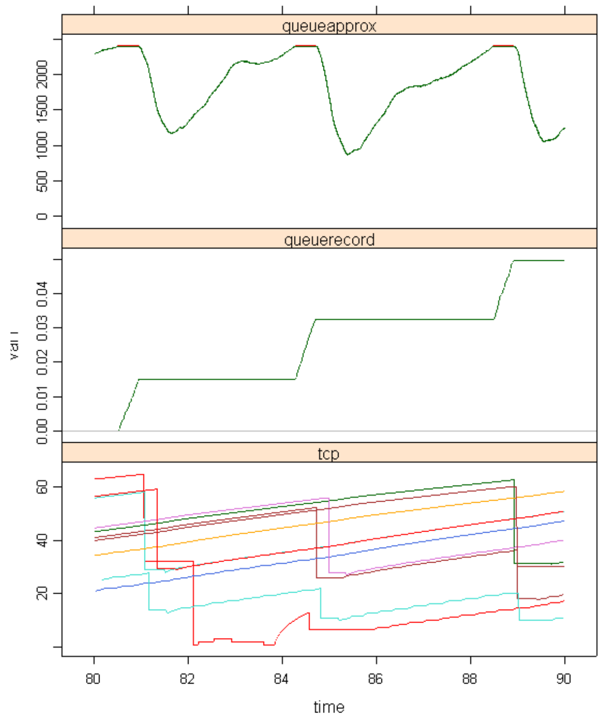
service C=12 kpkt/sec, RTT=200 ms, #flows N=200



B=20 pkts
(small buffer)



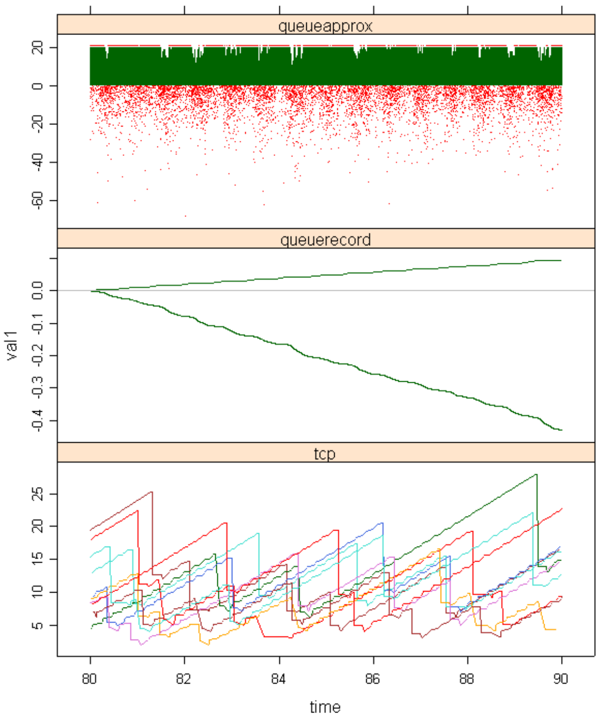
B=170 pkts
(intermediate buffer)



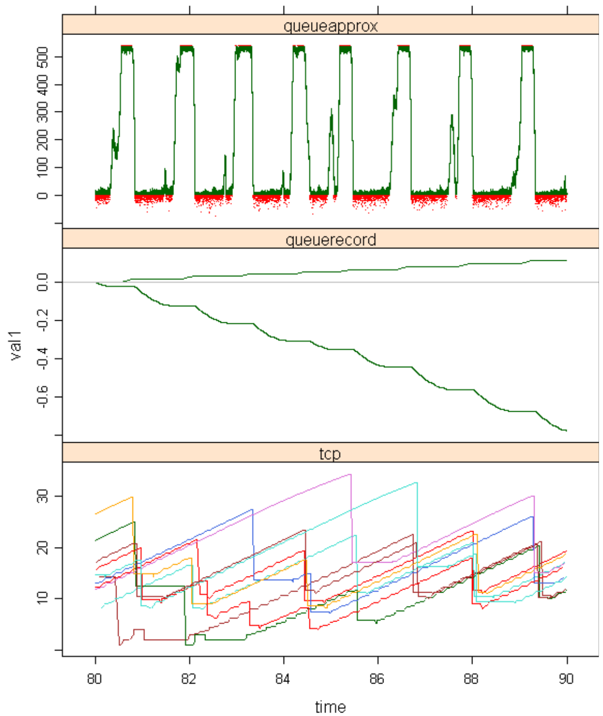
B=2,400 pkts
(large buffer)

Illustration 2000 flows

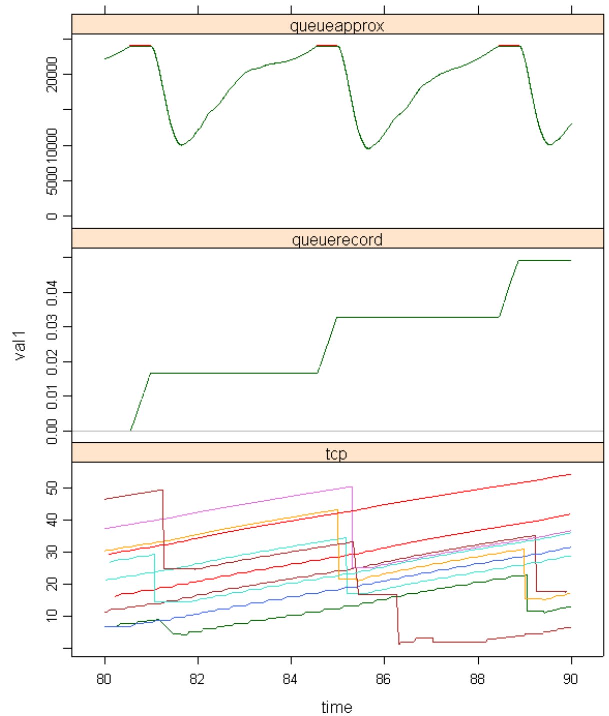
Standard TCP, single bottleneck link, no AQM
service *C*=120 kpkt/sec, *RTT*=200 ms, #flows *N*=2000



B=20 pkts
(small buffer)

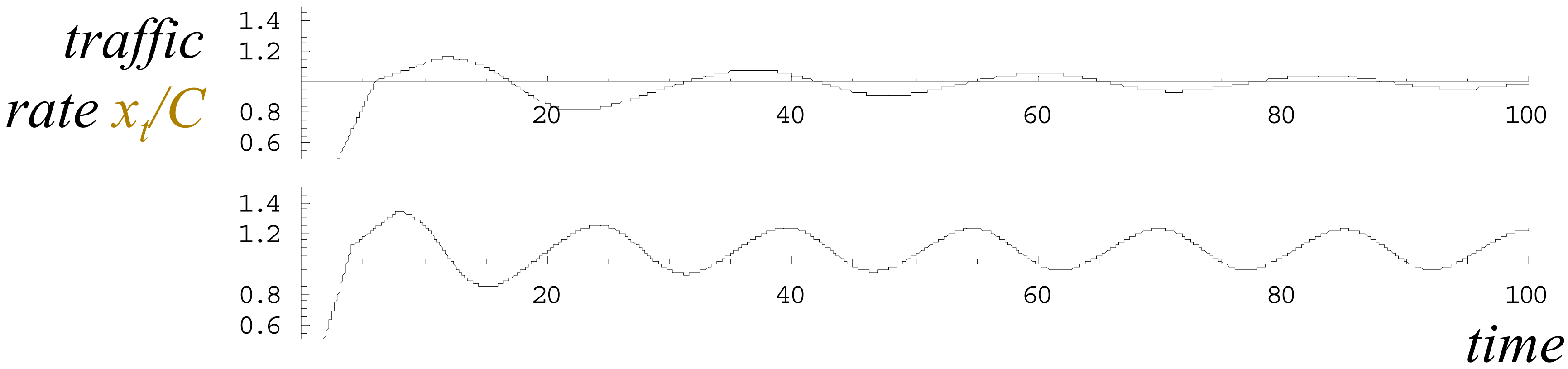


B=537 pkts
(intermediate buffer)



B=24,000 pkts
(large buffer)

Stability/instability analysis

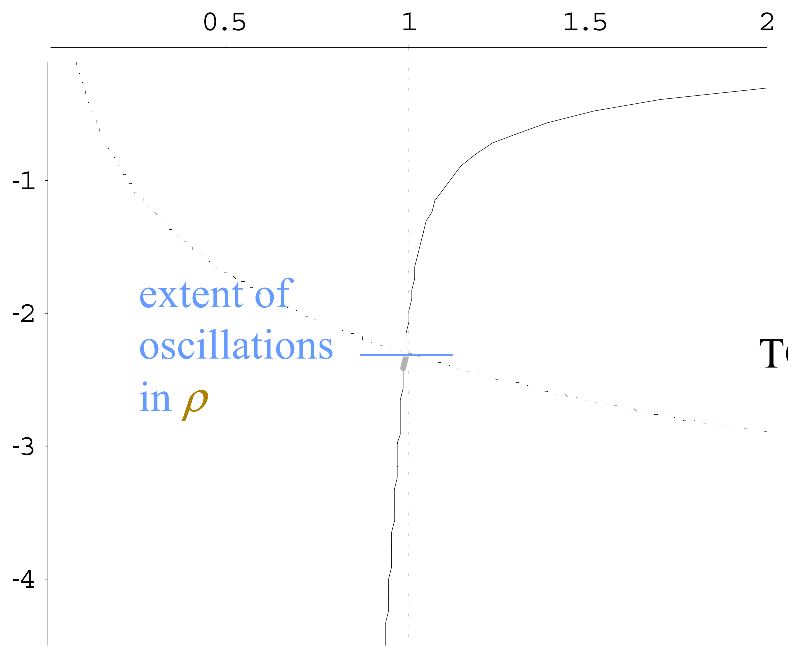


- For some values of *C*RTT*, the dynamical system is stable
  - we calculate the steady-state traffic rate, loss probability etc.
- For others it is unstable and there are oscillations (i.e. the flows are partially synchronized)
  - we calculate the amplitude of the oscillations [Gaurav Raina, PhD thesis, 2005]

Instability plot small-buffer case

traffic intensity $\rho = x/C$

$\log_{10}$ of pkt loss probability p



extent of oscillations in ρ

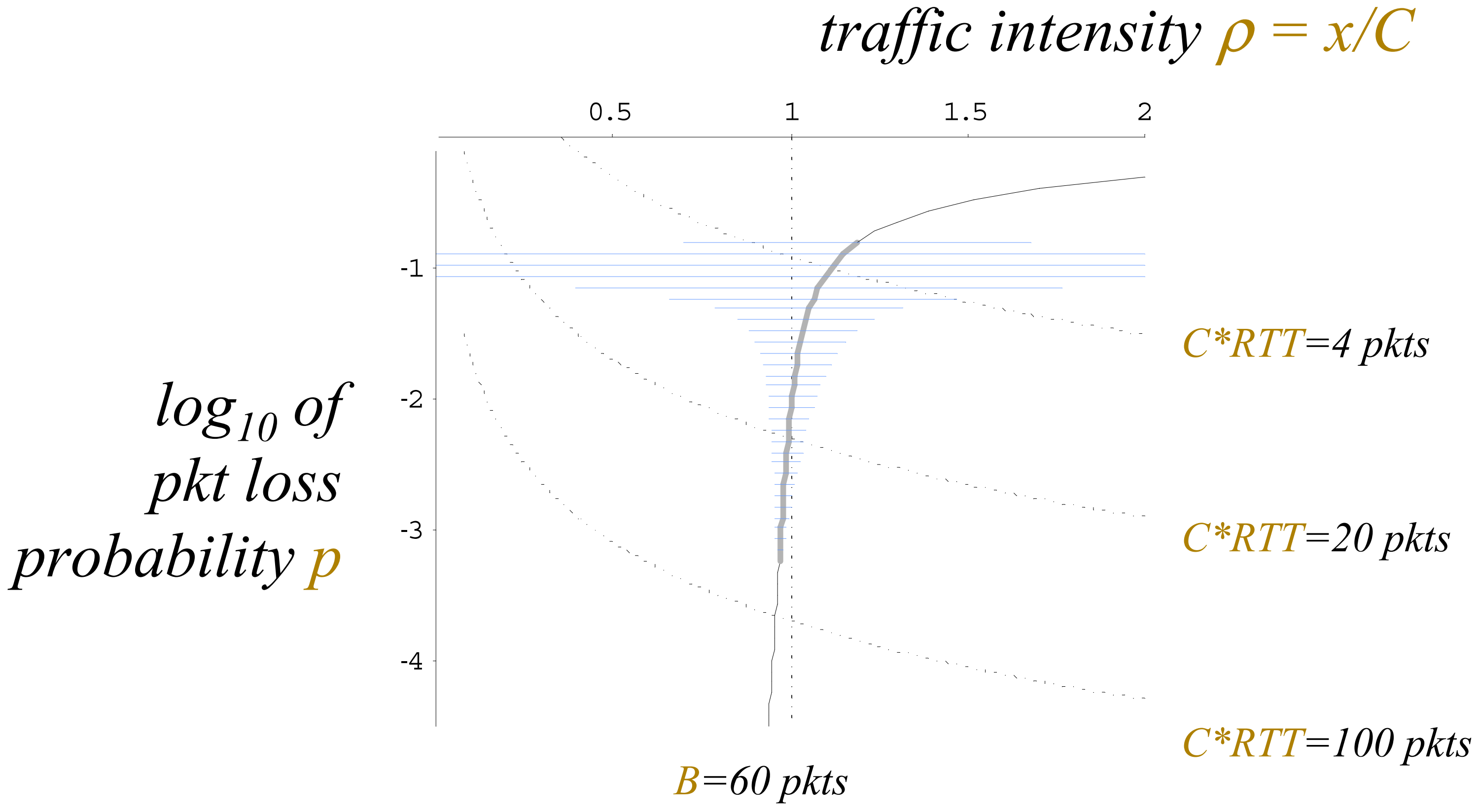
TCP throughput equation

$$x/C = \frac{\sqrt{2}}{C\,RTT\sqrt{p}}$$

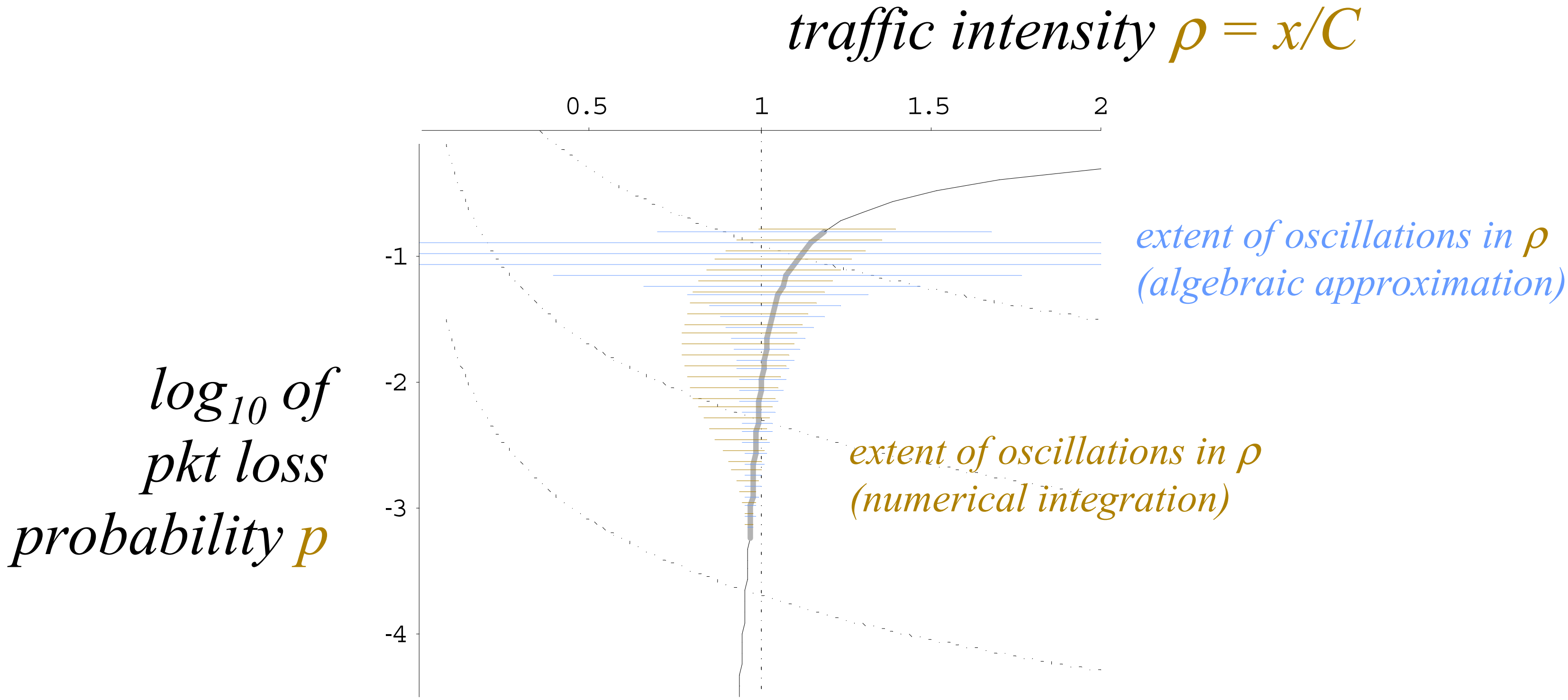
queue equation

$$p = \rho^B$$

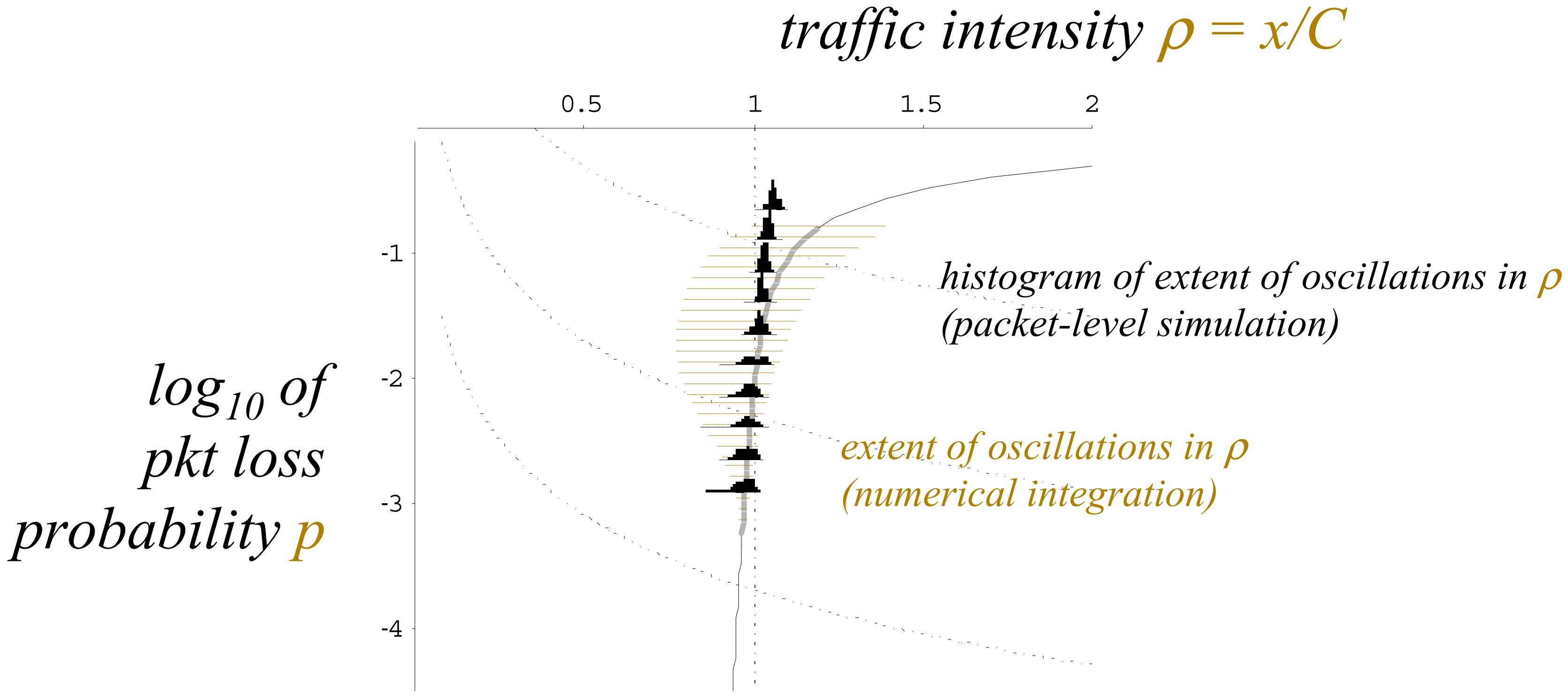
Instability plot small-buffer case



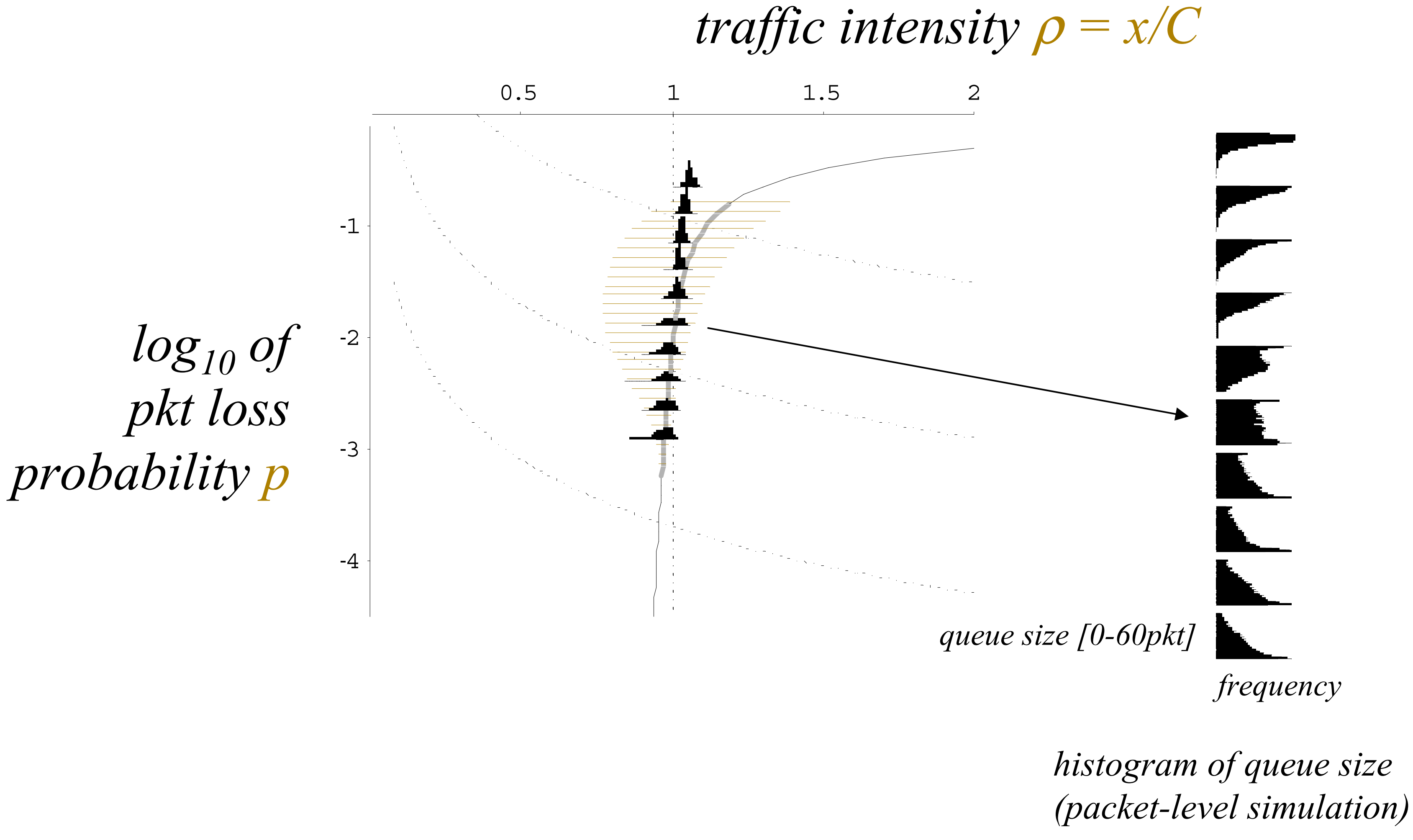
Instability plot small-buffer case



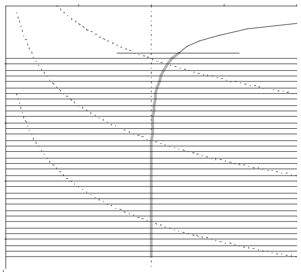
Instability plot small-buffer case



Instability plot small-buffer case



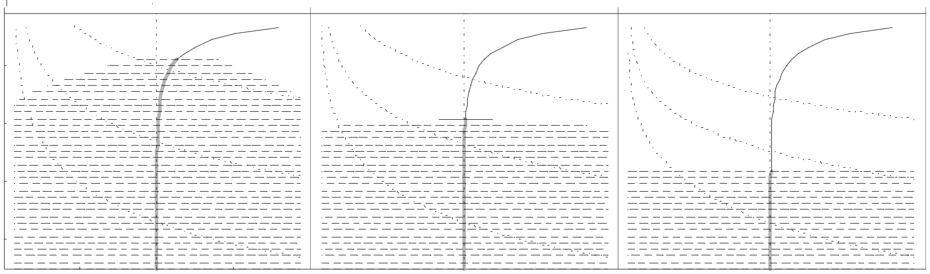
Alternative buffer-sizing rules



Intermediate buffers buffer *buffer = C*RTT*√N*

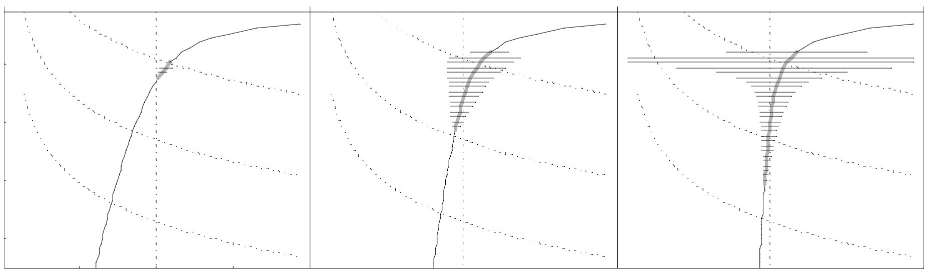
or

Large buffers *buffer = C*RTT*N*



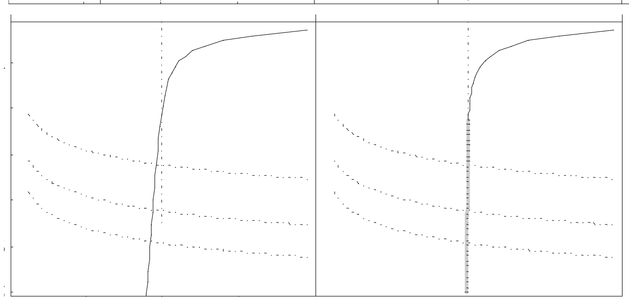
Large buffers with AQM

*buffer = C*RTT*N *{¼,1,4}*



Small buffers

buffer = {10,20,50} pkts



Small buffers, ScalableTCP

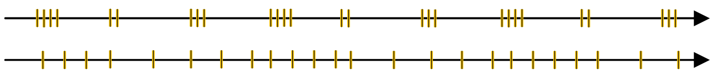
buffer = {50,1000} pkts

[Vinnicombe 2002, T.Kelly 2002]

Limitations/concerns

- Surely bottlenecks are at the access network, not the core network?
  - Unwise to rely on this!
  - If the core is underutilized, it definitely doesn’t need big buffers
  - The small-buffer theory works fine for as few as 20 flows

- The Poisson model sometimes breaks down
  - because of short-timescale packet clumps
  - need more measurement of short-timescale Internet traffic statistics



- Limited validation so far
  [McKeown et al. at Stanford, Level3, Internet2]
- Proper validation needs
  - goodly amount of traffic
  - full measurement kit
  - ability to control buffer size

Conclusion

- Buffer sizes can be very small
  - a buffer of 25pkt gives link utilization > 90%
  - small buffers mean that TCP flows get more regular feedback, so they can better judge how much capacity is available
  - use Poisson traffic models for the router, differential equation models for aggregate traffic

- TCP can be improved with simple changes
  - e.g. space out the packets
  - e.g. modify the window increase/decrease rules [ScalableTCP: Vinnicombe 2004, Kelly 2004; XCP: Katabi, Handley, Rohrs 2000]
  - any future transport protocol should be designed along these lines
  - improved TCP may find its way into Linux/Windows within 5 years