What are buffers for?

Frank Kelly

Workshop on buffer sizing
Stanford, 2-3 December 2019

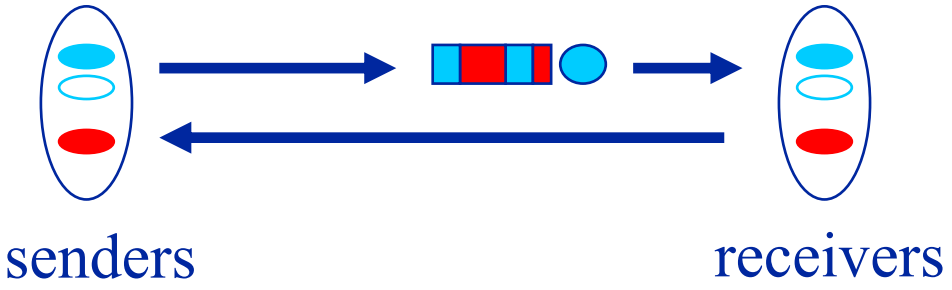
www.statslab.cam.ac.uk/~frank/TALKS

Summary

- TCP, bandwidth-delay product,…
  - interaction of congestion control and buffer sizes
  - what issues are unavoidable, and not a consequence of algorithm choices?
- Scalings for queueing delay
- Stability under propagation delays
- Fairness

TCP, bandwidth-delay product

Senders learn (through feedback from receivers) of congestion at queue, and slow down or speed up accordingly.



Jacobson's congestion avoidance algorithm attempts: to prevent too many packets entering the network by using window flow control; and to keep the bottleneck resource fully utilized using AIMD.

This requires a buffer at the bottleneck resource of the bandwidth-delay product for a single flow or for synchronized flows. With multiple non-synchronized flows a smaller buffer suffices (Appenzeller, Keslassy and McKeown 2019). But…..

TCP, macroscopic behaviour

The AIMD algorithm produces a flow rate proportional to

$$1/(T\sqrt{p})$$

T = round-trip time, p = packet drop probability. (Jacobson 1988, Mathis, Semke, Mahdavi, Ott 1997, Padhye, Firoiu, Towsley, Kurose 1998, Floyd and Fall 1999).

Consequences?

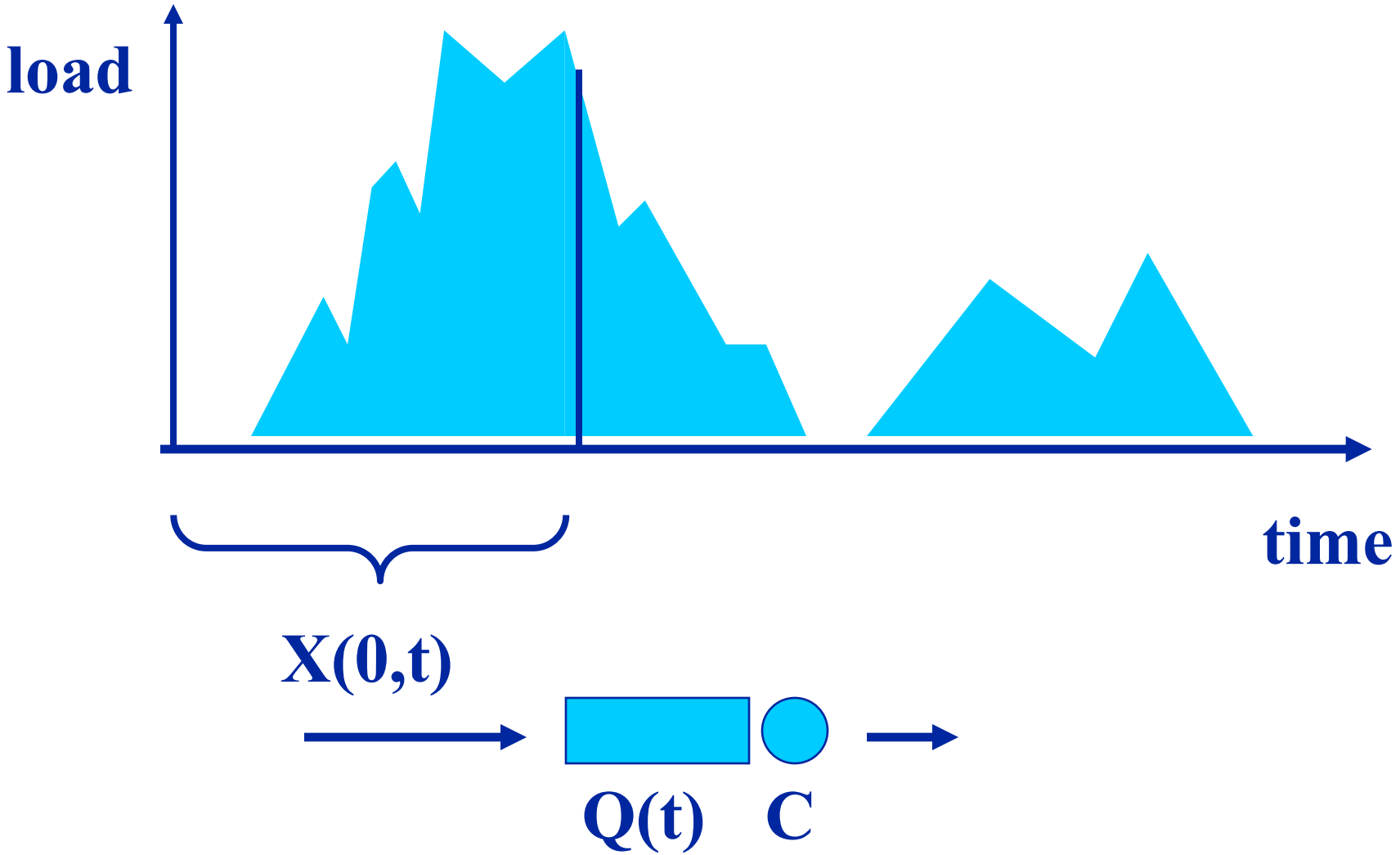
- Halving T will cause p to quadruple for the same flow rate. (Also the congestion window will halve.)
- For the flow rate to grow unboundedly, p must approach zero. (And for this buffer sizes must grow.)

Extracts from Jacobson 1988..

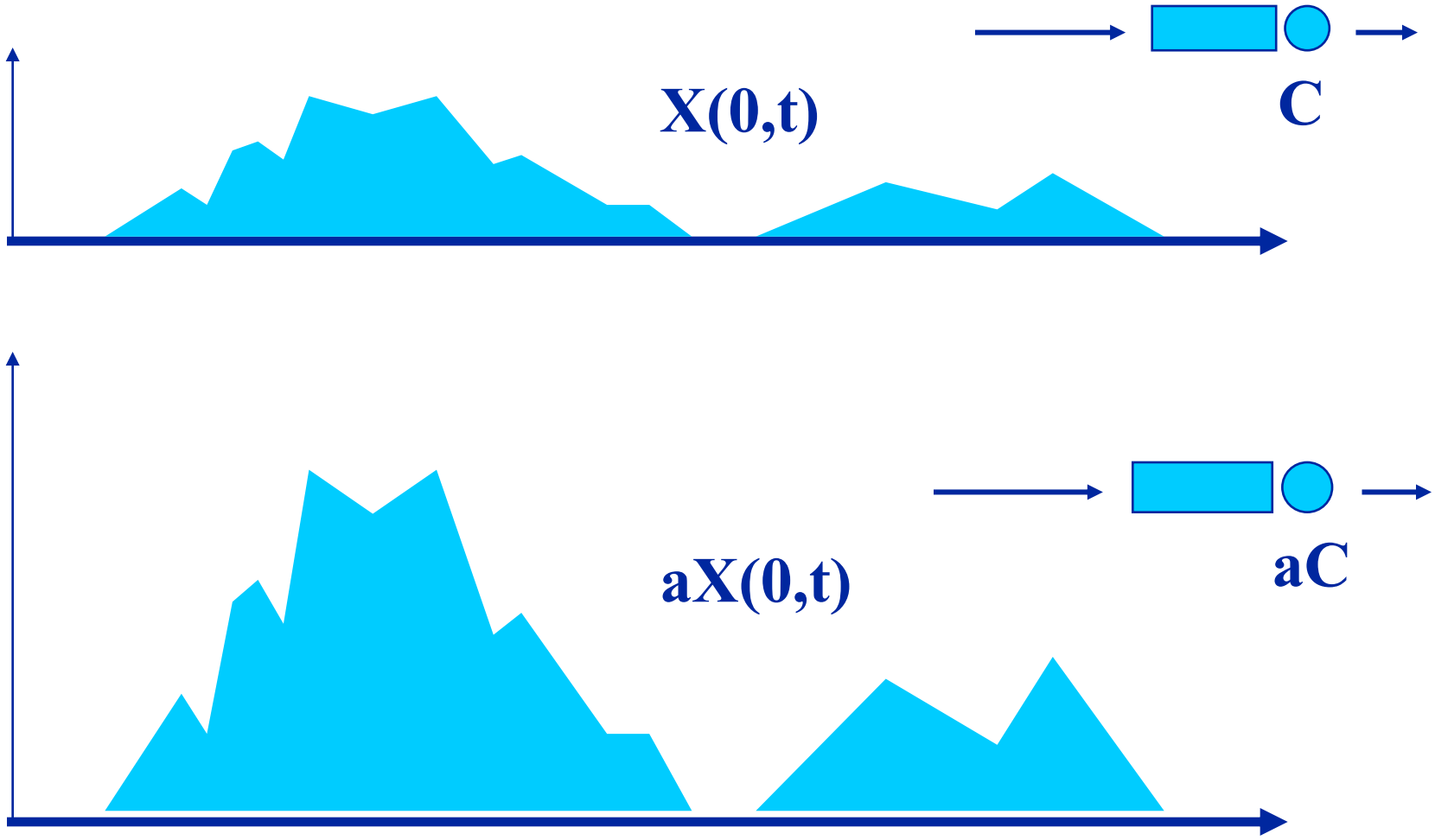
"Trace data of the start of a TCP conversation between two Sun 3/50s...connected by IP gateways driving a 230.4 Kbps point-to-point link... The window size for the connection was 16KB (32 512-byte packets) and there were 30 packets of buffer available at the bottleneck gateway. The actual path contains six store-and-forward hops..." {ed note: from LBL to UCB}

"We are concerned that the congestion control noise sensitivity is quadratic in *w* but it will take at least another generation of network evolution to reach window sizes where this will be significant. If experience shows this sensitivity to be a liability, a trivial modification to the algorithm makes it linear in *w*."

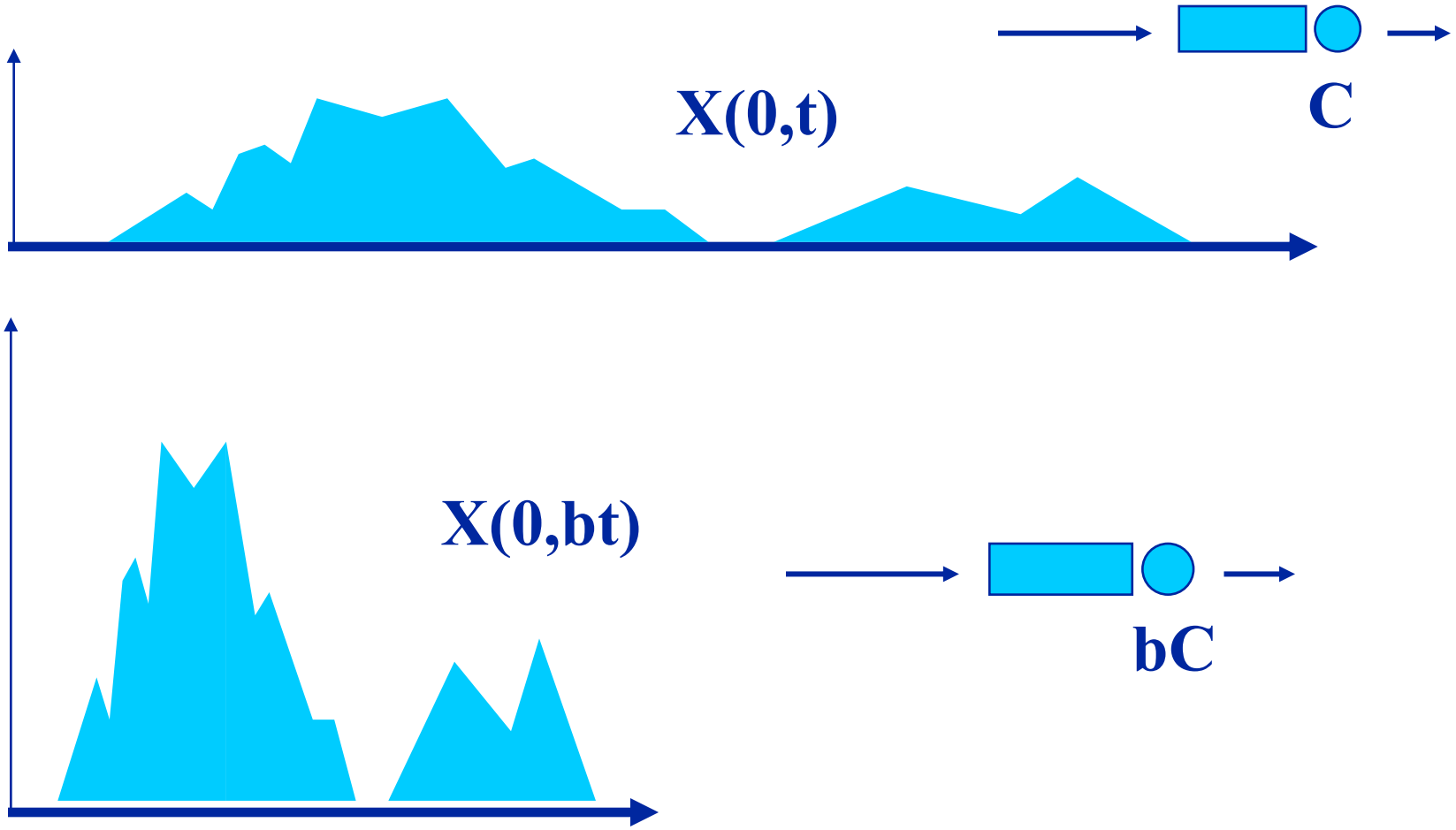
Workload arriving at a queue



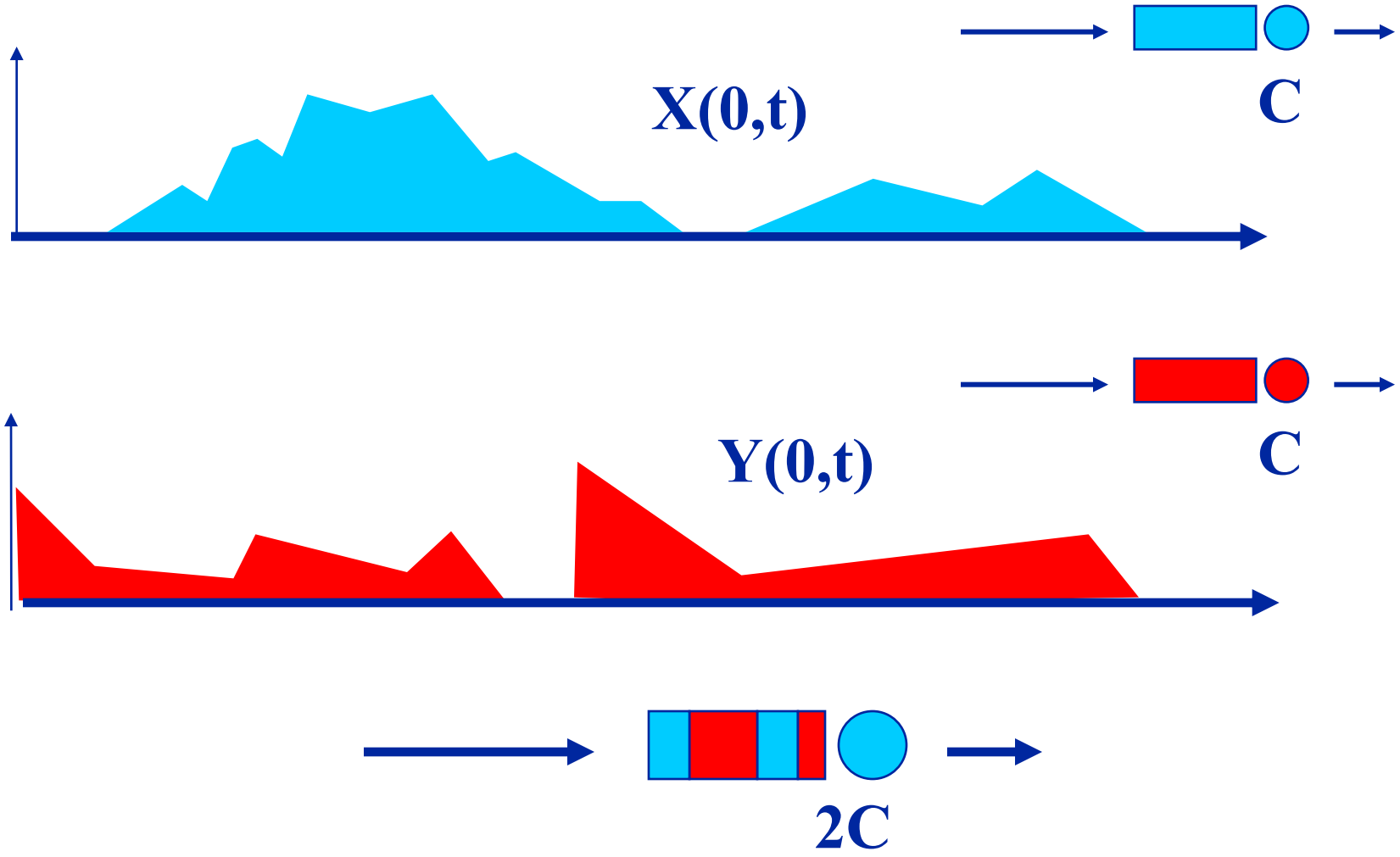
Volume scaling



Speed scaling



Multiplex scaling



Scalings for queueing delay

Let $X[s,t]$ be the amount of work that arrives at a queue in the time-interval $[s,t]$. If the queue has a service rate C and no work is lost, then the amount of work buffered in the queue at time 0 is

$$Q = \sup_{\{u \geq 0\}} \{ X[-u, 0] - Cu \}$$

(assume that the service rate C is adequate, so that the sup is finite and Q is defined as a proper random variable)

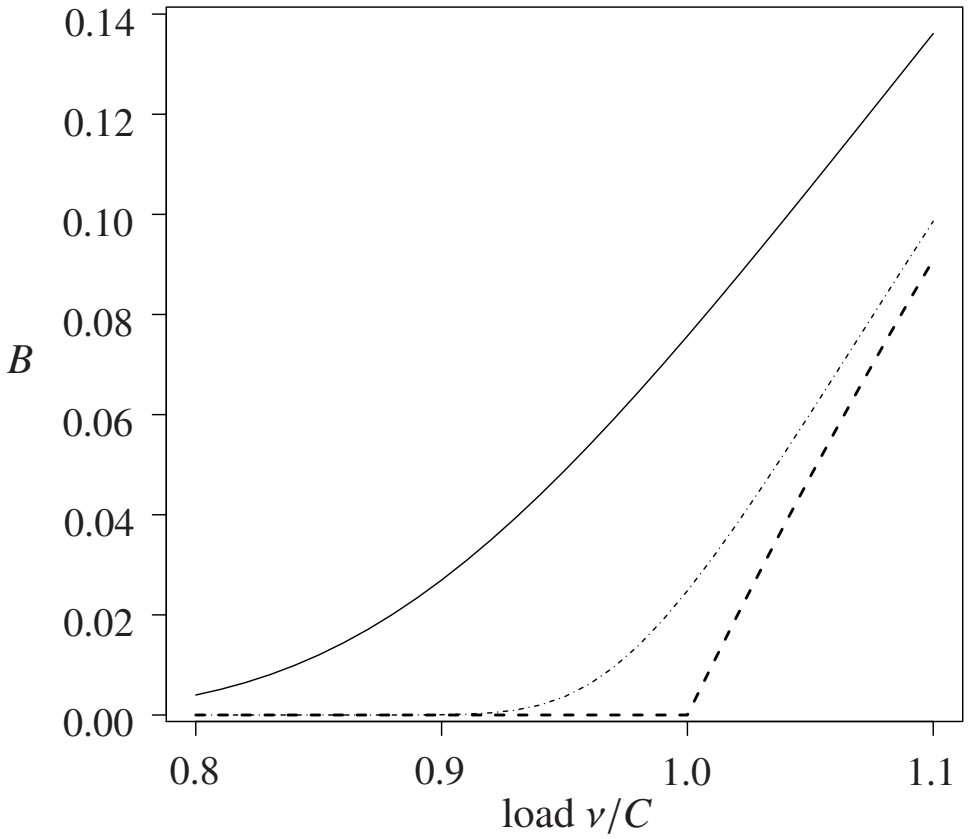
Now suppose that we replace $X[s,t]$ by $\sum_{i=1}^{c} a\, X_i[bs, bt]$ and C by $abcC$ so that queueing delay is

$$\tau(a,b,c) = \frac{Q(a,b,c)}{abcC}$$

Effect on queueing delays? (K 2000)

- Volume scaling by factor a leaves queueing delay unaltered
- Speed scaling by factor b reduces queueing delay by factor b
- Multiplex scaling by factor c reduces queueing delay by factor $c^{1/(2-2H)}$ where H is the Hurst parameter (short range order, $H=1/2$, gives a factor c and larger values, corresponding to long range order, give even larger factors)
- Conclusion? Volume scaling does not impact on queueing delay, but speed and multiplex scaling both cause substantial reductions in queueing delay

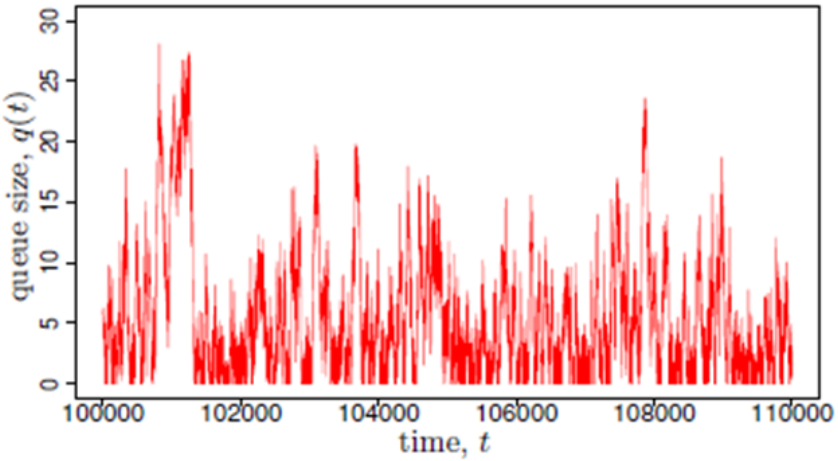
Resource pooling



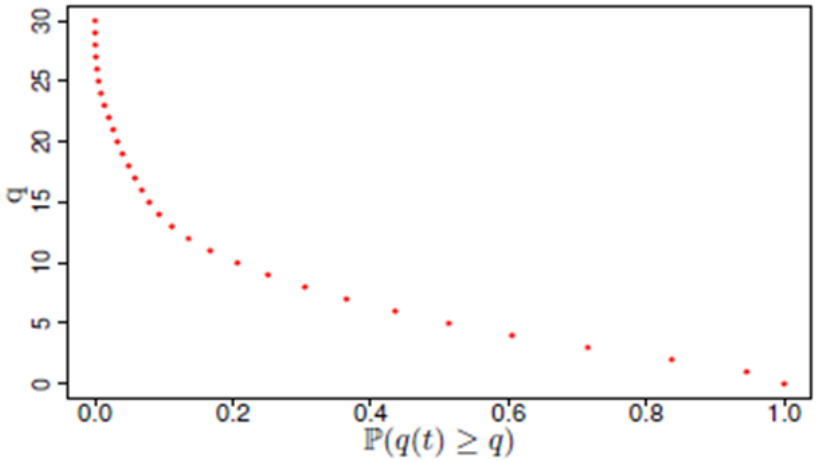
Resource pooling increases efficiency, but onset of congestion becomes sudden

Figure 3.14 Erlang blocking probability for $C = 100$ (solid), $C = 1000$ (dot-dashed) and the limit as $C \to \infty$ (dashed).

Fast fluctuations of queue size



(a) Evolution of the queue over a single round-trip time.

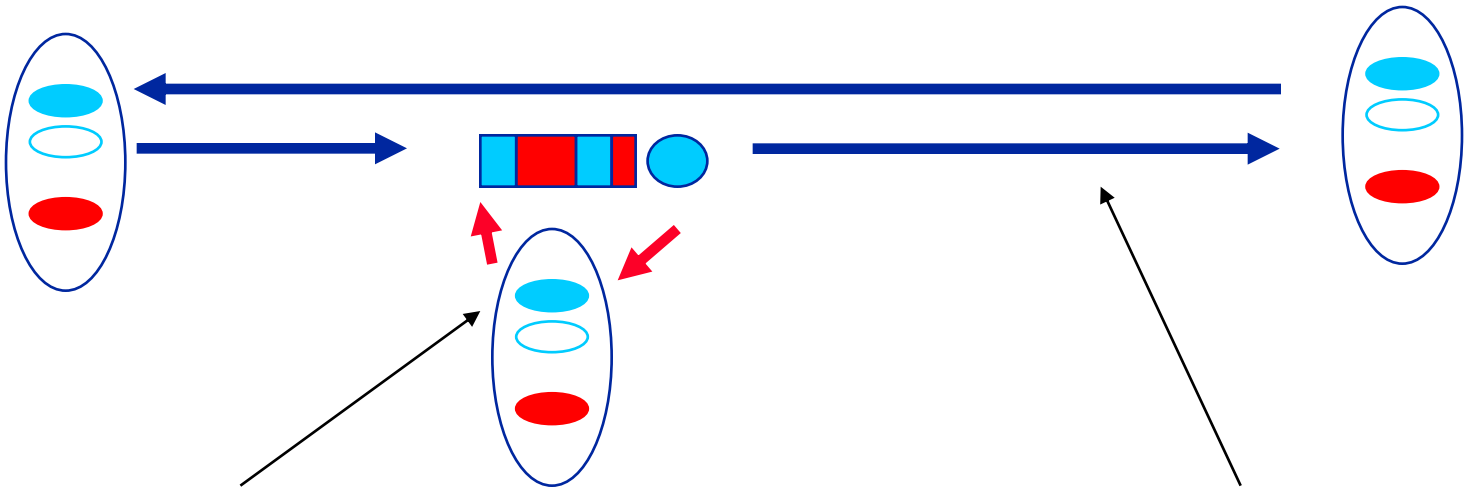


(b) Empirical distribution of queue size within one round-trip time.

(100 sources, RTT=10000 - e.g. with packet size 1000 bytes, then 1 Gbyte/sec and RTT=10ms – from K, Raina and Voice 2008)

With small buffers and large rates the queue size fluctuations are very fast – so fast that it is impossible to control the queue size. Instead protocols act to control the *distribution* of queue size.

Different RTTs



Short round trip times, so quick response to feedback

Long round trip times, so slow response to feedback

Recap

If queueing delays become small, then round-trip times will be dominated by propagation delays

Propagation delays may vary by many orders of magnitude in a network

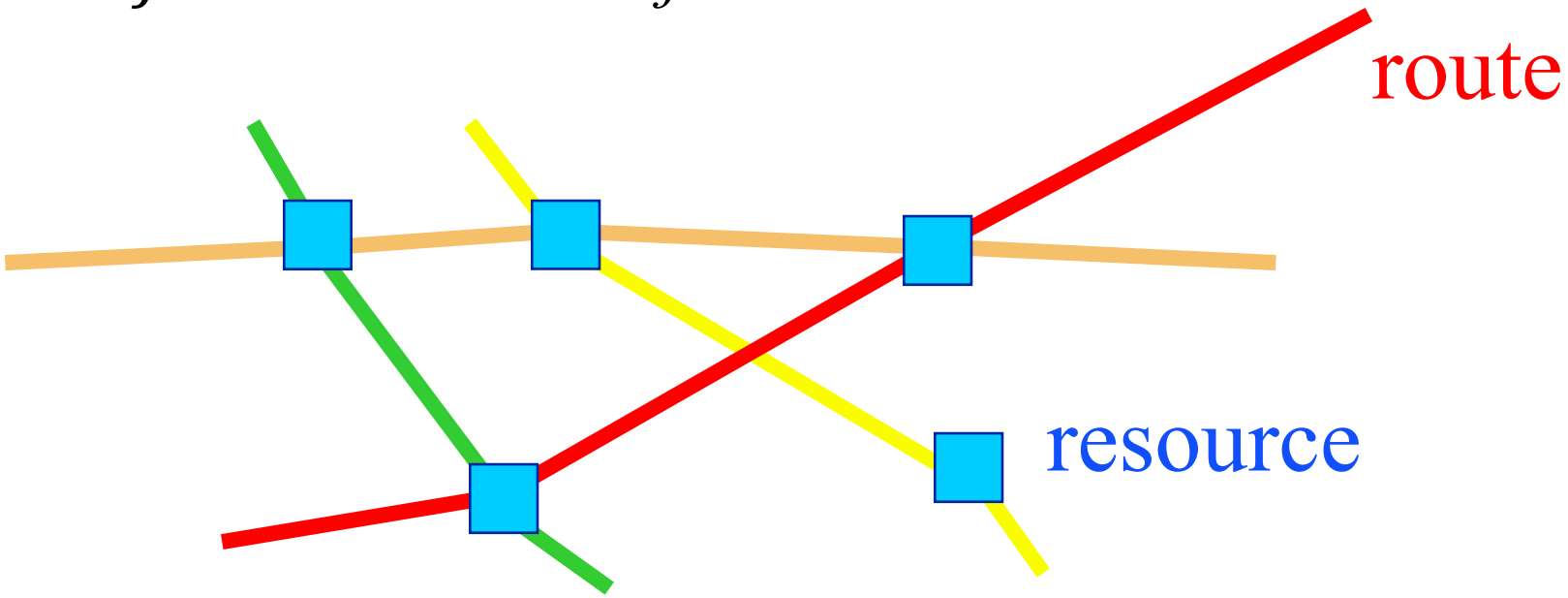
Stability of an equilibrium point then becomes a challenge (and synchronization of flows, route flap, etc are all symptoms of loss of stability of an equilibrium point)

But there is a well-developed theory of network stability under arbitrary network topologies and arbitrary propagation delays

Network abstraction

$J =$ set of resources $R =$ set of routes

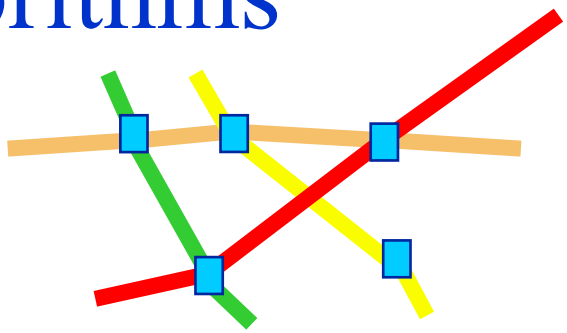
$j \in r$ if resource j is on route r



$x_r(t)$ - flow rate on route r at time t

$\mu_j(t)$ - shadow price, or rate of congestion indication, at resource j at time t

Delayed differential equation model of primal (TCP-like) algorithms

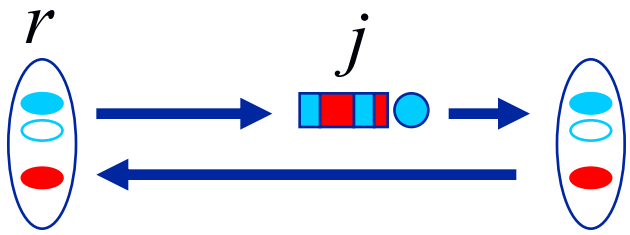


$$\frac{d}{dt}x_r(t) = \frac{x_r(t-T_r)}{T_r}$$

$$\Big(a_r\,(x_r(t)T_r)^n\,(1-\lambda_r(t)) - b_r\,(x_r(t)T_r)^m\,\lambda_r(t)\Big)$$

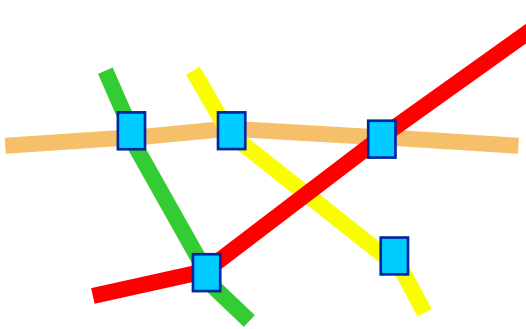
$$\lambda_r(t) = 1 - \prod_{j\in r}\Big(1-\mu_j(t-T_{jr})\Big)$$

$$\mu_j(t) = p_j\left(\sum_{s:\,j\in s} x_s(t-T_{sj})\right)$$



$$T_{rj} + T_{jr} = T_r$$

Delay stability



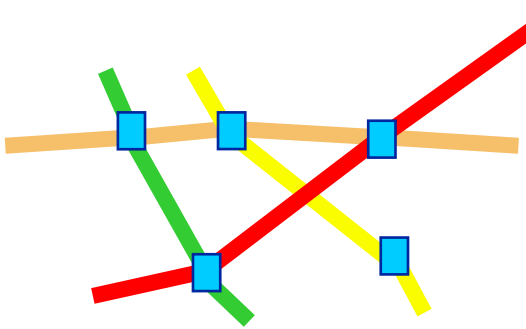
Johari and Tan (1999), Massoulié (2000), Vinnicombe (2000), Paganini, Doyle and Low (2001)
- equilibrium is stable if there exists a global constant β such that

$$x p_j'(x) < \beta\, p_j(x), \qquad a_r (x_r T_r)^n < \frac{\pi}{2\beta}$$

condition on sensitivity for each resource j

condition on aggressiveness for each route r

Delay stability



Johari and Tan (1999), Massoulié (2000), Vinnicombe (2000), Paganini, Doyle and Low (2001)
- equilibrium is stable if there exists a global constant β such that

$$x p_j'(x) < \beta\, p_j(x), \qquad a_r (x_r T_r)^n < \frac{\pi}{2\beta}$$

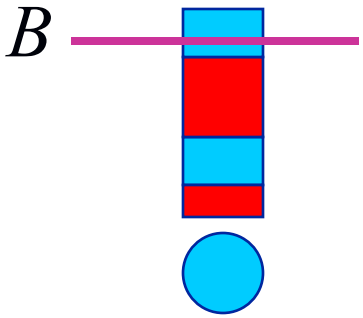
condition on sensitivity for each resource j

condition on aggressiveness for each route r

But parameter β must be chosen, and feedback noisy

Example: threshold marking

Mark an arriving packet if number of packets at queue is at least B.



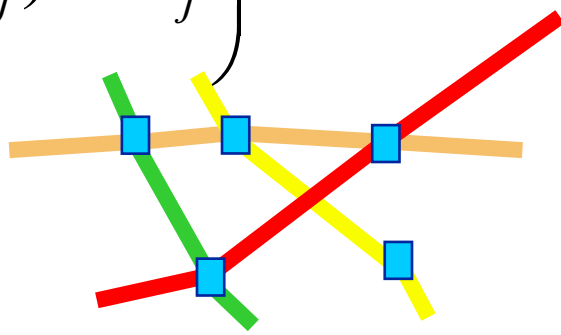
Then β should be B, everywhere.

i.e. the same parameter choice at every queue in the network, and at every source. (And choice at a queue impacts utilization; and at a source, aggressiveness.)

Fair dual algorithms (K 2003)

$$\frac{d}{dt}\mu_j(t) = \kappa_j\ \mu_j(t)\left(\sum_{s:j\in s} x_s(t-T_{sj}) - C_j\right)$$

$$x_r(t) = \frac{w_r}{\sum_{j\in r}\mu_j(t-T_{jr})}$$



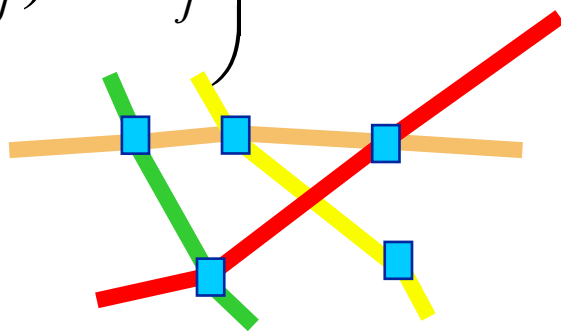
A sufficient condition for stability: $\kappa_j\ C_j\ \bar{T}_j < \frac{\pi}{2}$

$\bar{T}_j =$ average round-trip time of packets through resource j

Fair dual algorithms (K 2003)

$$\frac{d}{dt}\mu_j(t) = \kappa_j\ \mu_j(t)\left(\sum_{s:j\in s} x_s(t-T_{sj}) - C_j\right)$$

$$x_r(t) = \frac{w_r}{\sum_{j\in r}\mu_j(t-T_{jr})}$$

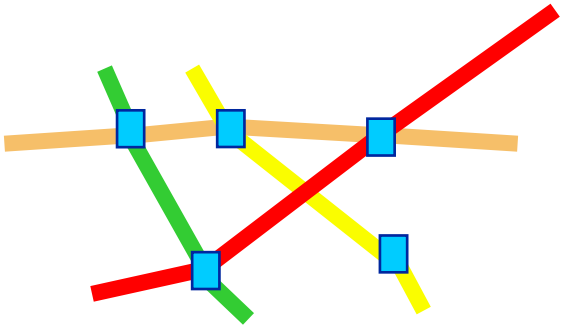


A sufficient condition for stability: $\kappa_j\ C_j\ \bar{T}_j < \frac{\pi}{2}$

No free parameters left!

$\bar{T}_j =$ average round-trip time of packets through resource j

Self-scaling algorithm



Utilization (as a proportion of capacity) used to drive algorithm.
Multiplicative change in shadow price μ_j .
Gain at resource inverse to average RTT.
No free parameters.

Some intuition. Suppose a single resource, multiple sources, common RTTs and we update once per RTT. Discrete time equivalent of $\frac{\pi}{2}$ is 1 and we hit target utilization precisely in one RTT.

HPCC: High Precision Congestion Control

(Sigcomm 2019 – Li, Miao, Liu, Zhang et al)

- In-network telemetry allows precise link load information to be fed back
- This allows fast convergence to high (but *not* 100%) utilization for longer flows
- Rate pacing for ultra-low latency, but window limits to constrain packets in network
- Buffer size determined by magnitude of transient overloads (e.g. start of an incast); and queue size measurements essential for dealing with these transients