Multipath routing and congestion control

Frank Kelly
University of Cambridge

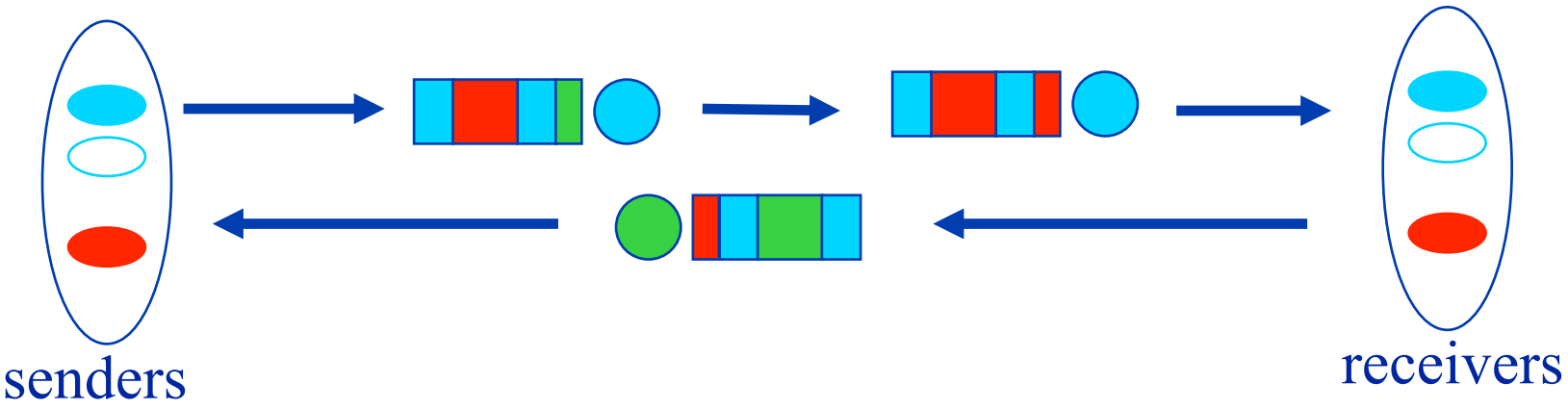
(with work of Damon Wischik, Mark Handley and Costin Raiciu, University College London)

Chinese University of Hong Kong
(Institute of Network Coding)
2 March 2012

Outline

- Congestion control - TCP
- Differential equation models, stability
- Multipath routing – from theory to practice
- UCL's working implementation

End-to-end congestion control

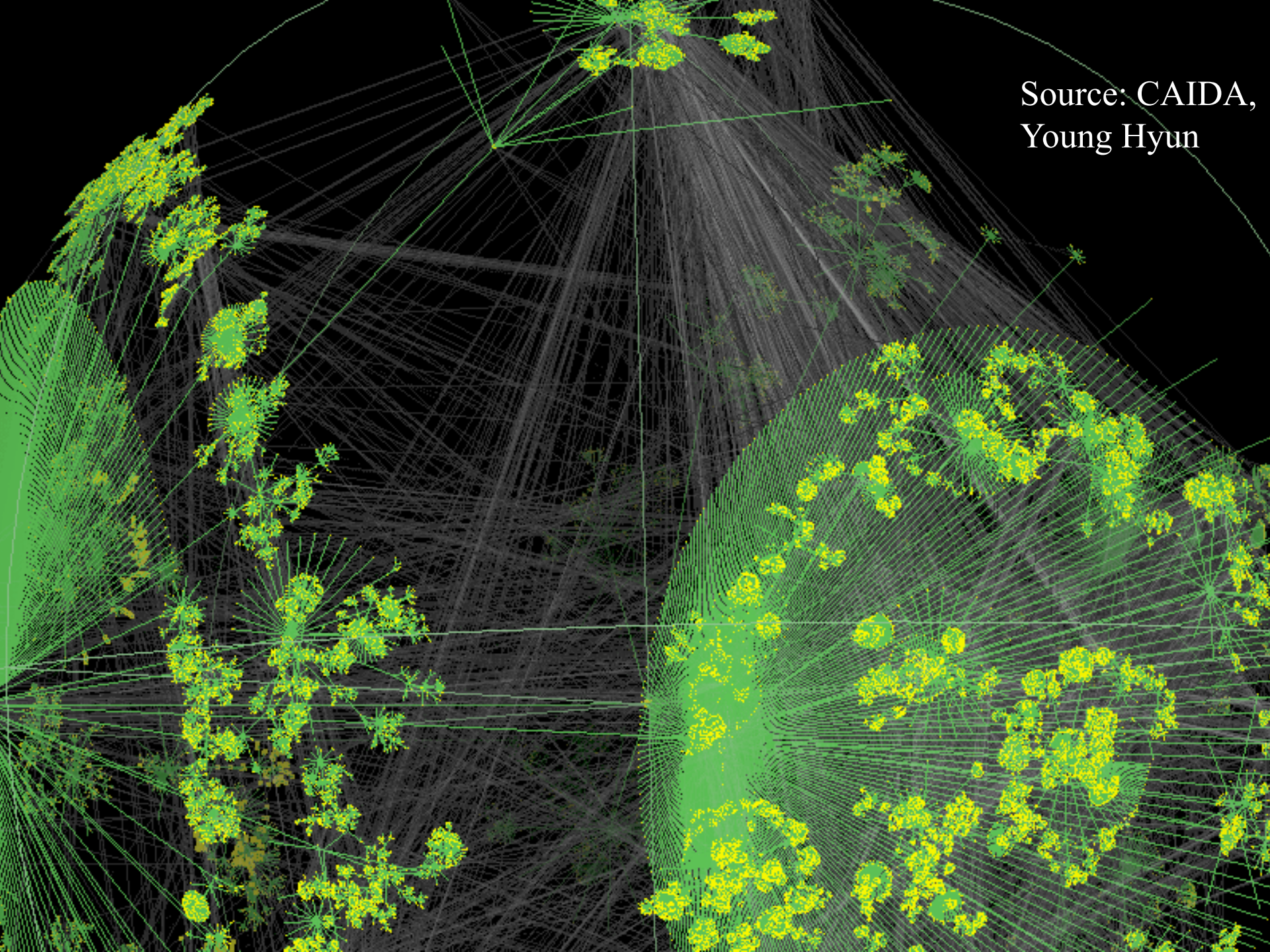


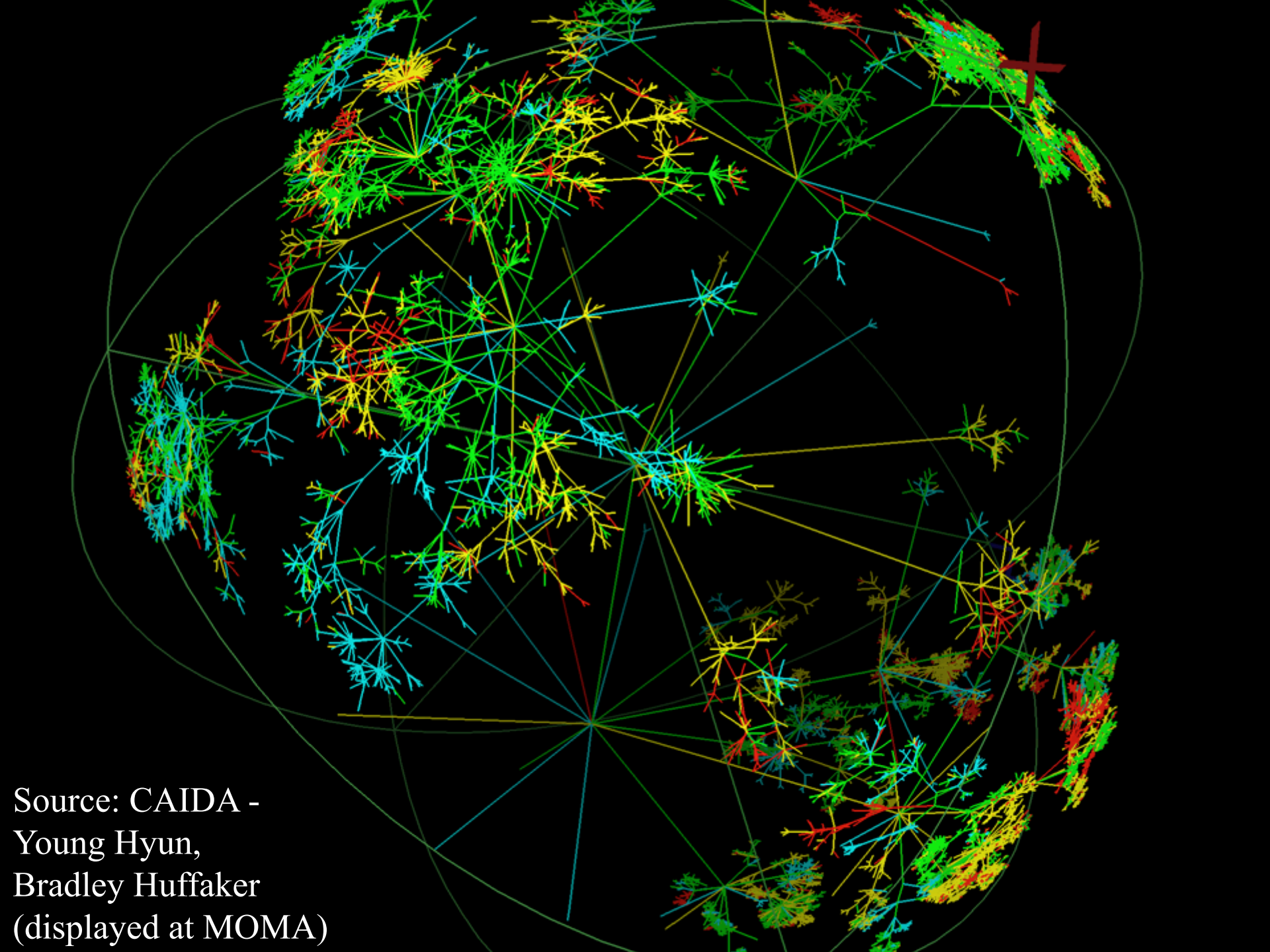
Senders learn (through feedback from receivers) of congestion at queue, and slow down or speed up accordingly. With current TCP, throughput of a flow is proportional to

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

T = round-trip time, p = packet drop probability.

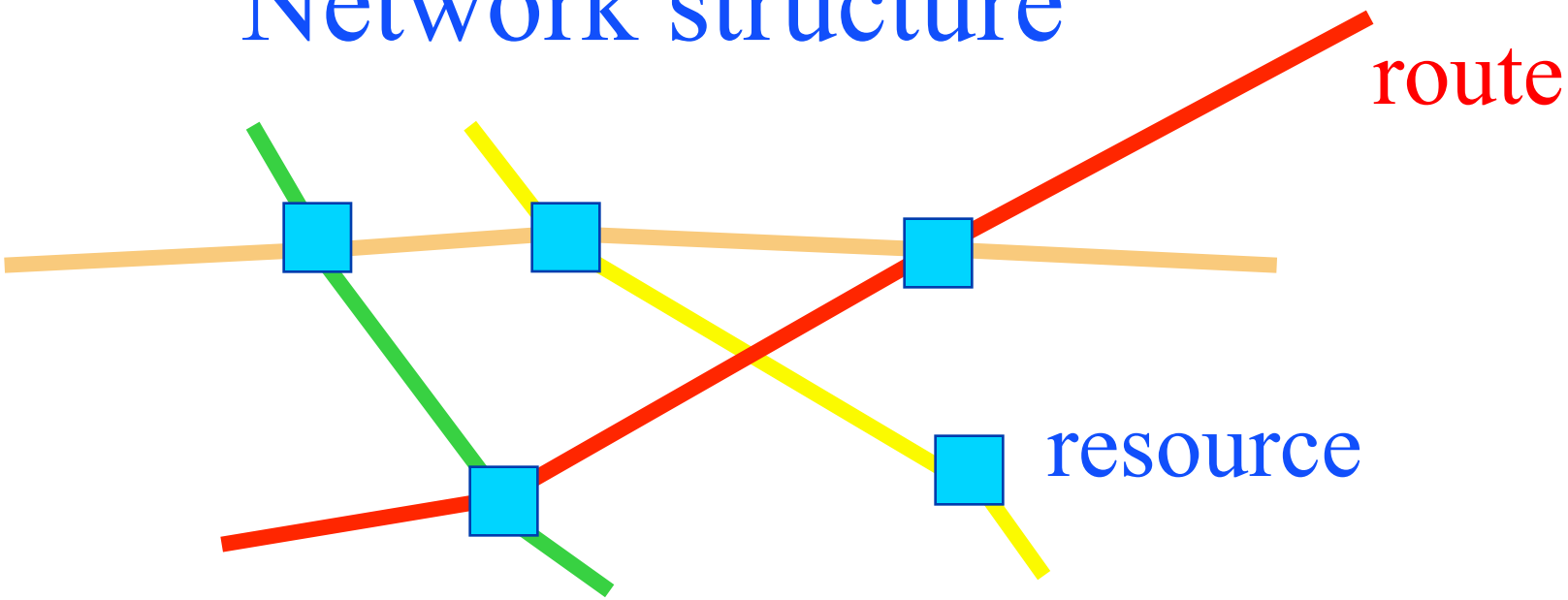
(Jacobson 1988, Mathis, Semke, Mahdavi, Ott 1997, Padhye, Firoiu, Towsley, Kurose 1998, Floyd and Fall 1999)


Source: CAIDA,
Young Hyun



Source: CAIDA -
Young Hyun,
Bradley Huffaker
(displayed at MOMA)

Network structure



J - set of resources

R - set of routes

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

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

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

TCP (Jacobson's congestion avoidance algorithm)

Source maintains a window of sent, but not yet acknowledged, packets - size *cwnd*

$$cwnd \approx xT$$

- *cwnd* incremented by *1/cwnd* on positive ack
- *cwnd* decremented by *cwnd/2* on congestion
- change in rate *x* per unit time is about

$$\frac{\left(\frac{1}{cwnd}(1-p)-\frac{cwnd}{2}p\right)/T}{T/cwnd}=\frac{1-p}{T^2}-\frac{x^2p}{2}$$

Differential equations for TCP

$$\frac{d}{dt}x_r(t) = \frac{1-\lambda_r(t)}{T_r^2} - \frac{x_r(t)^2\lambda_r(t)}{2}$$

$$\lambda_r(t) = 1 - \prod_{j \in r}\left(1-\mu_j(t)\right)$$

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

Equilibrium point

$$x_r = \frac{1}{T_r}\left(2\frac{1-\lambda_r}{\lambda_r}\right)^{1/2} \qquad r \in R$$

- check: the equilibrium of the dynamical system recovers the inverse square root formula for TCP
- note again the round-trip time bias

More general algorithm

On route *r*,

- *cwnd* incremented by $a_r\, cwnd^{\,n}$ on positive acknowledgement
- *cwnd* decremented by $b_r\, cwnd^{\,m}$ for each congestion indication $(m>n)$
- $a_r = 1,\ b_r = 1/2,\ m=1,\ n=-1$ corresponds to Jacobson's TCP

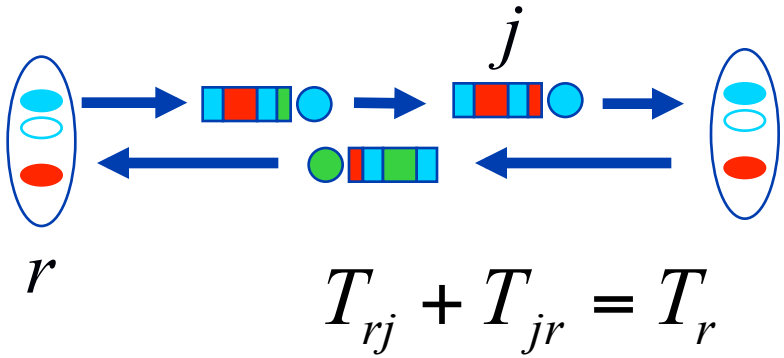
Differential equations with delays

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

$$\cdot\left(a_r \left(x_r(t)T_r\right)^n \left(1 - \lambda_r(t)\right) - b_r \left(x_r(t)T_r\right)^m \lambda_r(t)\right)$$

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

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

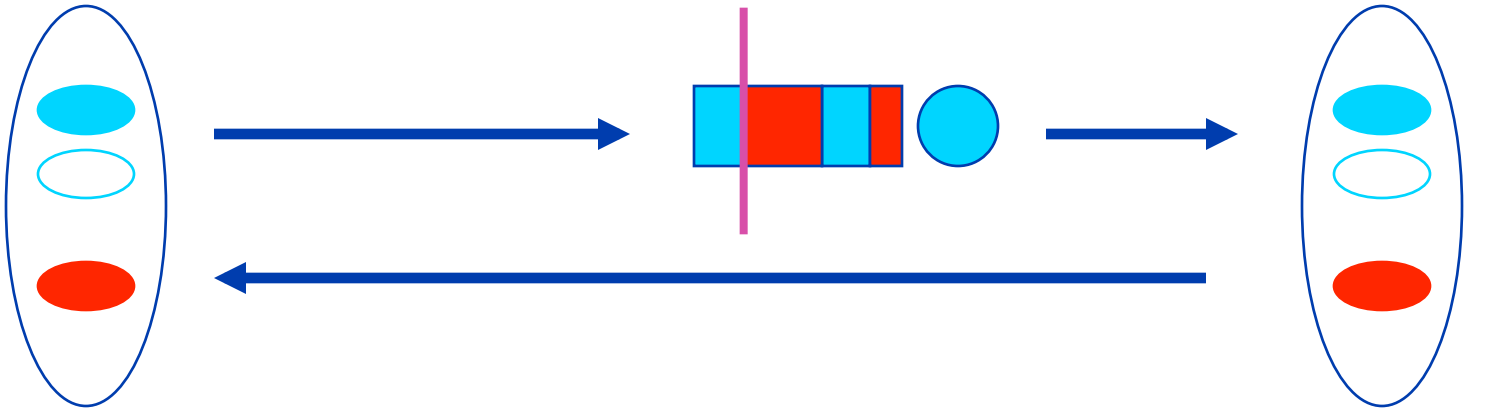


Equilibrium point

$$x_r = \frac{1}{T_r}\left(\frac{a_r}{b_r}\frac{1-\lambda_r}{\lambda_r}\right)^{1/m-n} \qquad r \in R$$

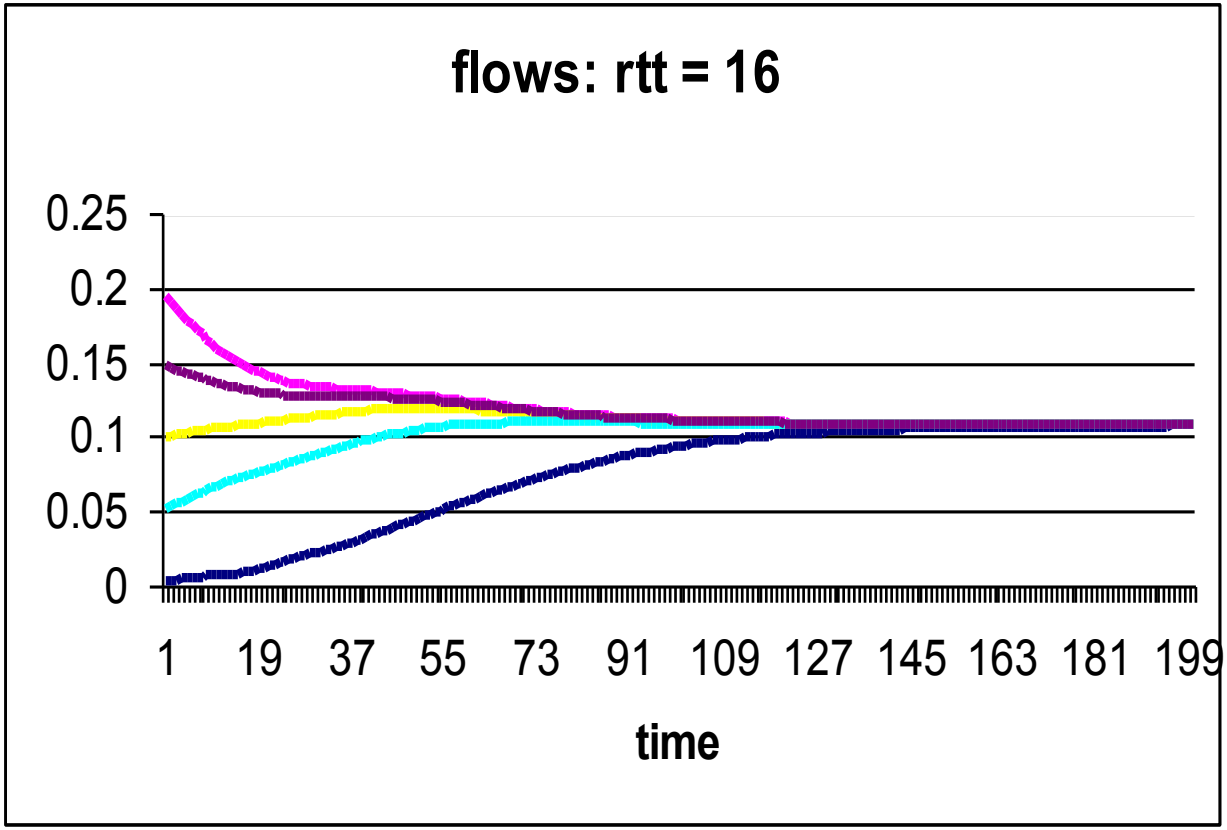
- choice of a_r, b_r can remove round trip time bias
- but… what is the impact of delays on stability?

Delays and stability

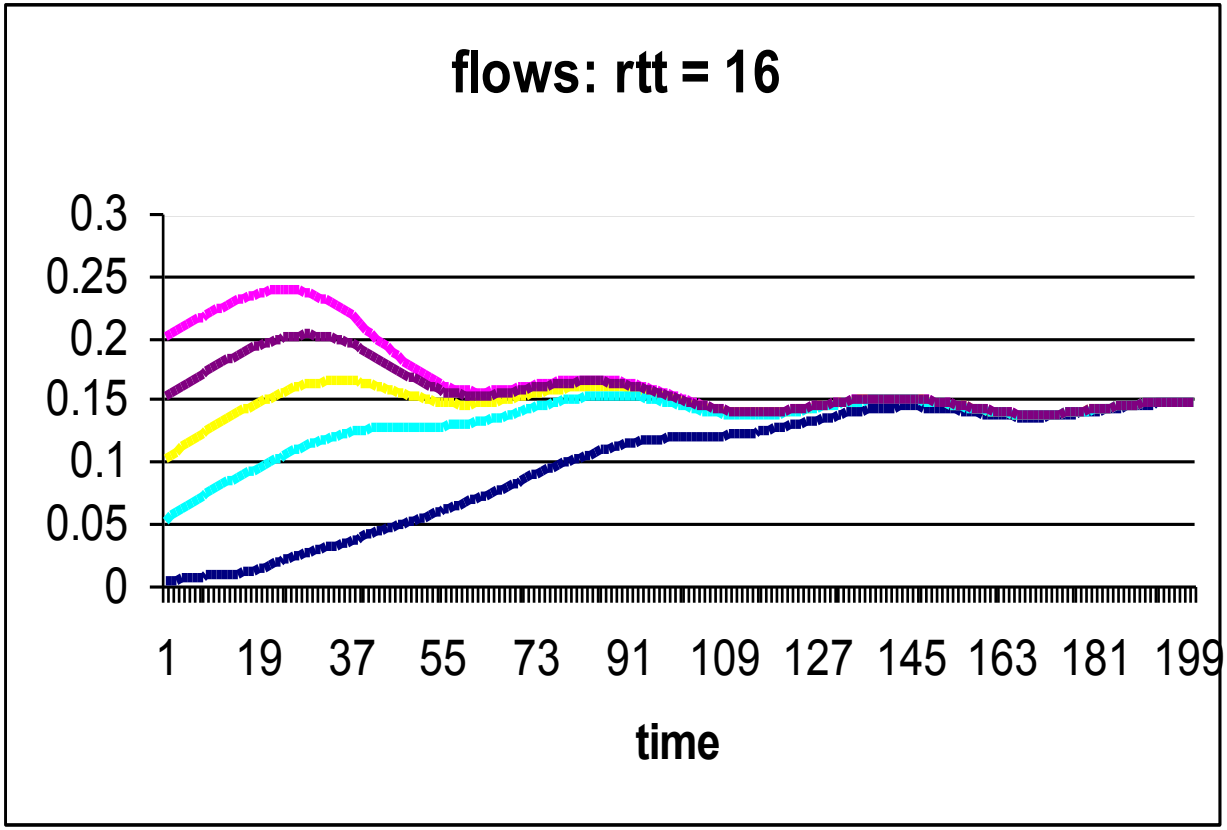


Might the inevitable delay
associated with the round-trip time
induce instability?

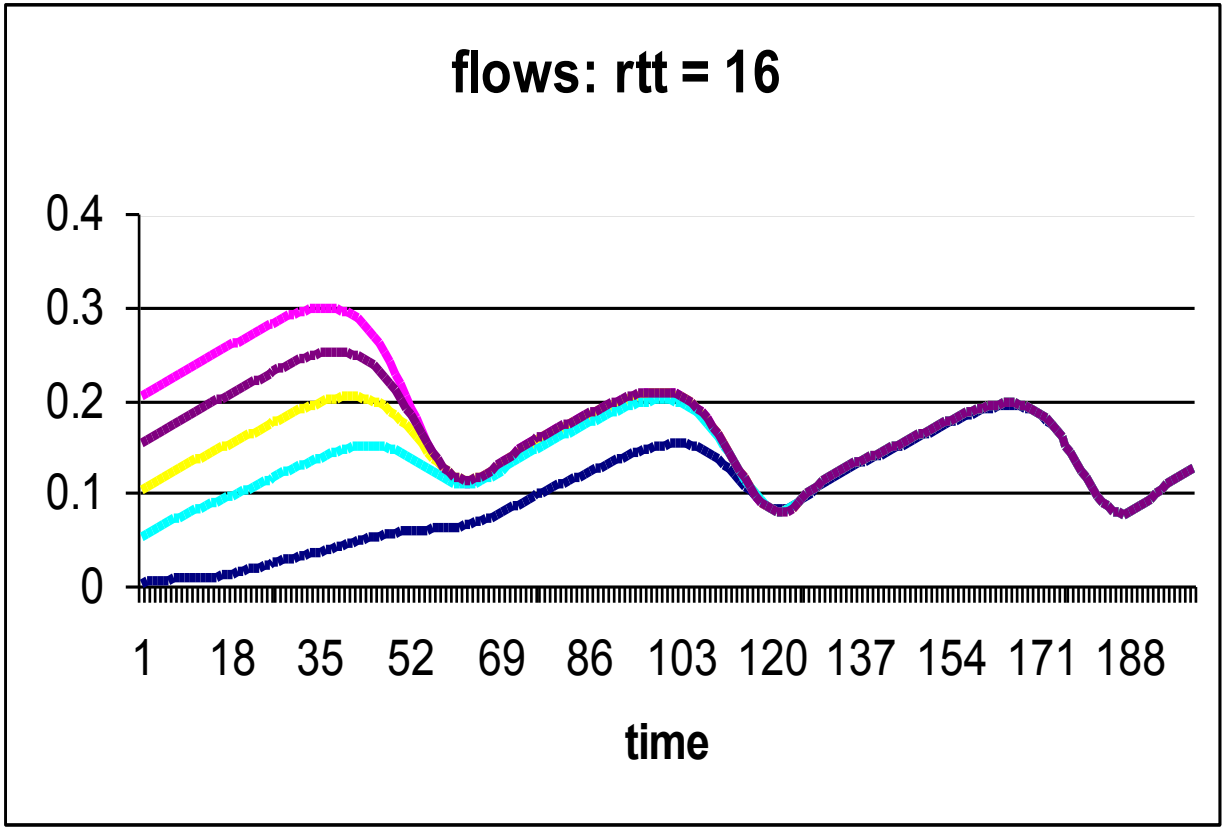
Threshold = 5


flows: rtt = 16
0.25
0.2
0.15
0.1
0.05
0
1
19
37
55
73
91
109
127
145
163
181
199
time

Threshold = 10



Threshold = 20



Delay stability

Johari and Tan (1999), Massoulié (2000), Vinnicombe (2000):
Equilibrium is locally 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

Consequences?

Delay stability condition: $a_r(cwnd_r)^n < \frac{\pi}{2\beta}$

- $n = -1$ (Jacobson's TCP)

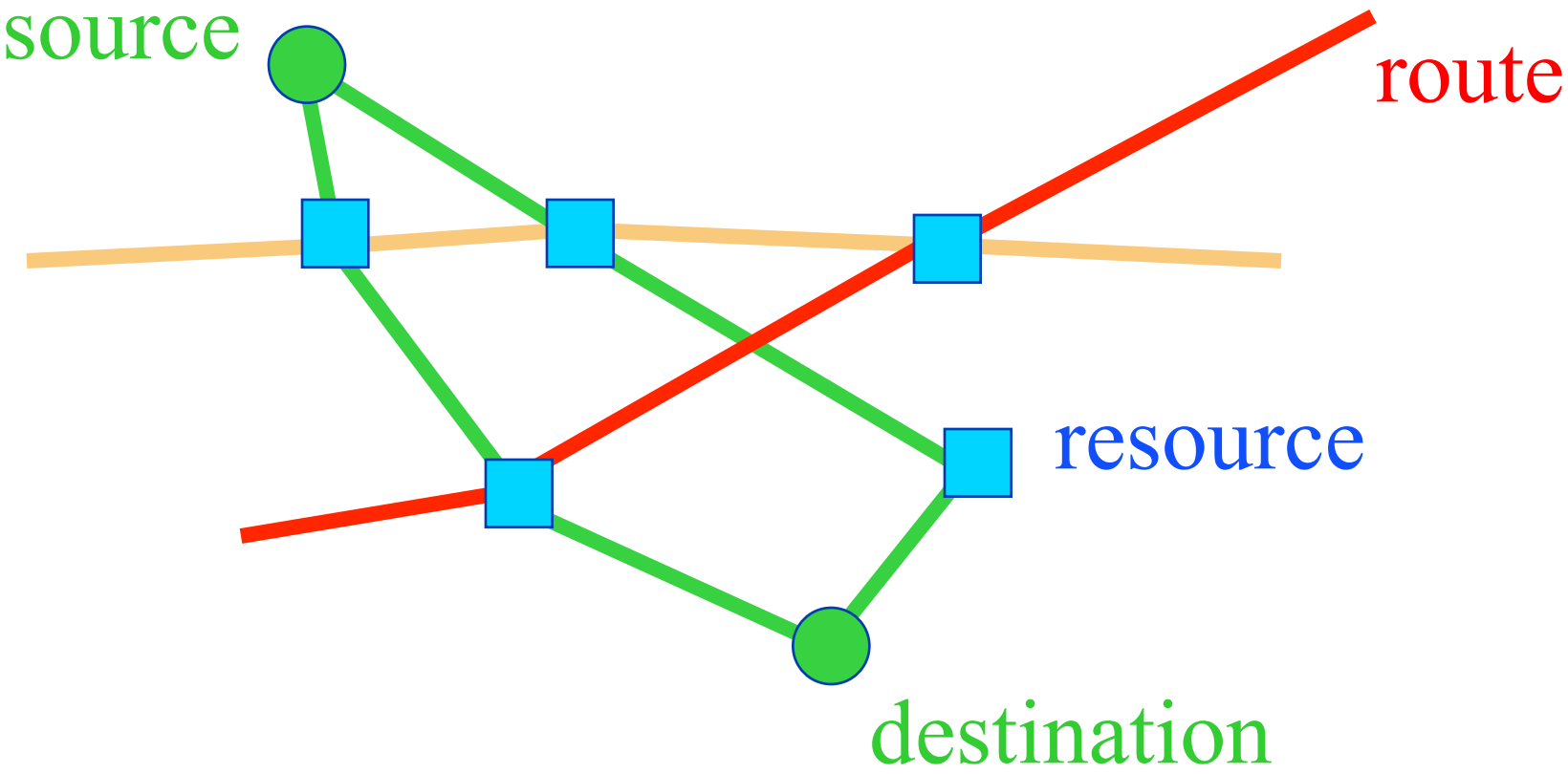
instability if congestion windows are too *small*,
sluggishness if congestion windows are too *big*

- $n = 0$ (Scalable TCP, Glenn Vinnicombe, Tom Kelly)

condition *independent* of size of congestion window,
and choice of b_r can remove round-trip time bias

Routing?

Can we extend algorithms to allow dynamic routing? Danger: route flap.



Notation

J - set of resources

R - set of routes

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

S - set of source-destination pairs

$r \in s$ - route r serves s-d pair s

$$y_s(t) = \sum_{r \in s} x_r(t - T_r)$$

- rate of returning acknowledgements for s-d pair s at time t

Combined rate control and routing algorithm

On route r

- $x_r(t)$ increased by a / T_r on positive acknowledgement
- $x_r(t)$ decreased by $b_r y_{s(r)}(t) / T_r$ for each congestion indication

$s = \{r\}$ corresponds to scalable TCP

Delay stability

Kelly and Voice, 2005
Han, Shakkottai, Hollot, Srikant and Towsley, 2006

Equilibrium is locally stable if there exists a global constant β such that

$$x p_j'(x) < \beta \; p_j(x), \qquad a\,(1+\beta) < \frac{\pi}{2}$$

condition on sensitivity for each resource j

condition on aggressiveness of sources

Delay stability

Kelly and Voice, 2005
Han, Shakkottai, Hollot, Srikant and Towsley, 2006

Equilibrium is locally stable if there exists a global constant β such that

impact of routing

$$x p_j'(x) < \beta \; p_j(x), \qquad a\,(1+\beta) < \frac{\pi}{2}$$

condition on sensitivity for each resource j

condition on aggressiveness of sources

Conclusion?

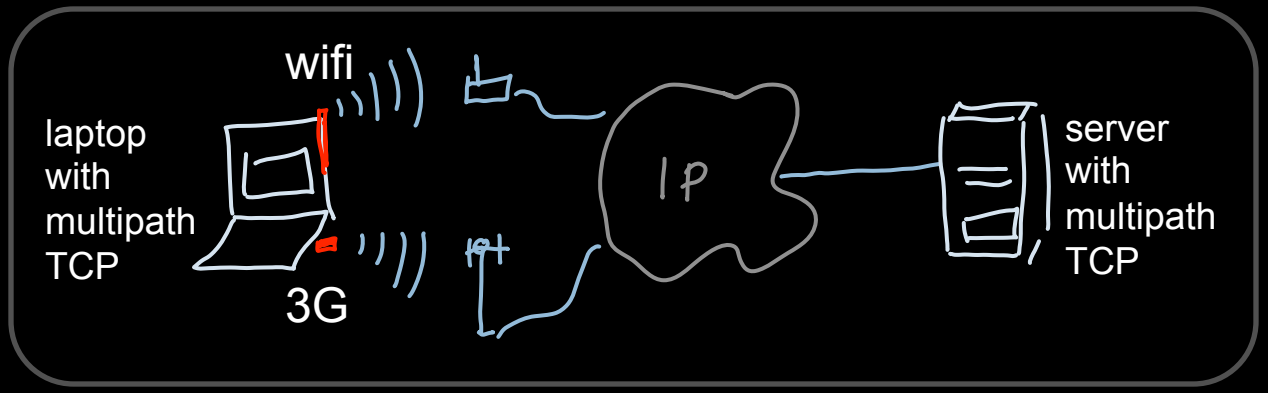
- It may be possible to achieve stable, scalable load sharing across paths, based on end-to end measurements, on the same time-scale as rate control
- For dynamic routing, the key constraint on the responsiveness of each route is the round-trip time of that route, information which is naturally available at sources

Multipath TCP

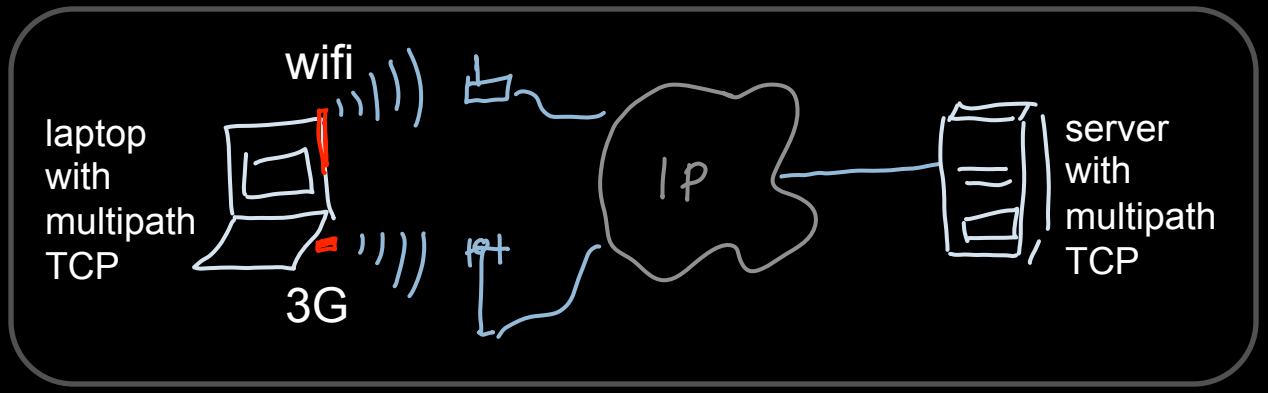
Mark Handley
Costin Raiciu
Damon Wischik
UCL

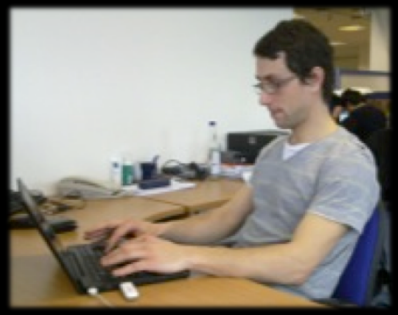
- from http://www.cs.ucl.ac.uk/staff/D.Wischik/

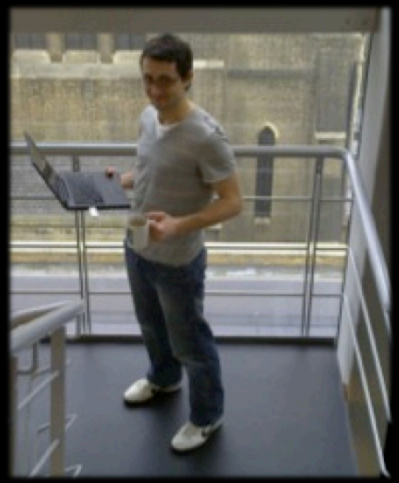
We have a working implementation of multipath transport



We have a working implementation of multipath transport

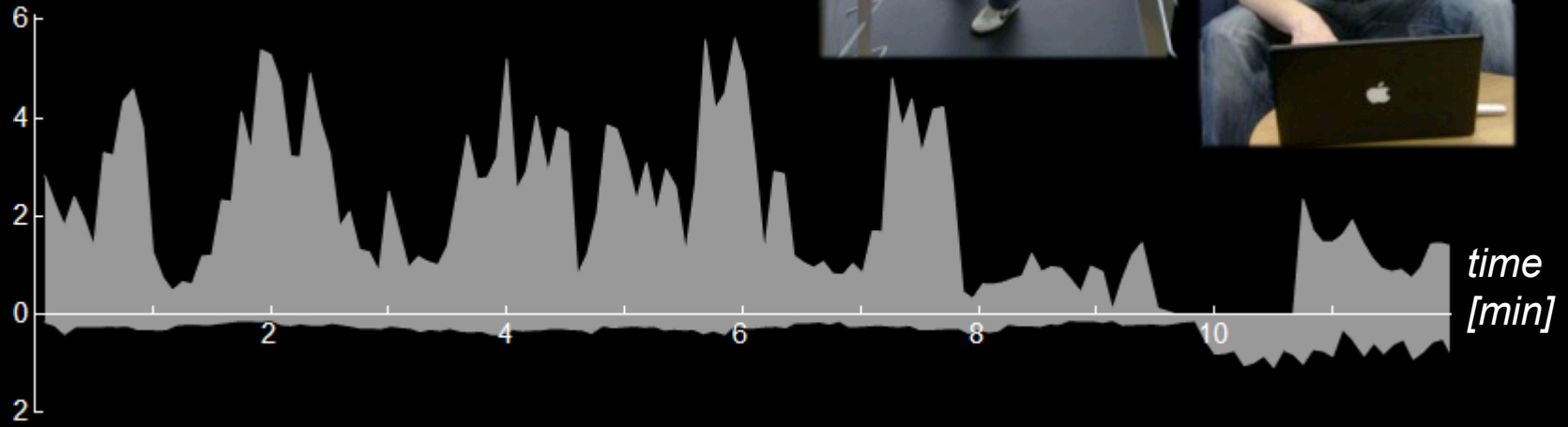






wifi through-put [Mb/s]

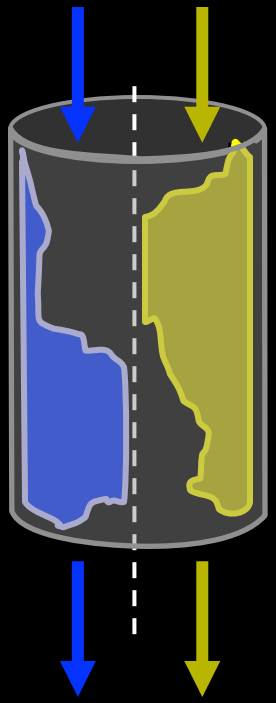
3G through-put [Mb/s]



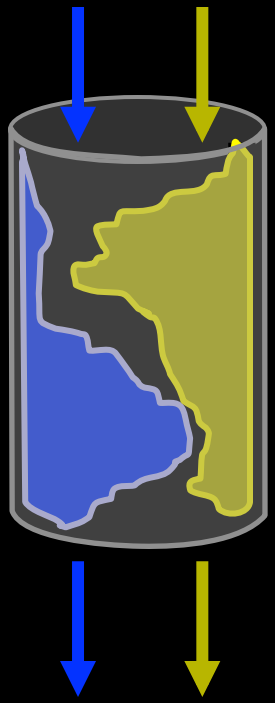
This user clearly benefits from multipath. But is it safe for the network and other users? Or does it cause instability, route flap, unfairness, disaster?

I. Resource pooling as a design principle

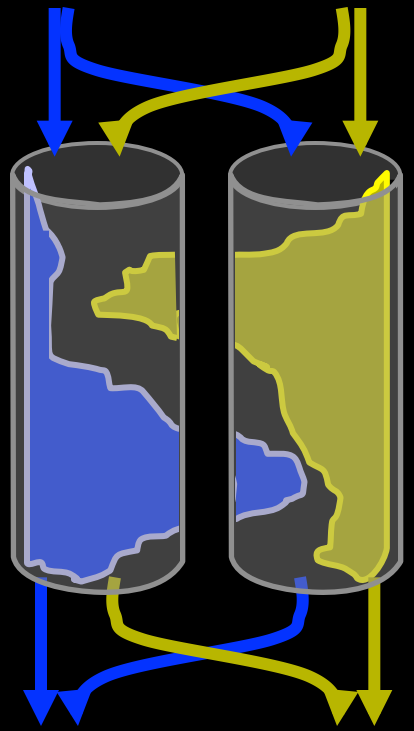
Resource pooling means "making a collection of resources behave like a single pooled resource". It has been a design goal of the Internet from the beginning.



A single link, split into two circuits

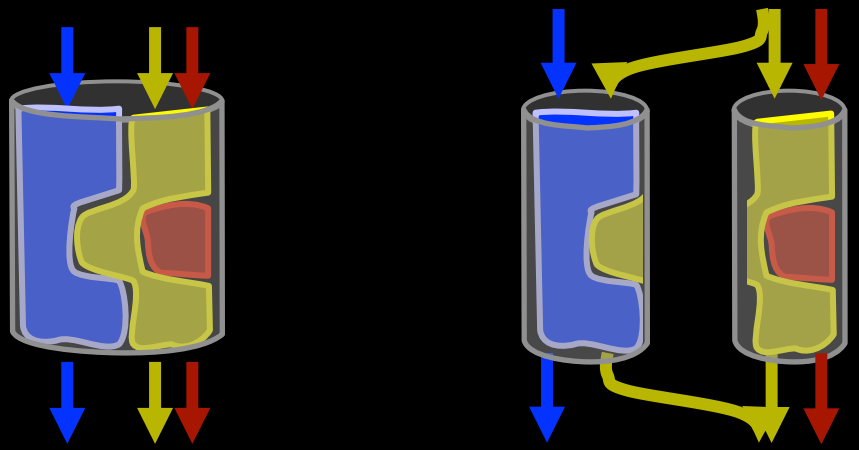


Packet switching "pools" the two circuits

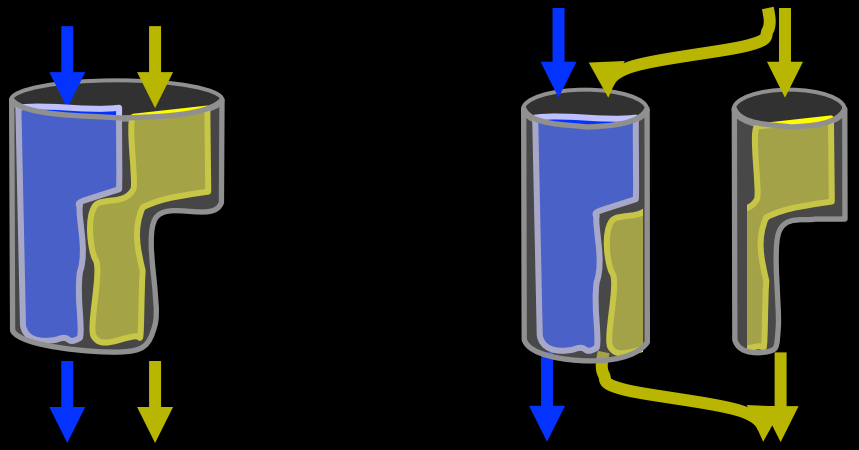


Multipath "pools" the two links

Resource pooling means the network is better able to accommodate a surge in traffic

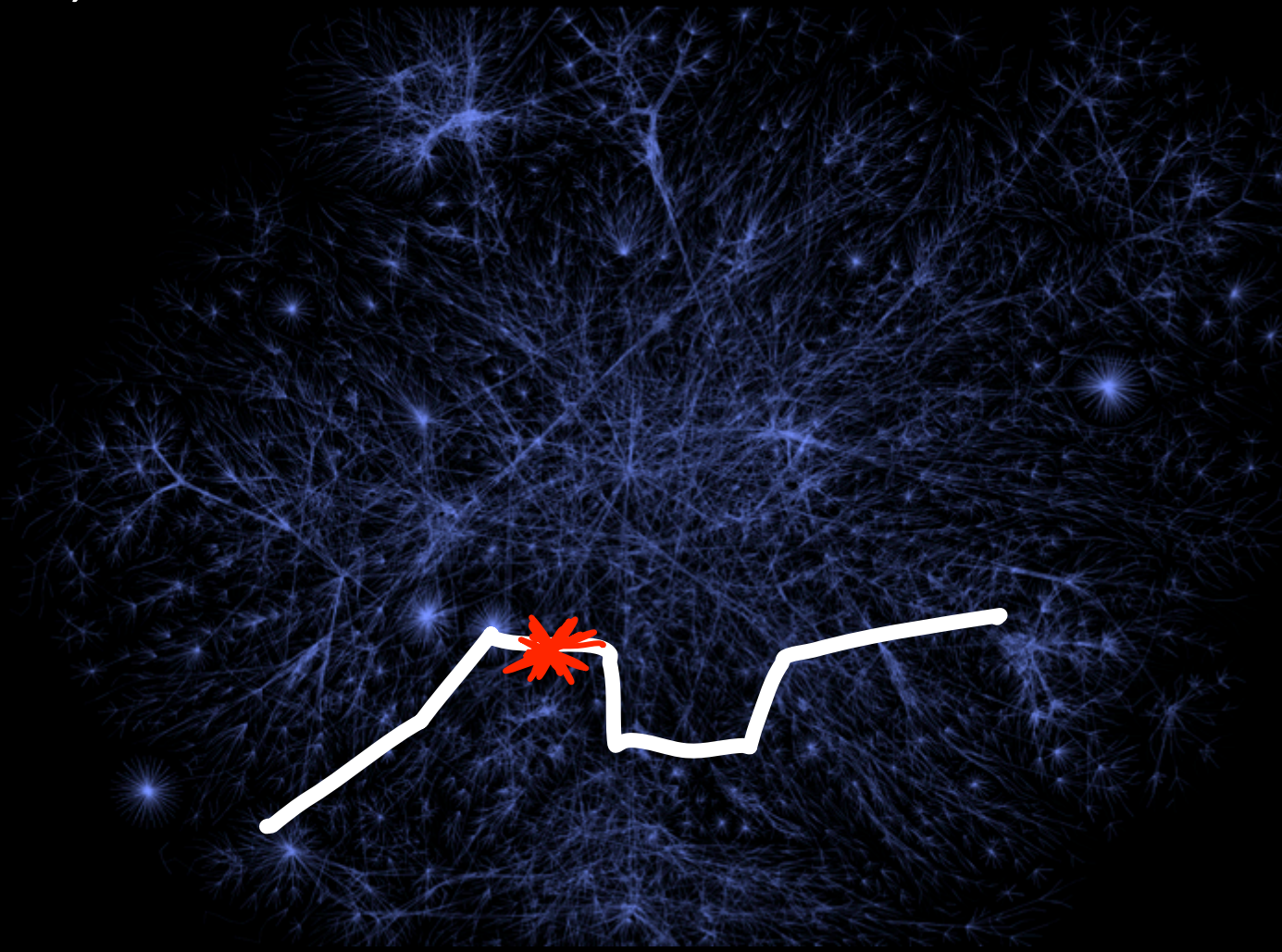


or a loss of capacity

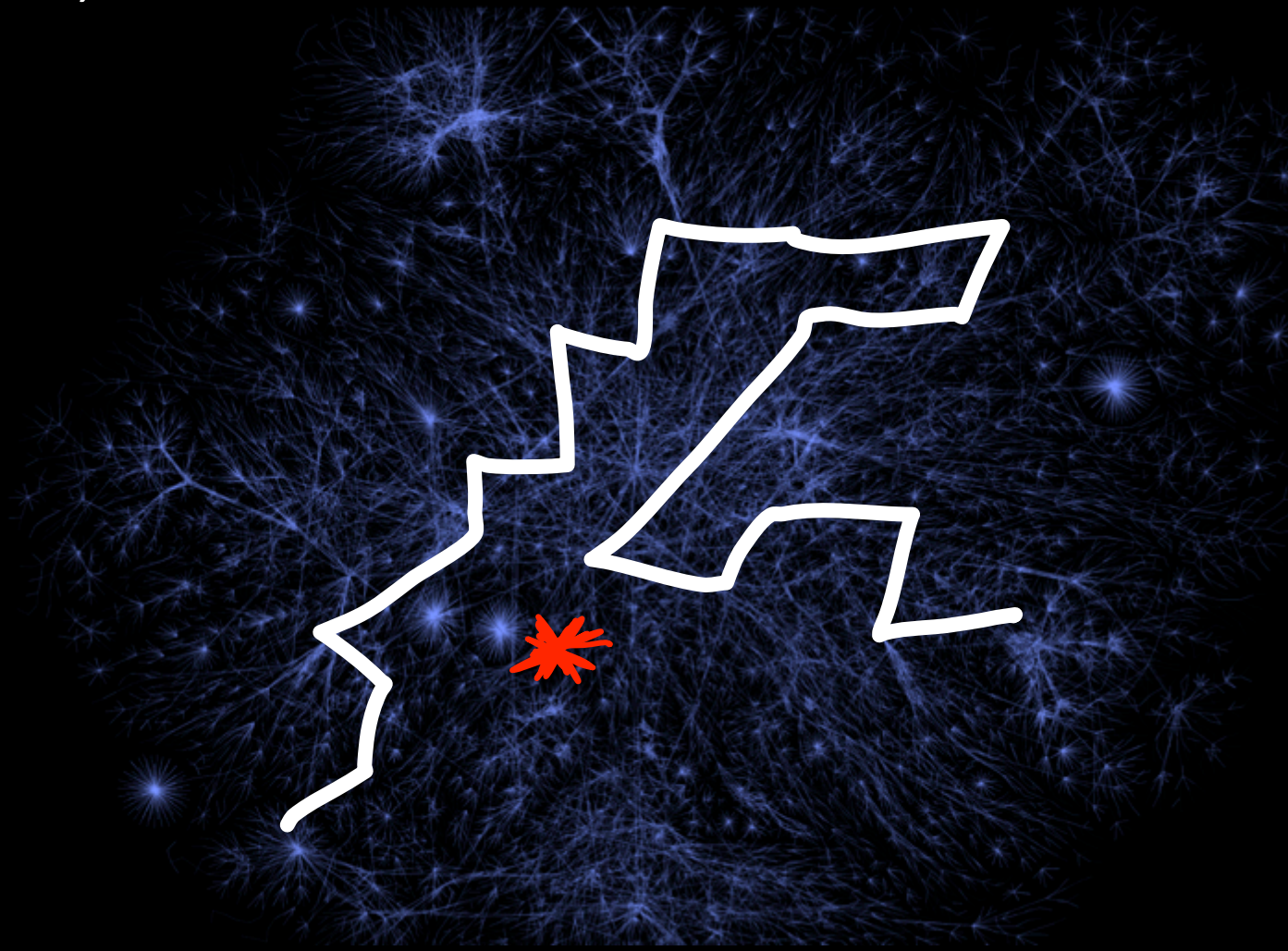


by shifting traffic and thereby “diffusing” congestion across the network.

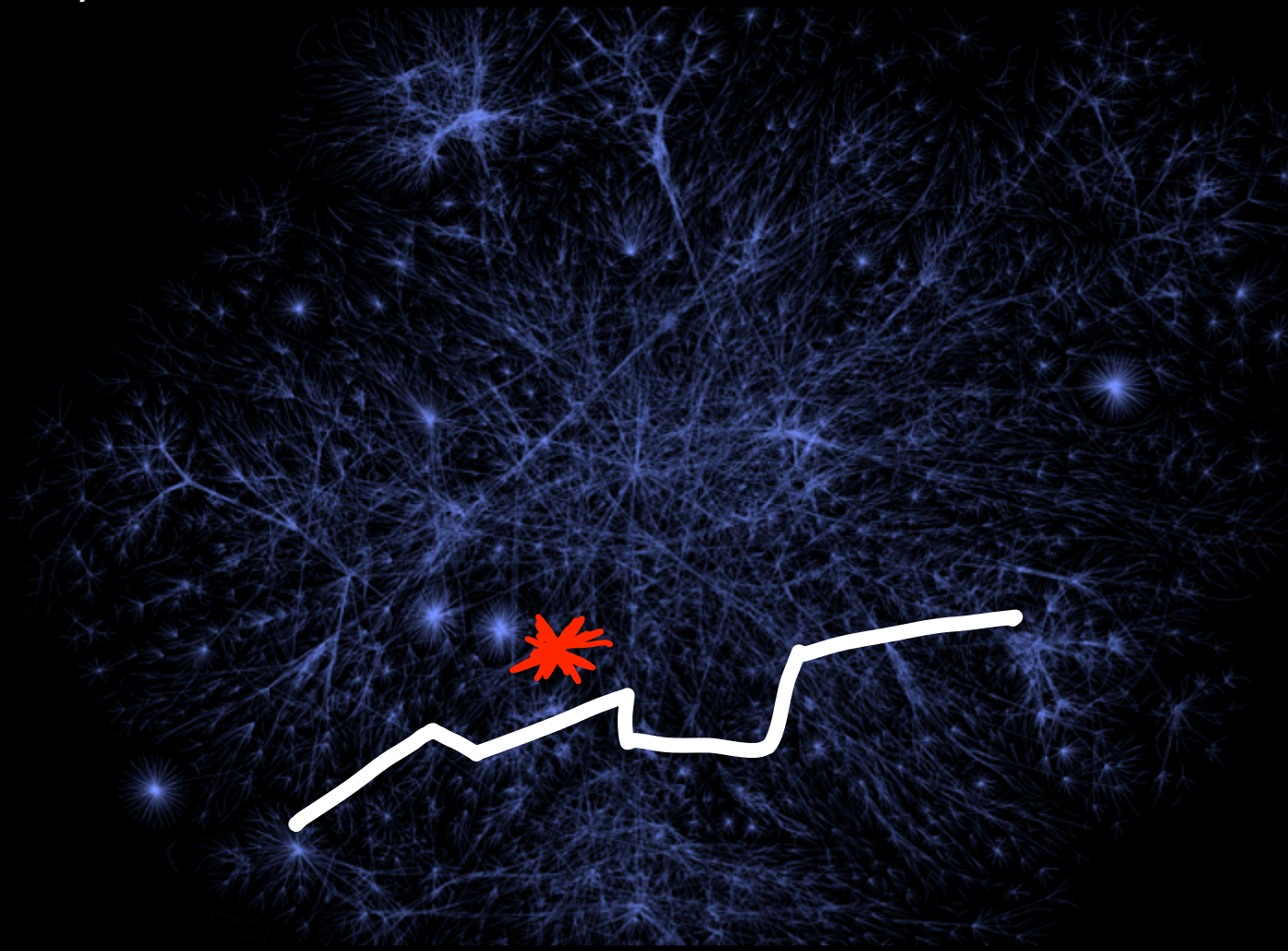
The Internet already has resource pooling, in the form of multi-homing, BGP, etc.



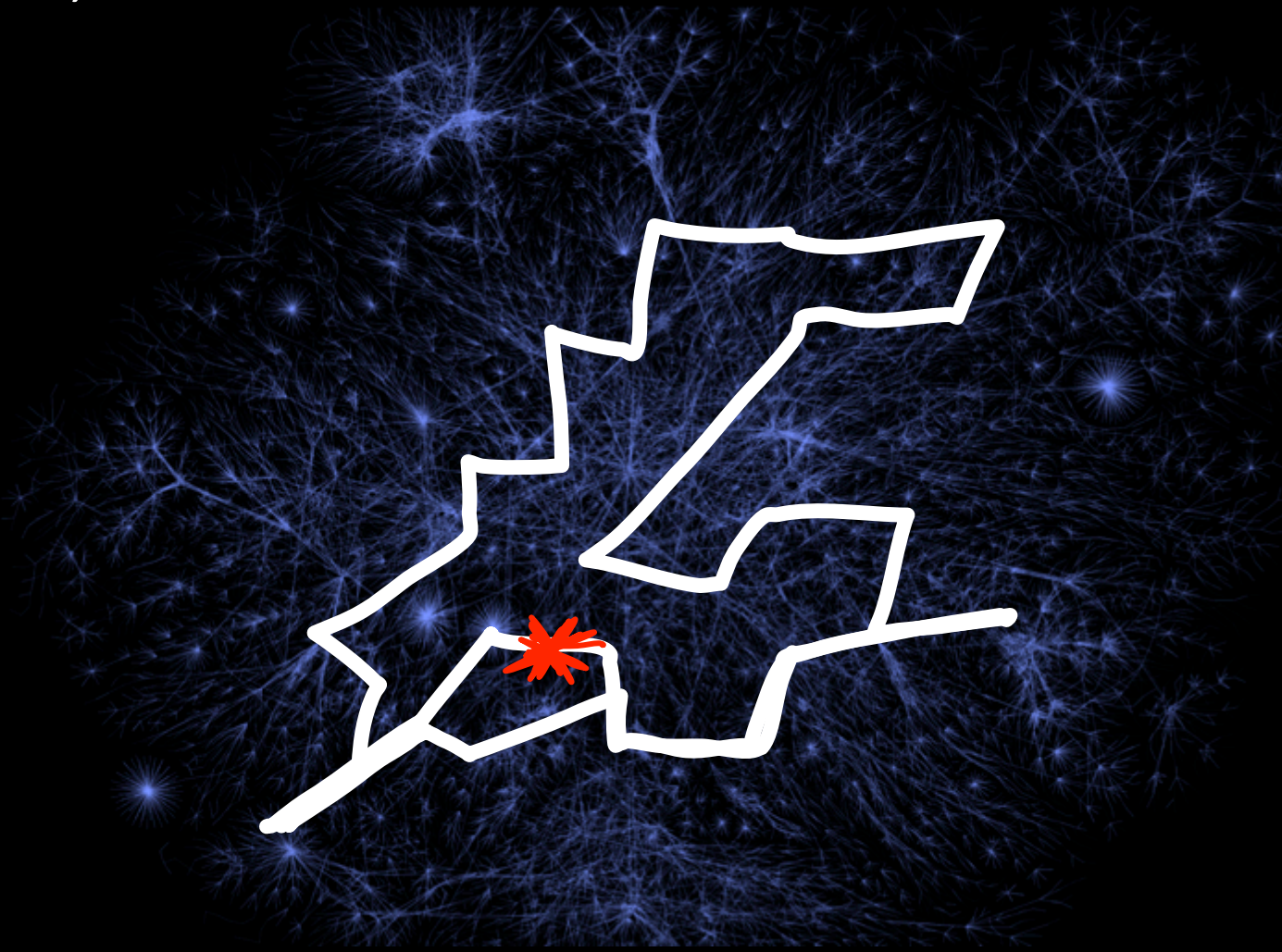
The Internet already has resource pooling, in the form of multi-homing, BGP, etc.



The Internet already has resource pooling, in the form of multi-homing, BGP, etc.



The Internet already has resource pooling, in the form of multi-homing, BGP, etc.



We think resource pooling should be achieved by end-system multipath. This would harness the rapid responsiveness of end systems.

There is a large body of work on fluid models of congestion control:

- write down a network utility maximization problem,
- write down a system of differential equations,
- show that the (unique) fixed point solves the utility maximization,
- and interpret it as a discrete congestion control algorithm.

Multipath congestion control theory has been developed by Kelly and Voice (2005), and by Han, Shakkottai, Hollot, Srikant, Towsley (2006).

e.g. $$\frac{d}{dt} x_r(t) = \kappa_r \left(x_r(t) - x_r(t)\, p_r(t)\, y(t) \right)$$

where $x_r(t)$ = sending rate on path r at time t

$y(t)$ = total rate over all flows for this user

$p_r(t)$ = packet drop probability on path r

Interpretation

- Increase x_r by a constant, every time you get an acknowledgement on path r
- Decrease x_r by an amount proportional to $y_{s(r)}$ if you detect a drop on path r

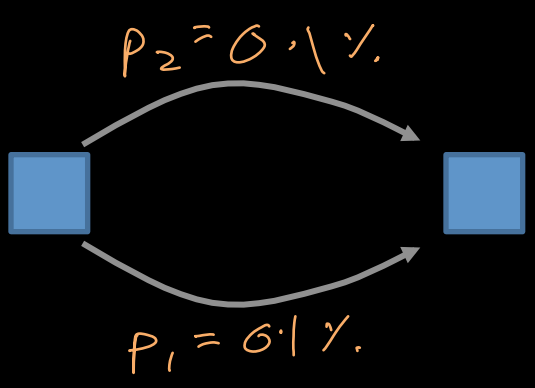
How we expect the fluid model to behave:

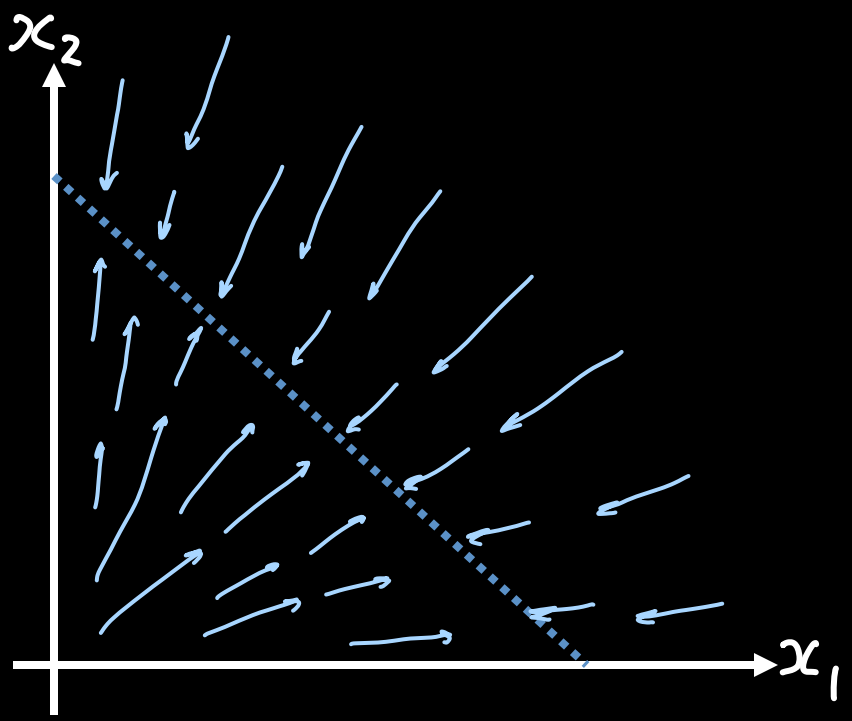
$$\frac{d}{dt} x_r(t) = \kappa_r \left(x_r(t) - x_r(t)\, p_r(t)\, y(t) \right)$$

where $x_r(t)$ = sending rate on path r at time t

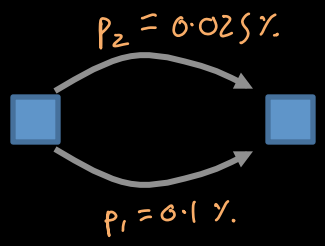
$y(t)$ = total rate over all flows for this user

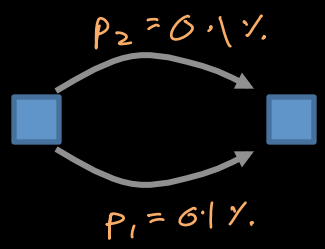
$p_r(t)$ = packet drop probability on path r

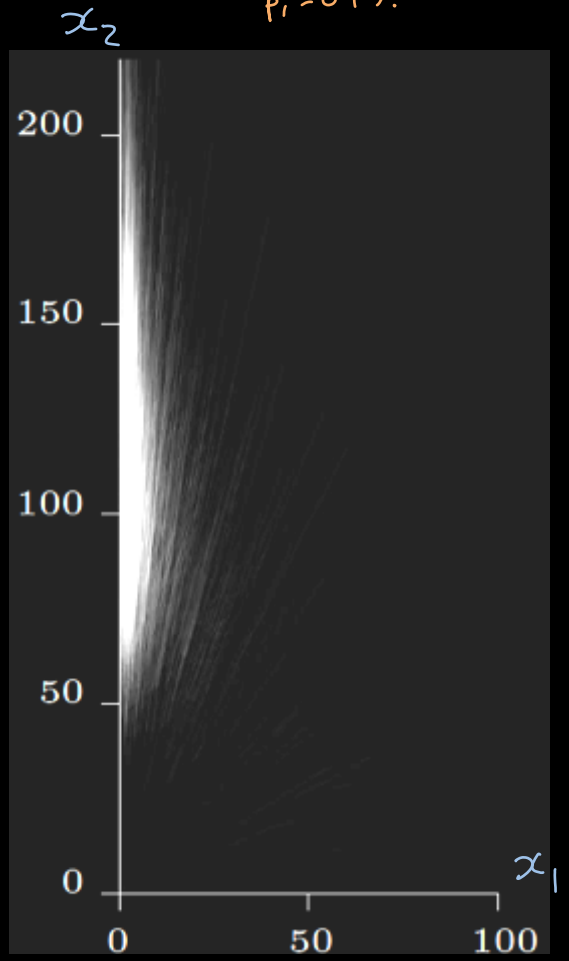


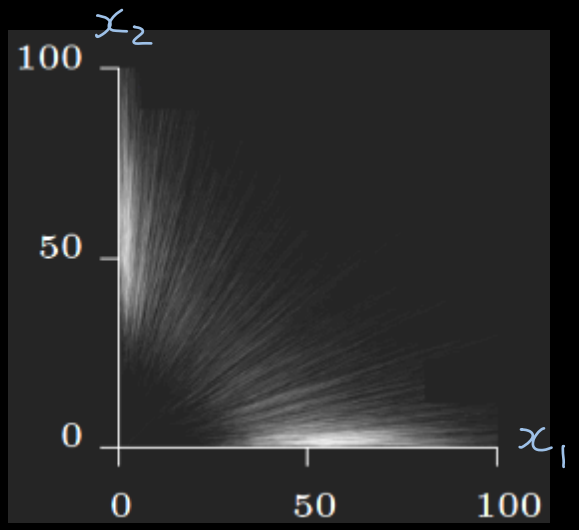


How they behave in simulation:









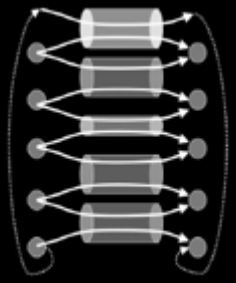
When there are many flows, then each flow will flip independently, and the aggregate will behave how the fluid models predict.

the naïve coupled congestion controller, inspired by Kelly+Voice

run independent TCP control on each path

$\phi=0$ $\phi=2$

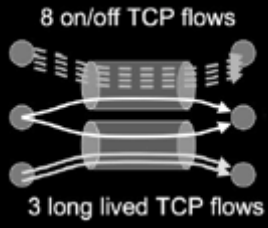
static network



good at resource pooling:
even though the links have unequal capacities, congestion is balanced perfectly

bad at resource pooling:
the low-capacity link is highly congested

dynamic network



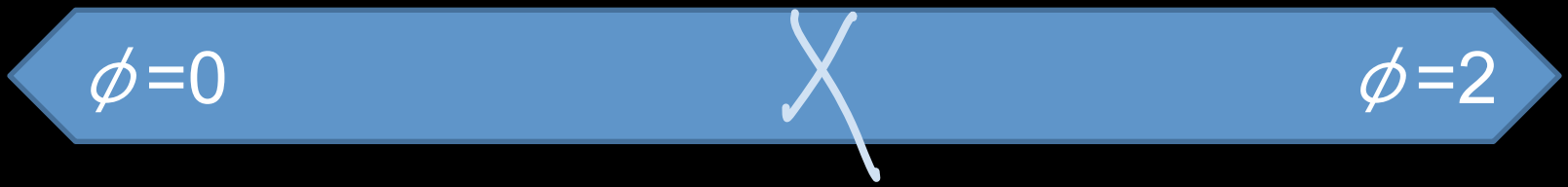
bad at resource pooling:
shifts too enthusiastically to the less loaded link, and is slow to learn when the other link improves

good at resource pooling:
constantly probes both links, so learns quickly when congestion levels change

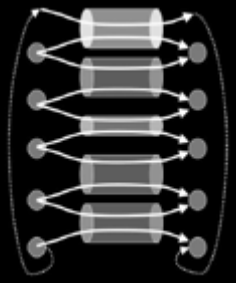
our choice

the naïve coupled congestion controller, inspired by Kelly+Voice

run independent TCP control on each path



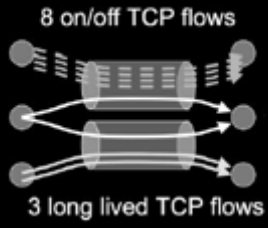
static network



good at resource pooling:
even though the links have unequal capacities, congestion is balanced perfectly

bad at resource pooling:
the low-capacity link is highly congested

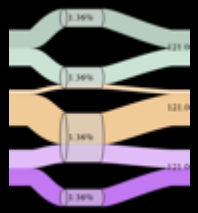
dynamic network



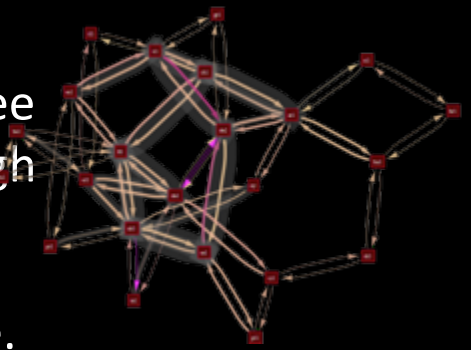
bad at resource pooling:
shifts too enthusiastically to the less loaded link, and is slow to learn when the other link improves

good at resource pooling:
constantly probes both links, so learns quickly when congestion levels change

SUMMARY. We have a working implementation of multipath transport.

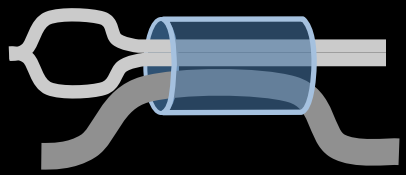


It achieves a reasonable degree of load balancing.



This means that the network achieves some degree of resource pooling (subject to having good enough routes).

It maintains a reasonable degree of equipoise. This means it adapts sensibly to fluctuating congestion.



It is guaranteed to be fair compared to TCP.

I E T F

The algorithm is ready for deployment. It is an experimental RFC in the mptcp working group at the IETF.

Multipath TCP

Mark Handley
Costin Raiciu
Damon Wischik
UCL

- from http://www.cs.ucl.ac.uk/staff/D.Wischik/