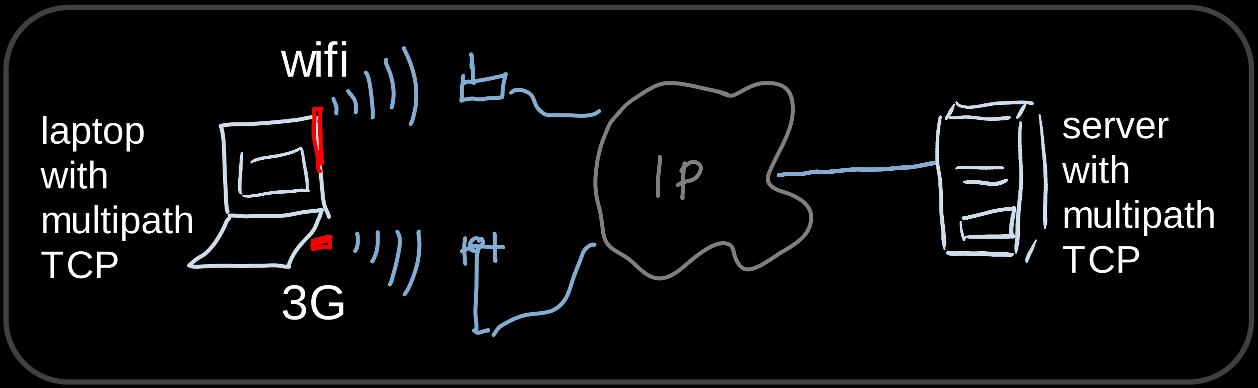
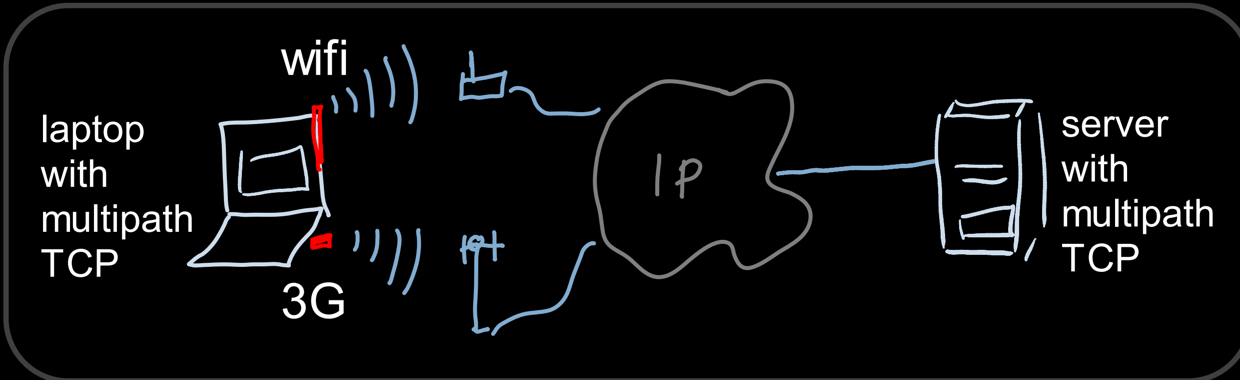
Multipath TCP

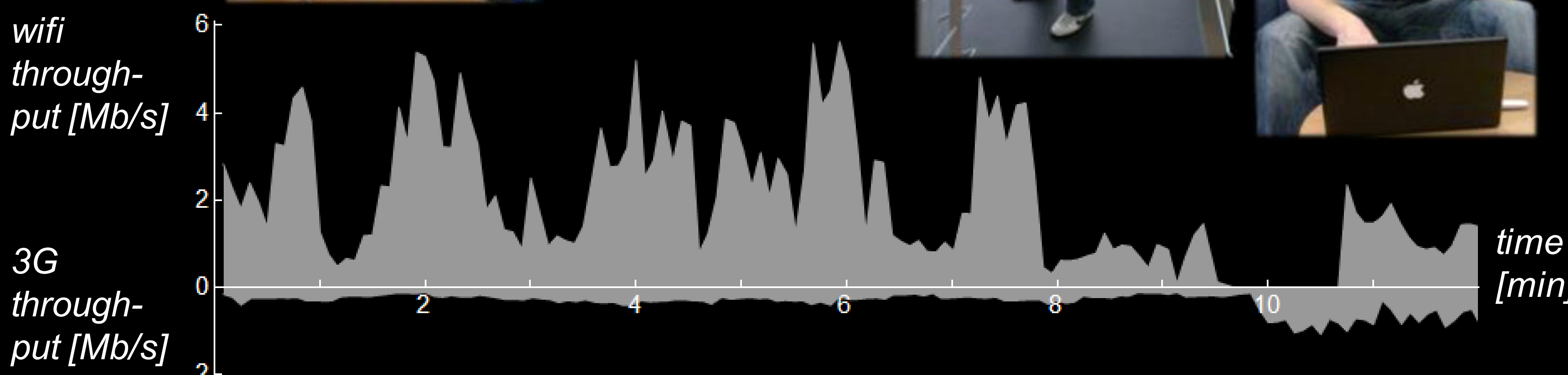
Mark Handley
Costin Raiciu
Damon Wischik
UCL

We have a working implementation of multipath transport

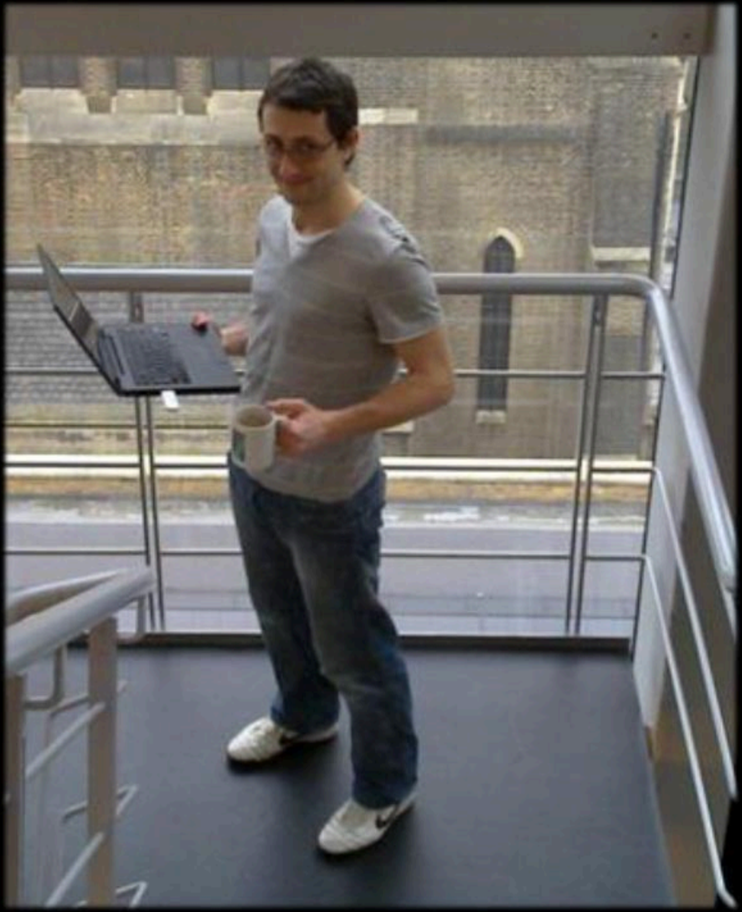


We have a working implementation of multipath transport





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?



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?

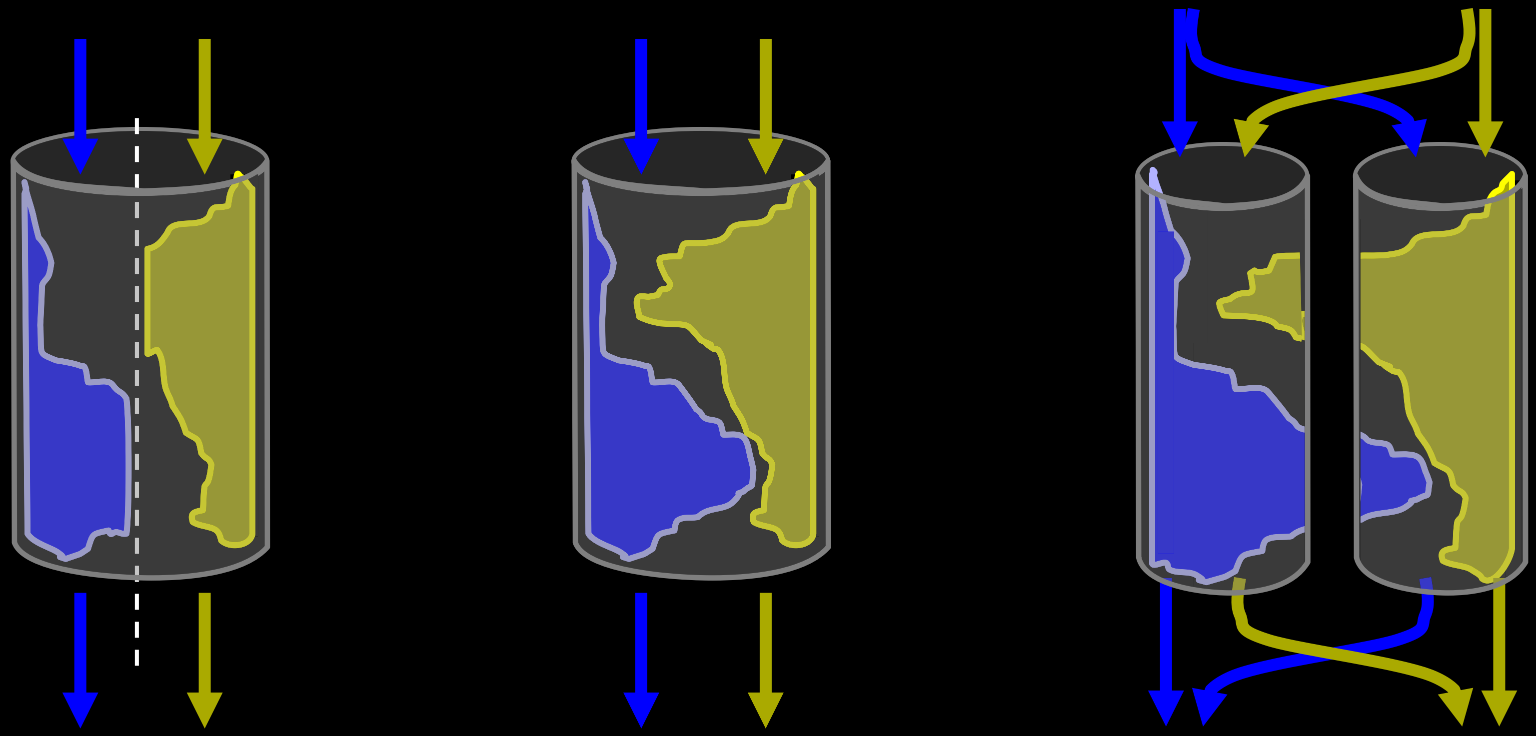
I. **Resource pooling as a design principle**
The earliest design goal of the Internet aimed to achieve “resource pooling”, and multipath transport is a natural extension.

II. **How to measure the pooling potential of a multipath topology**
We have a metric for measuring how much resource pooling there can be, given the topology and traffic matrix. This will be useful for designing multipath routing algorithms.

III. **A coupled congestion control algorithm**
We have designed and implemented a multipath congestion control algorithm that balances load, and we can guarantee it’s safe to deploy (but it’s harder than you’d think to do it right)

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.

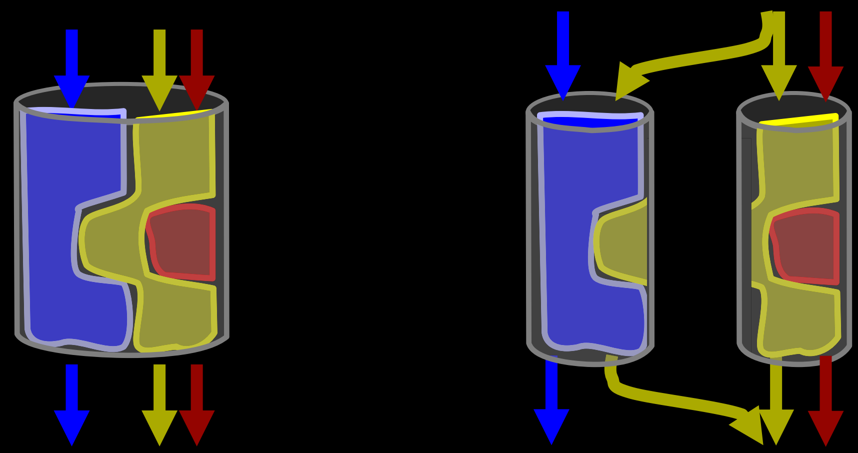


A single link, split into two circuits

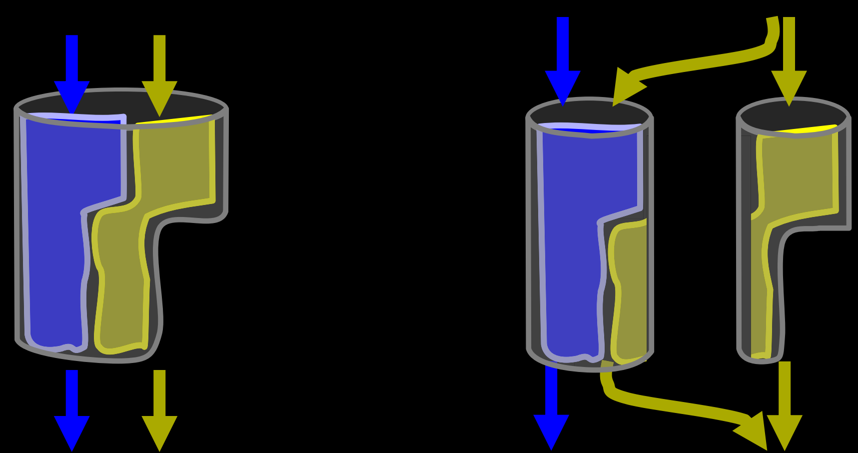
Packet switching "pools" the two circuits

Multipath "pools" the two links

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

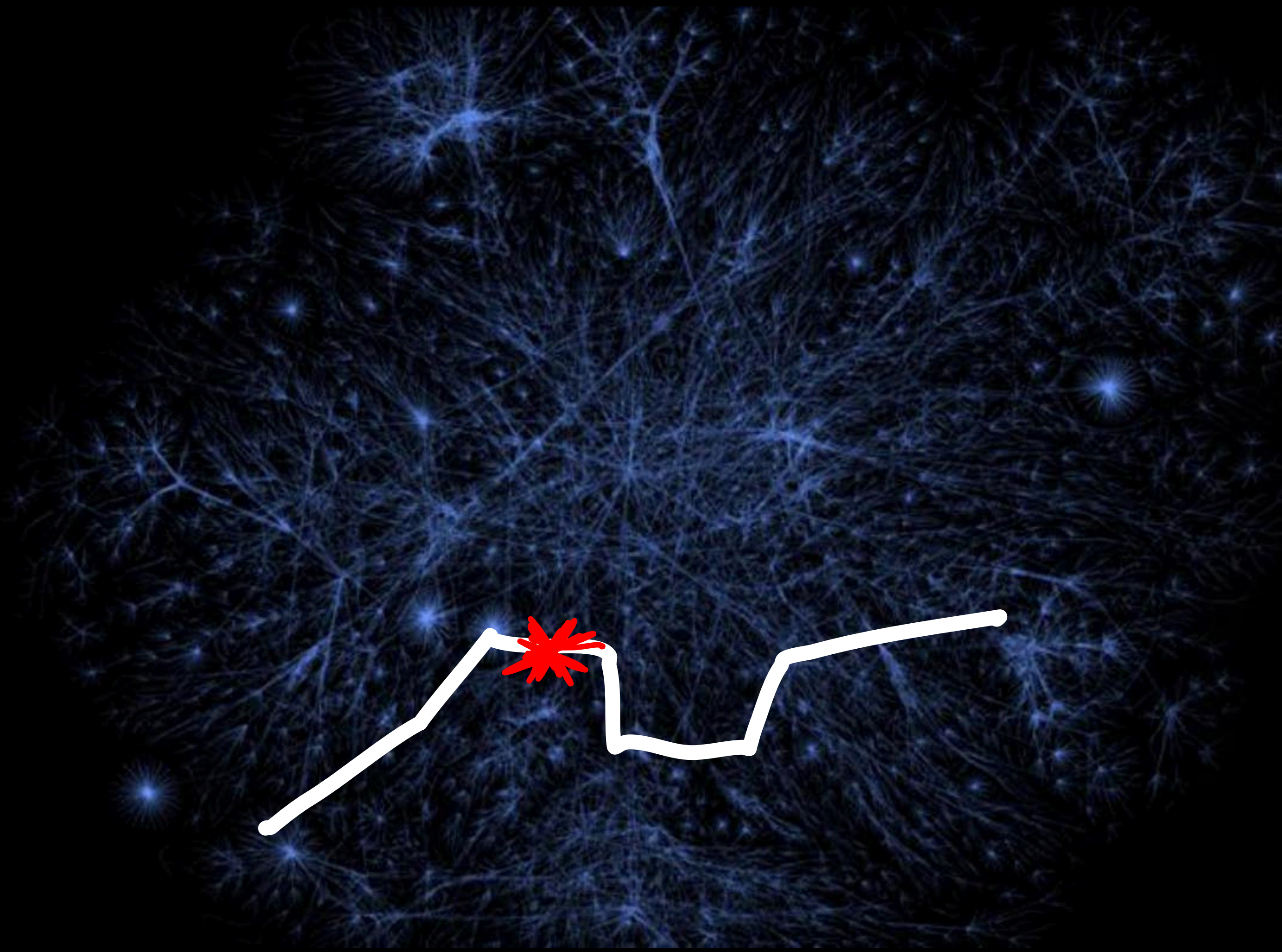


or a loss of capacity

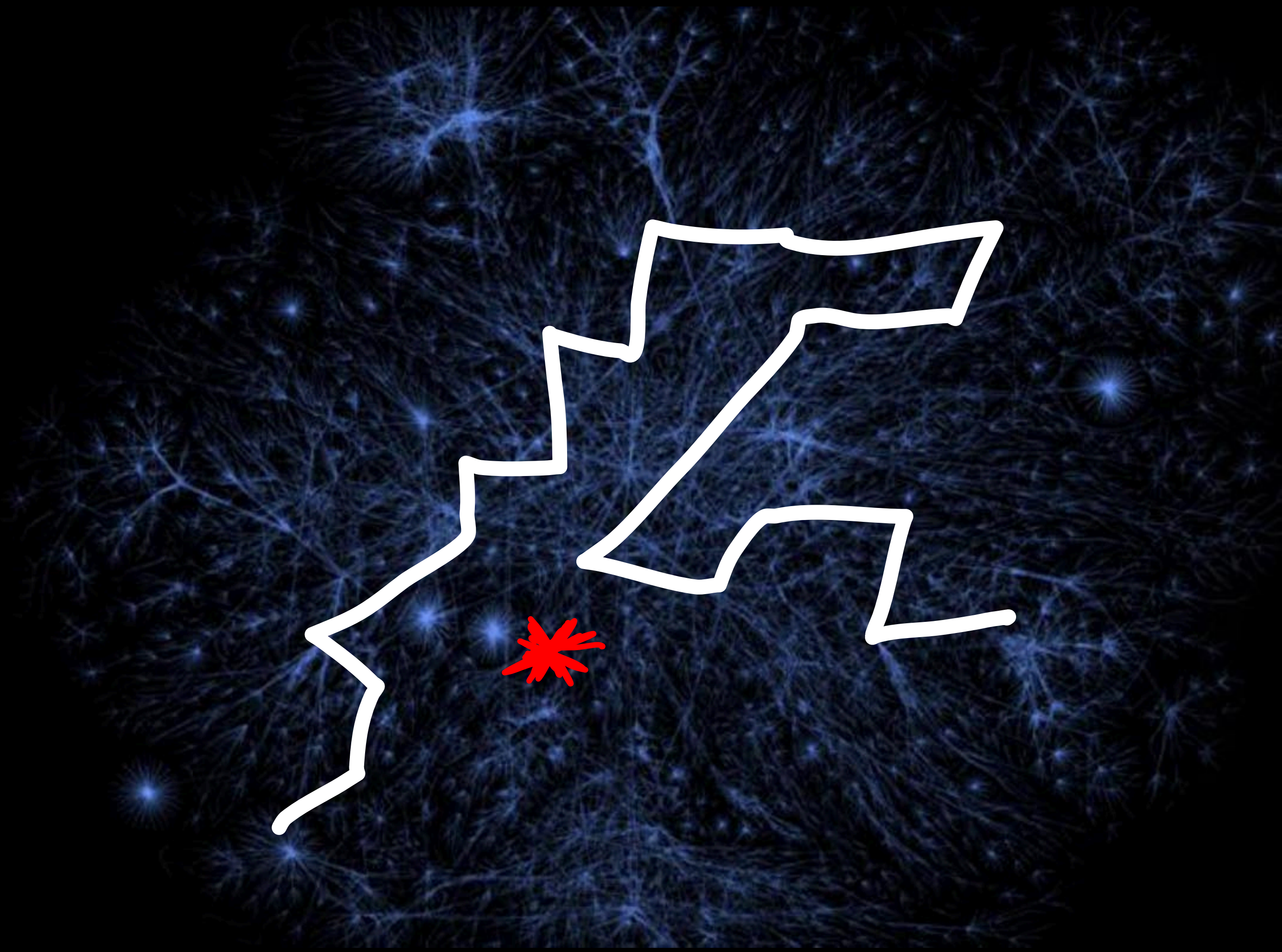


by shifting traffic and thereby “diffusing” congestion across the network.

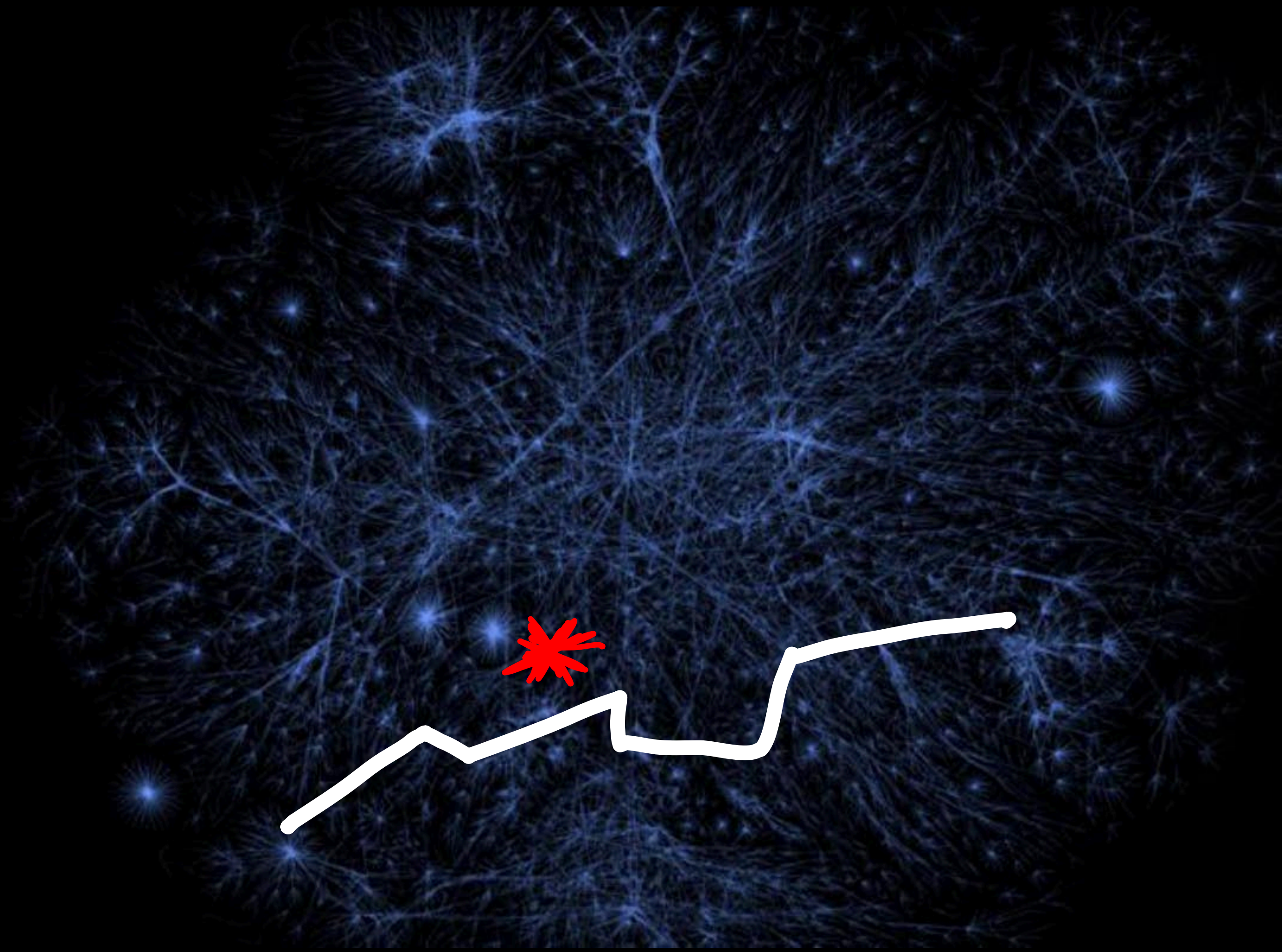
The Internet already has resource pooling, in the form of multi-homing, BGP, etc.



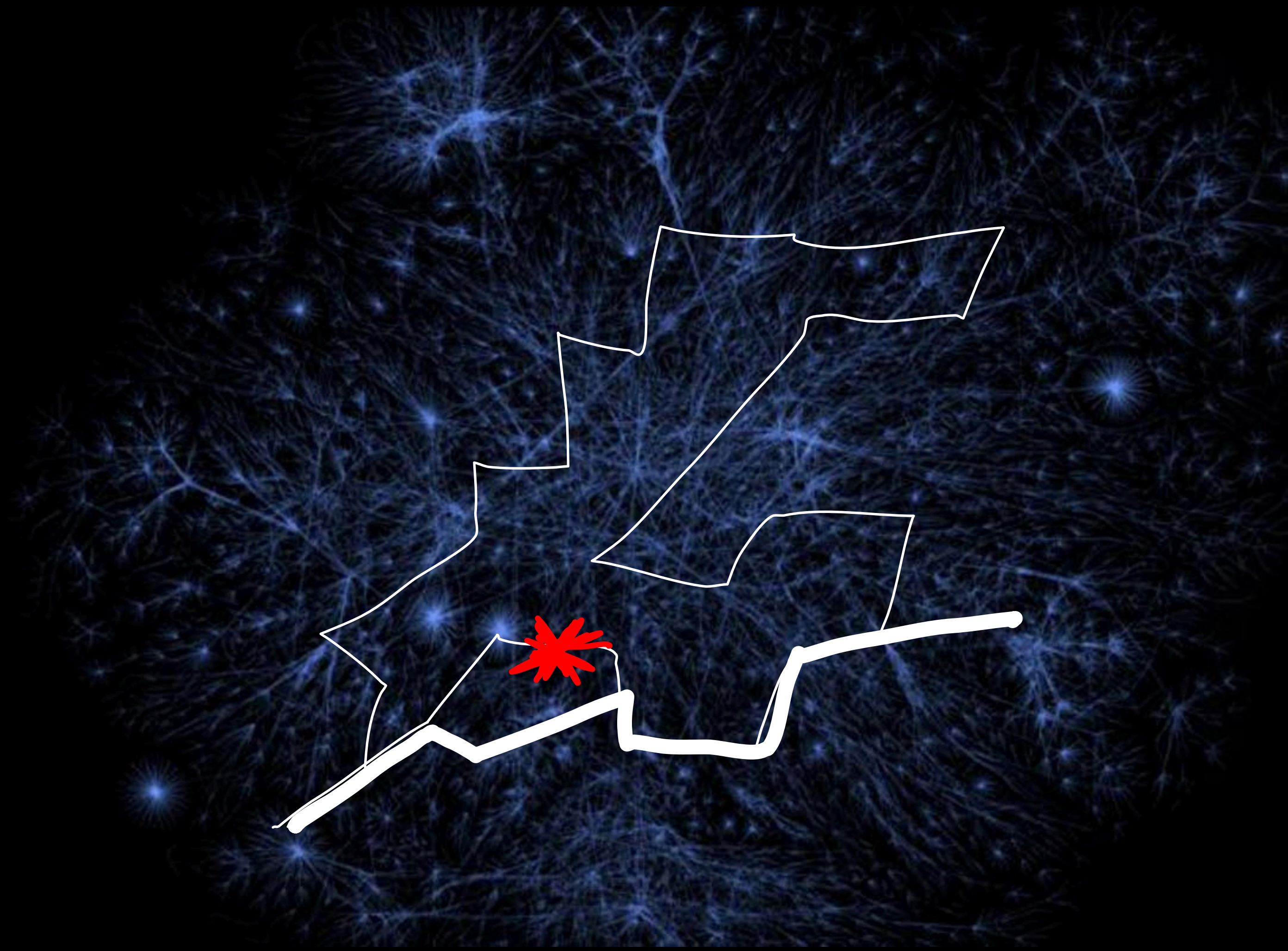
The Internet already has resource pooling, in the form of multi-homing, BGP, etc.



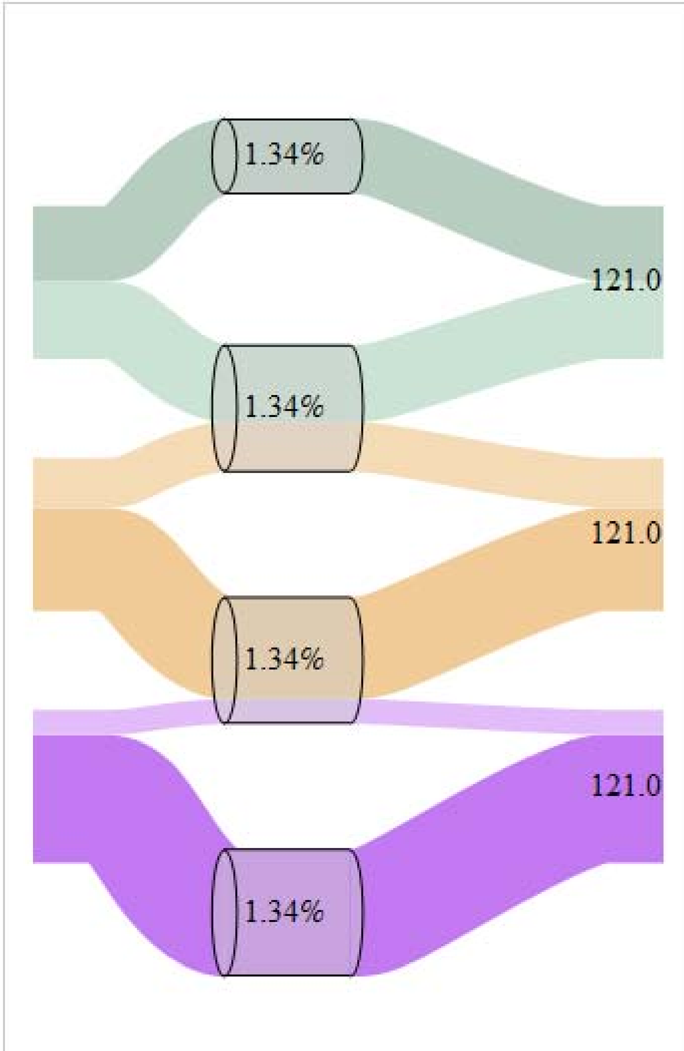
The Internet already has resource pooling, in the form of multi-homing, BGP, etc.

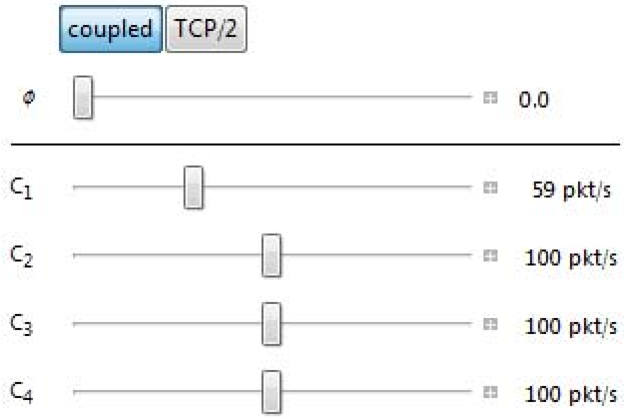


The Internet already has resource pooling, in the form of multi-homing, BGP, etc.



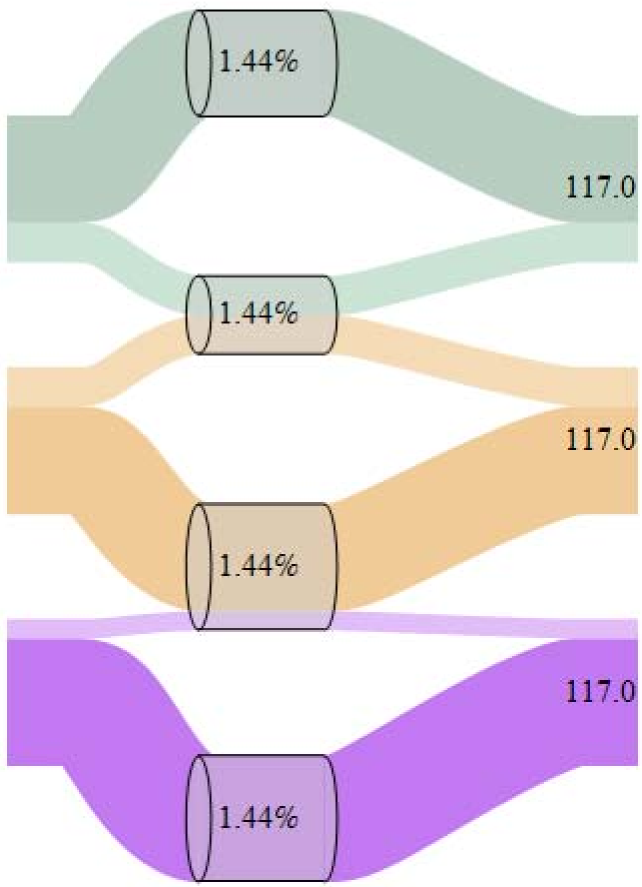
We think resource pooling should be achieved by end-system multipath. This would harness the rapid responsiveness of end systems.

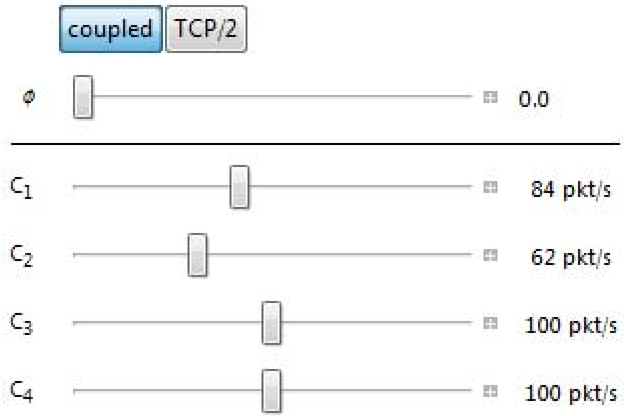




Resource pooling means:

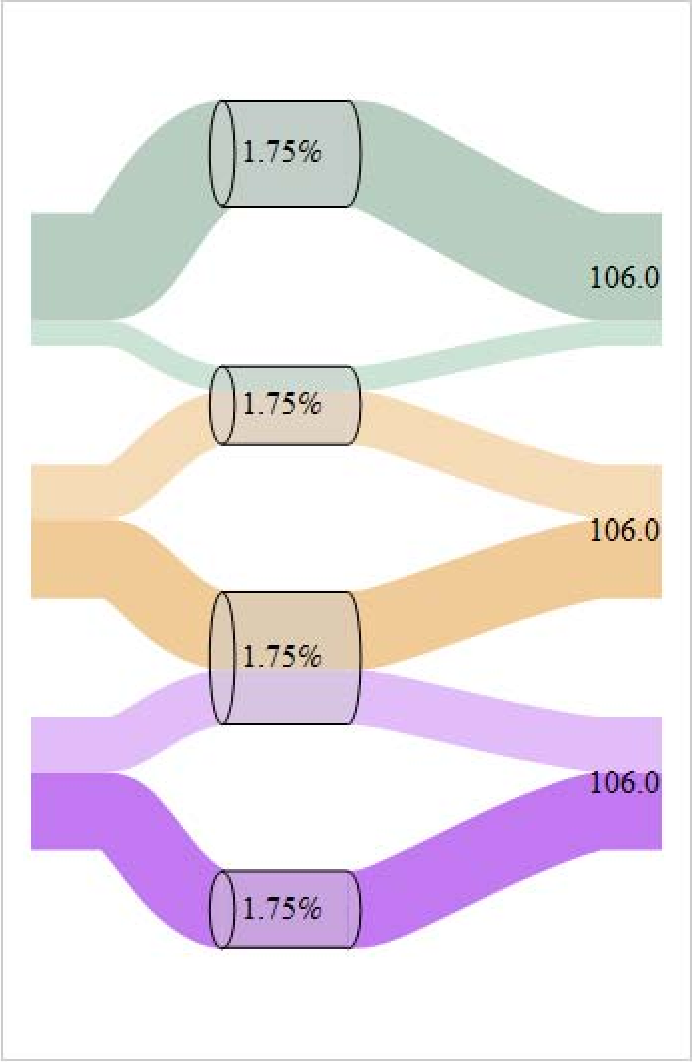
Congestion due to reduced capacity etc. can be shared across the entire network





Resource pooling means:

Congestion due to reduced capacity etc. can be shared across the entire network



coupled TCP/2

ϕ 0.0

C_1 84 pkt/s

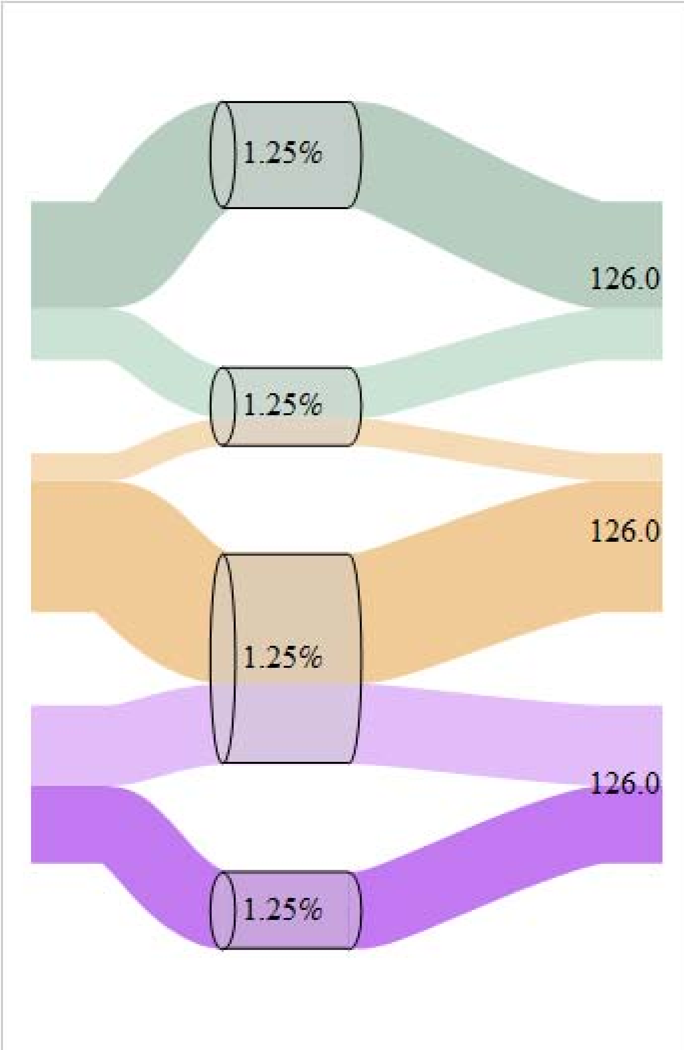
C_2 62 pkt/s

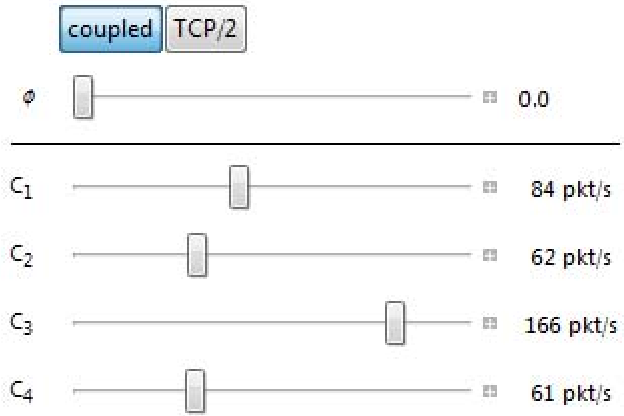
C_3 105 pkt/s

C_4 61 pkt/s

Resource pooling means:

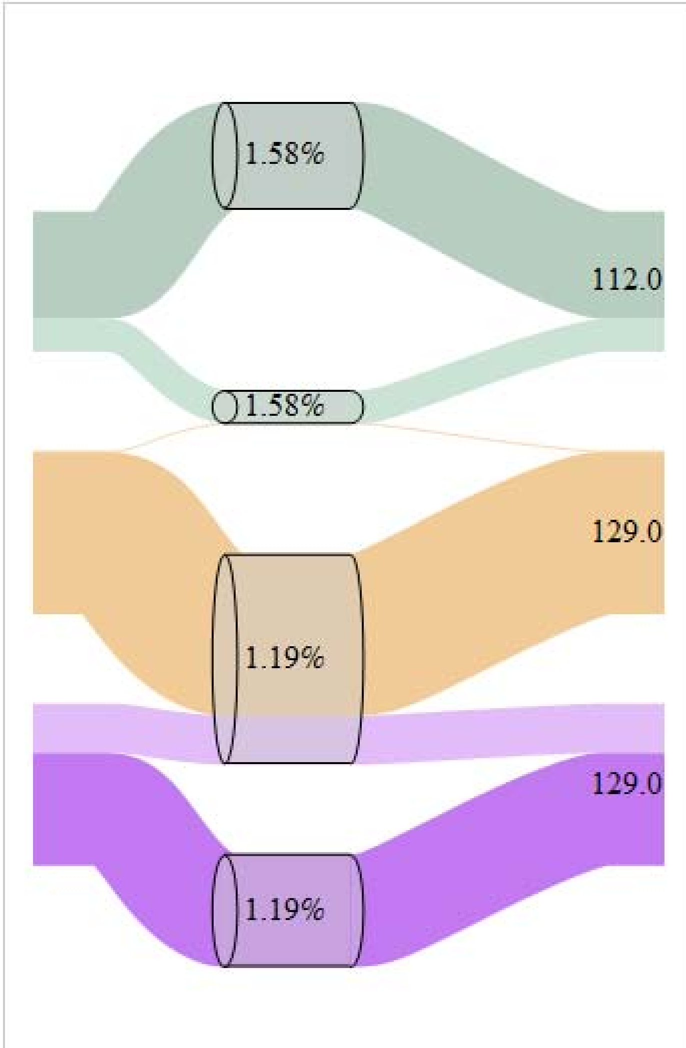
Congestion due to reduced capacity etc. can be shared across the entire network

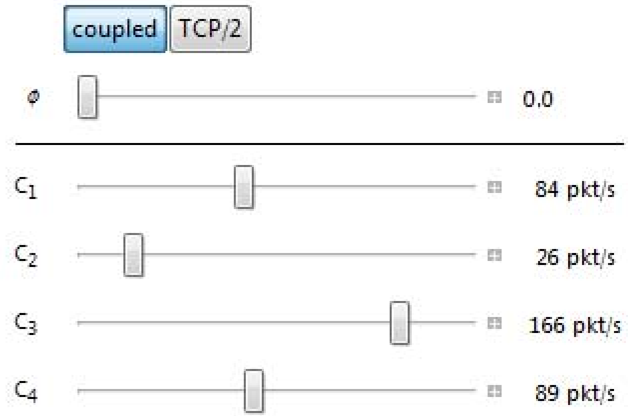




Resource pooling means:

Congestion due to reduced capacity etc. can be shared across the entire network

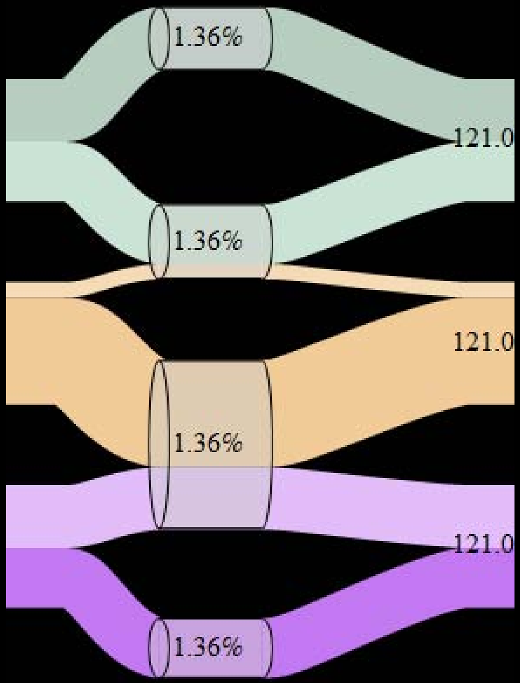




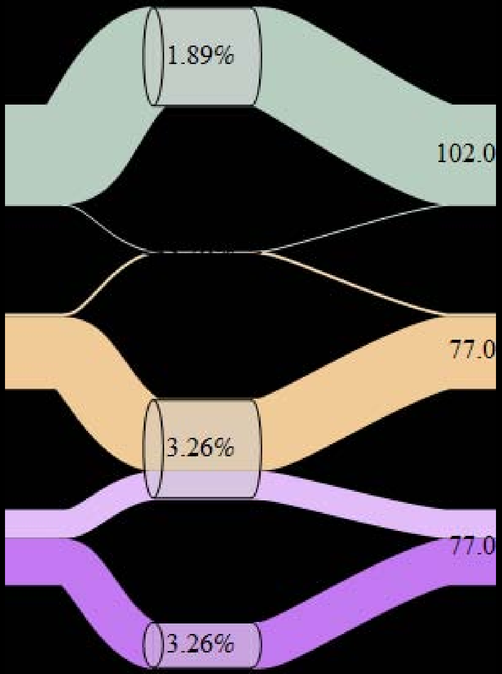
Resource pooling means:

Congestion due to reduced capacity etc. can be shared across the entire network

Resource pooling relies on there being enough path choices, and enough traffic that can make a choice.



There is enough diversity of useful paths to achieve complete resource pooling.

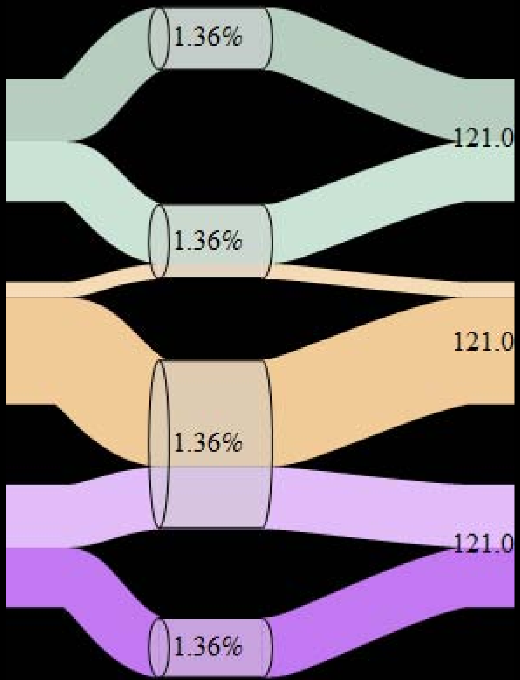


The network has split into two resource pools, because neither of the bottom two flows can access the top resource pool.

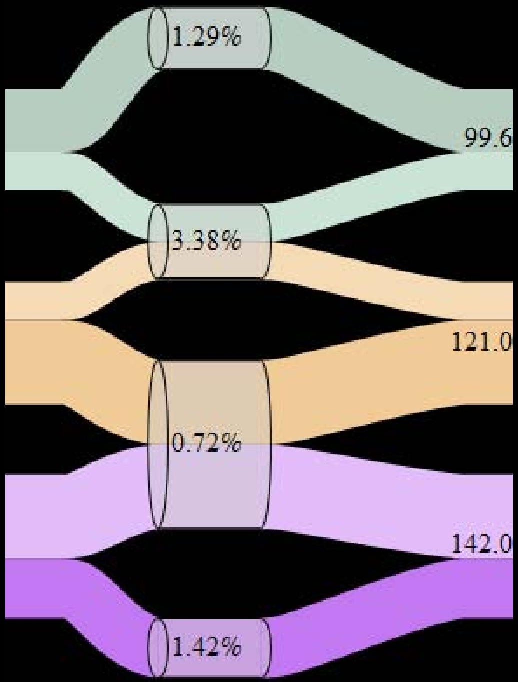
Topic II. How much resource pooling can be achieved, given a set of multipath routes and a traffic matrix?

Will there be one big pool, or many small pools?

Resource pooling relies on proper load-balancing by the end-systems.



Using an idealized coupled congestion controller, there is resource pooling



Using separate TCP controllers for each path, congestion is ***not*** *equalized and capacity is* ***not*** *shared*

Topic III. Can we design a congestion controller such that users react in the right way to achieve resource pooling?

If they don’t, there may be a single pool but it won’t be shared properly.

Topic II. How much resource pooling can be achieved, given a set of multipath routes and a traffic matrix?

For the purposes of network-wide resource pooling,

- Is it sufficient to use end-host addressing?
- How much path diversity is enough, and what sort of diversity is useful?

To answer this, we first need a metric for the amount of resource pooling that a network achieves.

How should we measure resource pooling? It means

“making a collection of resources behave like a single pooled resource”.

To measure resource pooling, we need to decide what we mean by “behave” and “like a single resource”.

How should we measure resource pooling? It means

"making a collection of resources behave like a single pooled resource".

To measure resource pooling, we need to decide what we mean by "behave" and "like a single resource".

"Behave"

Resource pooling has the consequence that congestion hotspots can be diffused across the network. So the behaviour I shall examine is "what is the change in congestion at a link, in response to a change in the capacity at that link?"

How should we measure resource pooling? It means

"making a collection of resources behave like a single pooled resource".

To measure resource pooling, we need to decide what we mean by "behave" and "like a single resource".

"Behave"

Resource pooling has the consequence that congestion hotspots can be diffused across the network. So the behaviour I shall examine is "what is the change in congestion at a link, in response to a change in the capacity at that link?"

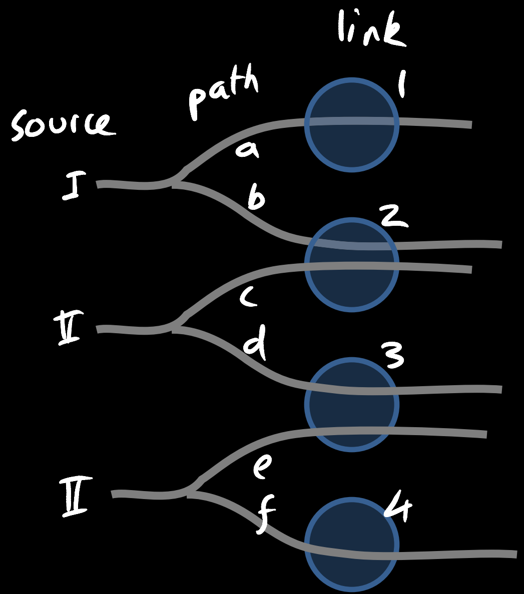
"Like a single resource"

Suppose for example that

- at an isolated link with capacity 100Mb/s, the loss of 50Mb/s increases packet loss by a factor of 20
- at an isolated link with capacity 1Gb/s, the loss of 50Mb/s increases packet loss by a factor of 1.03
- at a resource-pooling link with capacity 100Mb/s, the loss of 50Mb/s increases packet loss by a factor of 1.03

Then we'll say that the "effective pooled capacity at that link" is 1Gb/s.

A simple flow allocation problem



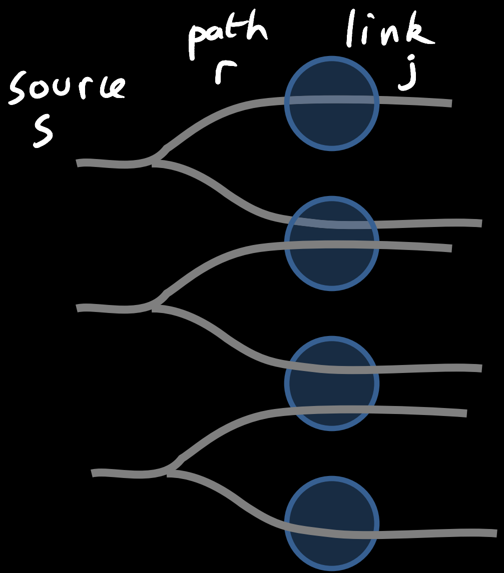
maximize $\quad U_I(y_I) + U_{II}(y_{II}) + U_{III}(y_{III})$

over $\quad x \geq 0,\ y \geq 0$

such that $\quad y_I = x_a + x_b,\quad y_{II} = x_c + x_d,\ \cdots$

$x_a \leq C_1,\quad x_b + x_c \leq C_2,\ \cdots$

A simple flow allocation problem (matrix form)



maximize $\sum_s U_s(y_s)$

over $x \geq 0, \; y \geq 0, \; z \geq 0$

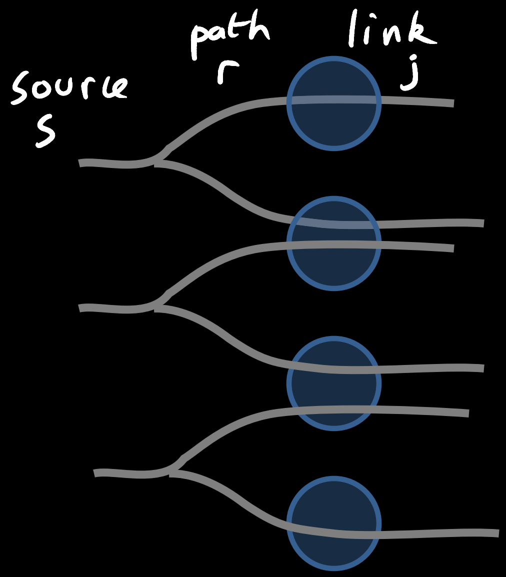
such that $y = Hx$

$z = Ax, \quad z \leq C$

$$H = \begin{array}{c} \\ I \\ II \\ III \end{array} \begin{array}{c} \begin{array}{cccccc} a & b & c & d & e & f \end{array} \\ \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \end{array}$$

$$A = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccccc} a & b & c & d & e & f \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \end{array}$$

A simple flow allocation problem (relaxed)



maximize $\sum_s U_s(y_s) - \sum_j c_j L_j(\rho_j)$

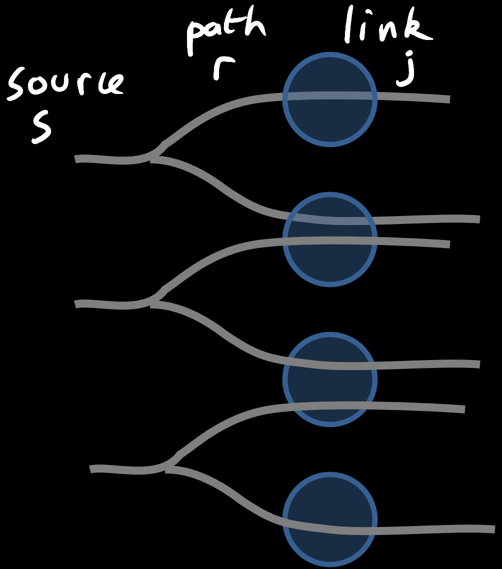
over $x \geq 0,\ y \geq 0,\ z \geq 0$

such that
$$y = Hx$$
$$z = Ax$$
$$\rho_j = z_j / c_j$$

where $L_j(\rho) = \int_0^{\rho} p_j(\rho)\, d\rho$

and $p_j(\rho)$ = packet drop probability at link j, when the load is ρ

We want to know how the solution changes when capacities change.
I shall take y to be fixed, and only look at how x changes.



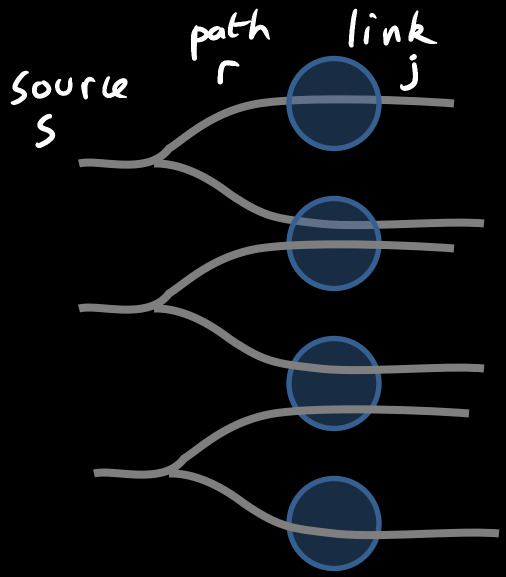
maximize $\sum_s U_s(y_s) - \sum_j c_j L_j(\rho_j)$

over $x \geq 0,\ y \geq 0,\ z \geq 0$

such that
$y = Hx$
$z = Ax$
$\rho_j = z_j / c_j$

where $L_j(\rho) = \int_0^\rho p_j(\rho)\, d\rho$
and $p_j(\rho)$ = packet drop probability at link j, when the load is ρ

We want to know how the solution changes when capacities change.
I shall take y to be fixed, and only look at how x changes.



$$\text{minimize} \quad \sum_j C_j L_j(\rho_j)$$

$$\text{over} \quad x \geq 0, \quad z \geq 0$$

$$\text{such that} \quad y = Hx$$

$$z = Ax$$

$$\rho_j = z_j / C_j$$

Write out the complementary slackness conditions

Take the total derivative with respect to C_j for some j

Solve for dz_i/dC_j using linear algebra

Theorem

At an isolated link, $\frac{dp}{dC} = \frac{p}{\tilde{C}}$ where $\tilde{C} = \frac{C}{L''(p)}$

In a network with idealized multipath congestion control

$$\frac{dp_j}{dC_j} = \frac{p_j}{\left(\frac{\tilde{C}_j}{1-\Psi_{jj}}\right)} \quad \text{where} \quad \tilde{C}_j = \frac{C_j}{L''(p_j)}$$

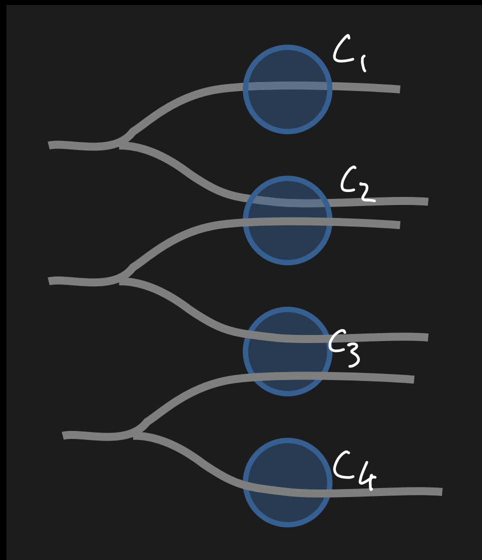
I call Ψ_{jj} the "poolability score", and $C_j/(1-\Psi_{jj})$ the "effective pooled capacity".

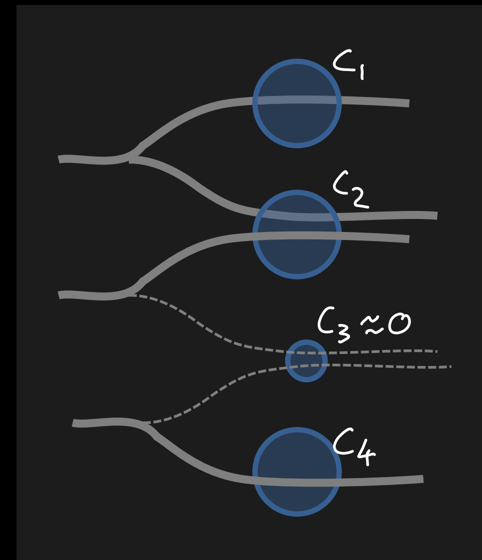
Here, $\begin{bmatrix} \Psi \\ \Phi \end{bmatrix} = \begin{bmatrix} \bar{A} & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} \bar{A}^T \tilde{C}^{-1} \bar{A} & -\bar{H}^T \\ H & 0 \end{bmatrix} \begin{bmatrix} \bar{A}^T \tilde{C}^{-1} \\ 0 \end{bmatrix}$

and $\tilde{C} = \begin{bmatrix} C_1/L_1''(p_1) & & 0 \\ & \ddots & \\ 0 & & C_n/L_n''(p_n) \end{bmatrix}$

and $\bar{A}$, $\bar{H}$ are the adjacency matrix and the source/path matrix, restricted to paths with non-zero traffic.

If the poolability score is $\Psi_{jj} \approx 1$ then the link sheds load easily.
If the poolability score is $\Psi_{jj} \approx 0$ then the link is "solitary".





$$\Psi_{11} = \frac{\tilde{c}_2 + \tilde{c}_3 + \tilde{c}_4}{\tilde{c}_1 + \tilde{c}_2 + \tilde{c}_3 + \tilde{c}_4}$$

$$\Psi_{22} = \frac{\tilde{c}_1 + \tilde{c}_3 + \tilde{c}_4}{\tilde{c}_1 + \tilde{c}_2 + \tilde{c}_3 + \tilde{c}_4}$$

$$\Psi_{33} = \frac{\tilde{c}_1 + \tilde{c}_2 + \tilde{c}_4}{\tilde{c}_1 + \tilde{c}_2 + \tilde{c}_3 + \tilde{c}_4}$$

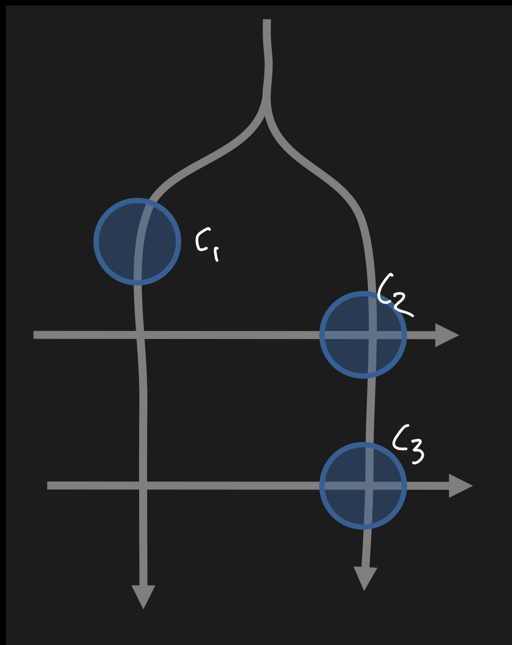
$$\Psi_{44} = \frac{\tilde{c}_1 + \tilde{c}_2 + \tilde{c}_3}{\tilde{c}_1 + \tilde{c}_2 + \tilde{c}_3 + \tilde{c}_4}$$

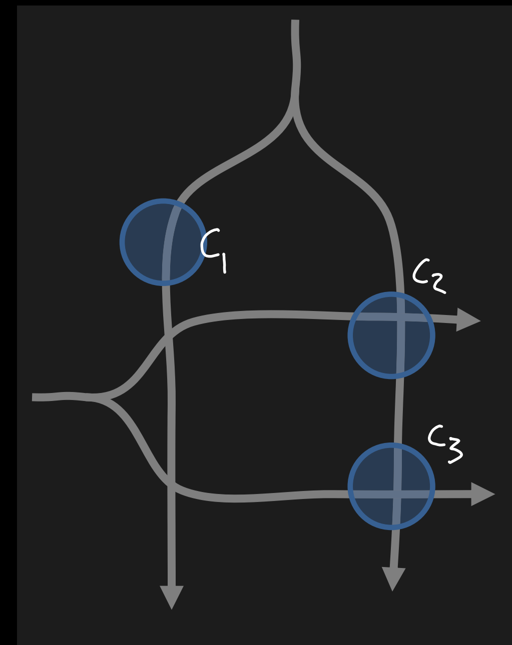
$$\Psi_{11} = \frac{\tilde{c}_2}{\tilde{c}_1 + \tilde{c}_2}$$

$$\Psi_{22} = \frac{\tilde{c}_1}{\tilde{c}_1 + \tilde{c}_2}$$

$$\Psi_{44} = 0$$

If the poolability score is $\Psi_{jj} \approx 1$ then the link sheds load easily.
If the poolability score is $\Psi_{jj} \approx 0$ then the link is "solitary".





$$\Psi_{11} = \frac{1/\tilde{c}_1}{1/\tilde{c}_1 + 1/\tilde{c}_2 + 1/\tilde{c}_3}$$

$$\Psi_{22} = \frac{1/\tilde{c}_2}{1/\tilde{c}_1 + 1/\tilde{c}_2 + 1/\tilde{c}_3}$$

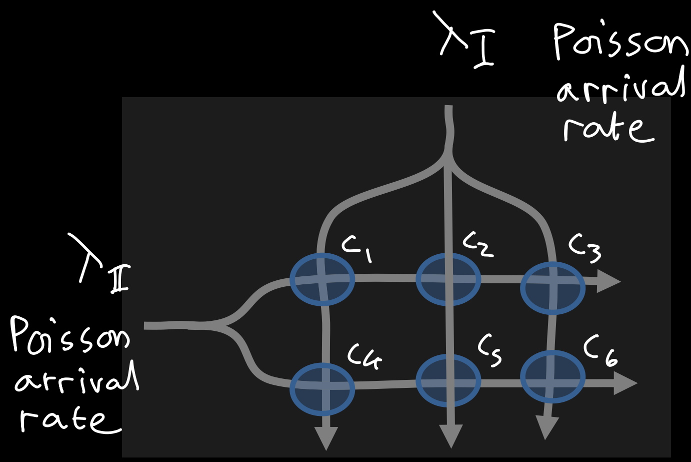
$$\Psi_{33} = \frac{1/\tilde{c}_3}{1/\tilde{c}_1 + 1/\tilde{c}_2 + 1/\tilde{c}_3}$$

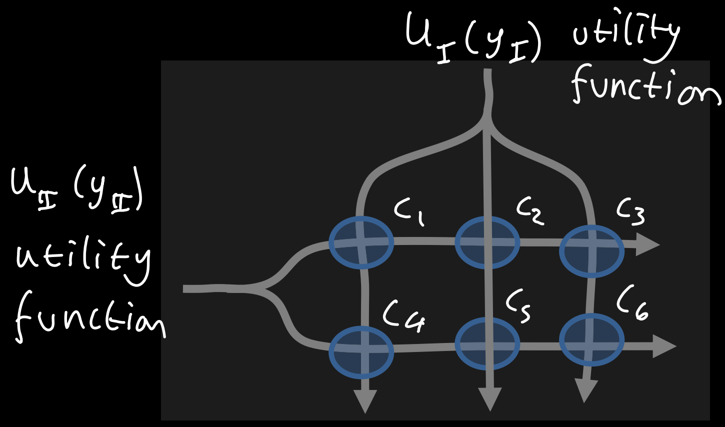
$$\Psi_{11} = \frac{\tilde{c}_2 + \tilde{c}_3}{4\tilde{c}_1 + \tilde{c}_2 + \tilde{c}_3} = \frac{1/\tilde{c}_1}{1/\tilde{c}_1 + 2/\tfrac{1}{2}(\tilde{c}_2 + \tilde{c}_3)}$$

$$\Psi_{22} = \frac{4\tilde{c}_1 + \tilde{c}_3}{4\tilde{c}_1 + \tilde{c}_2 + \tilde{c}_3}$$

$$\Psi_{33} = \frac{4\tilde{c}_1 + \tilde{c}_2}{4\tilde{c}_1 + \tilde{c}_2 + \tilde{c}_3}$$

There is a close link between the multi-commodity flow problem, and the multipath rate problem.





To check feasibility, we solve:

$$\begin{array}{ll} \text{minimize} & \delta \quad \text{over} \quad \delta \in \mathbb{R}, x \geq 0 \\ \text{such that} & Ax \leq C + \delta, \;\; Hx = y, \end{array}$$

which has dual:

$$\begin{array}{ll} \text{maximize} & \sum_s y_s q_s - \sum_j C_j p_j \quad \text{over} \quad p \geq 0, q \\ \text{such that} & \sum_j p_j = 1 \quad \text{and} \quad q_s \leq \min_{r \in s} \sum_{j \in r} p_j \text{ for all } s. \end{array}$$

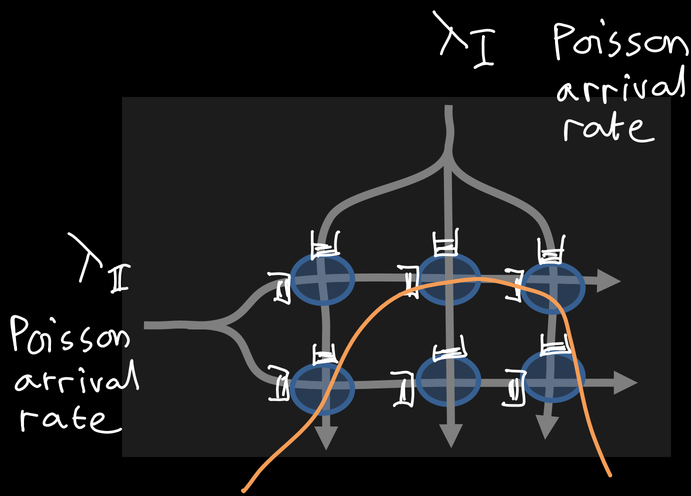
To find how flows balance themselves,

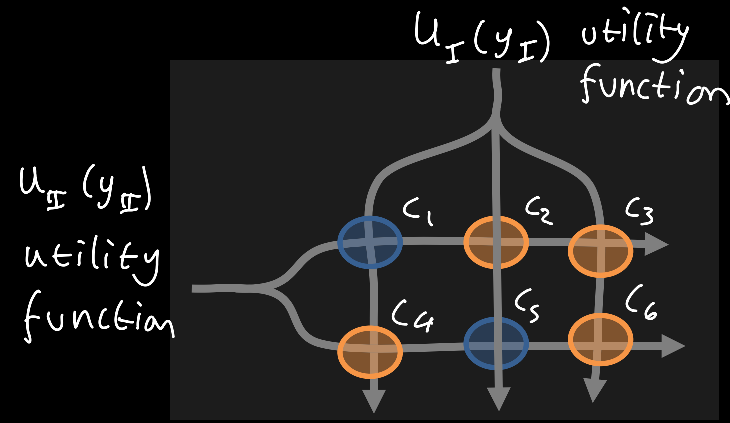
$$\begin{array}{ll} \text{minimize} & \sum_j C_j L_j(z_j/C_j) \quad \text{over} \quad x \geq 0, z \geq 0 \\ \text{such that} & Hx = y, \;\; Ax = z. \end{array}$$

which has dual:

$$\begin{array}{ll} \text{maximize} & \sum_s y_s q_s - \sum_j C_j L_j^*(p_j) \quad \text{over} \quad p \geq 0, q \geq 0 \\ \text{such that} & q_s \leq \min_{r \in s} \sum_{j \in r} p_j. \end{array}$$

There is a close link between the workloads in heavy traffic, and poolability.





Depending on λ_I and λ_{II}, and link capacities, we might in a heavy traffic queueing model find complete resource pooling, with workload

$$2Q_2 + 2Q_4 + Q_3 + Q_6 .$$

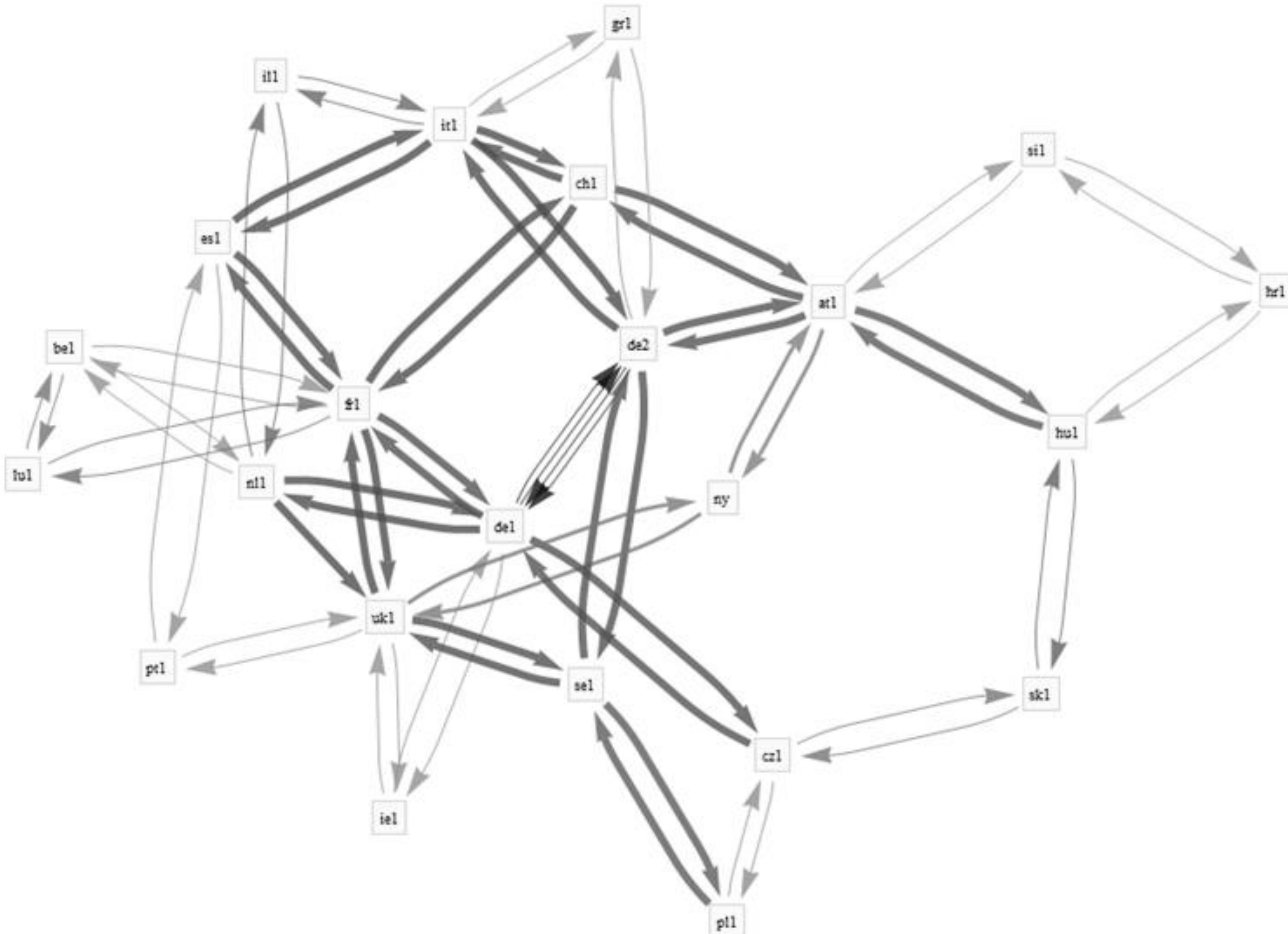
Other configurations are also possible.

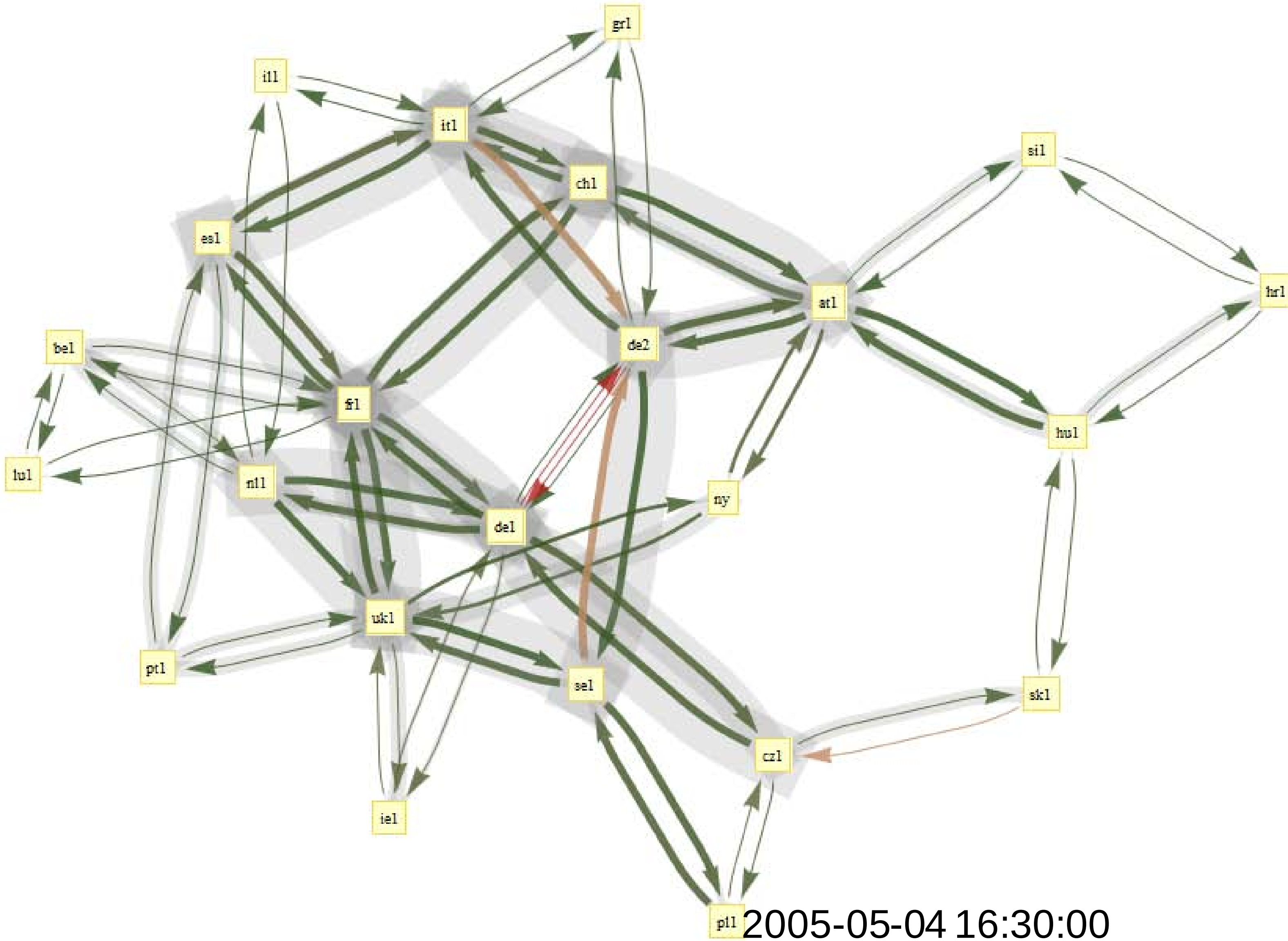
Depending on link capacities, we might get poolability scores

$$\Psi_{22} = \frac{4\tilde{c}_4 + \tilde{c}_3 + \tilde{c}_6}{4\tilde{c}_2 + 4\tilde{c}_4 + \tilde{c}_3 + \tilde{c}_6} \quad \text{etc.}$$

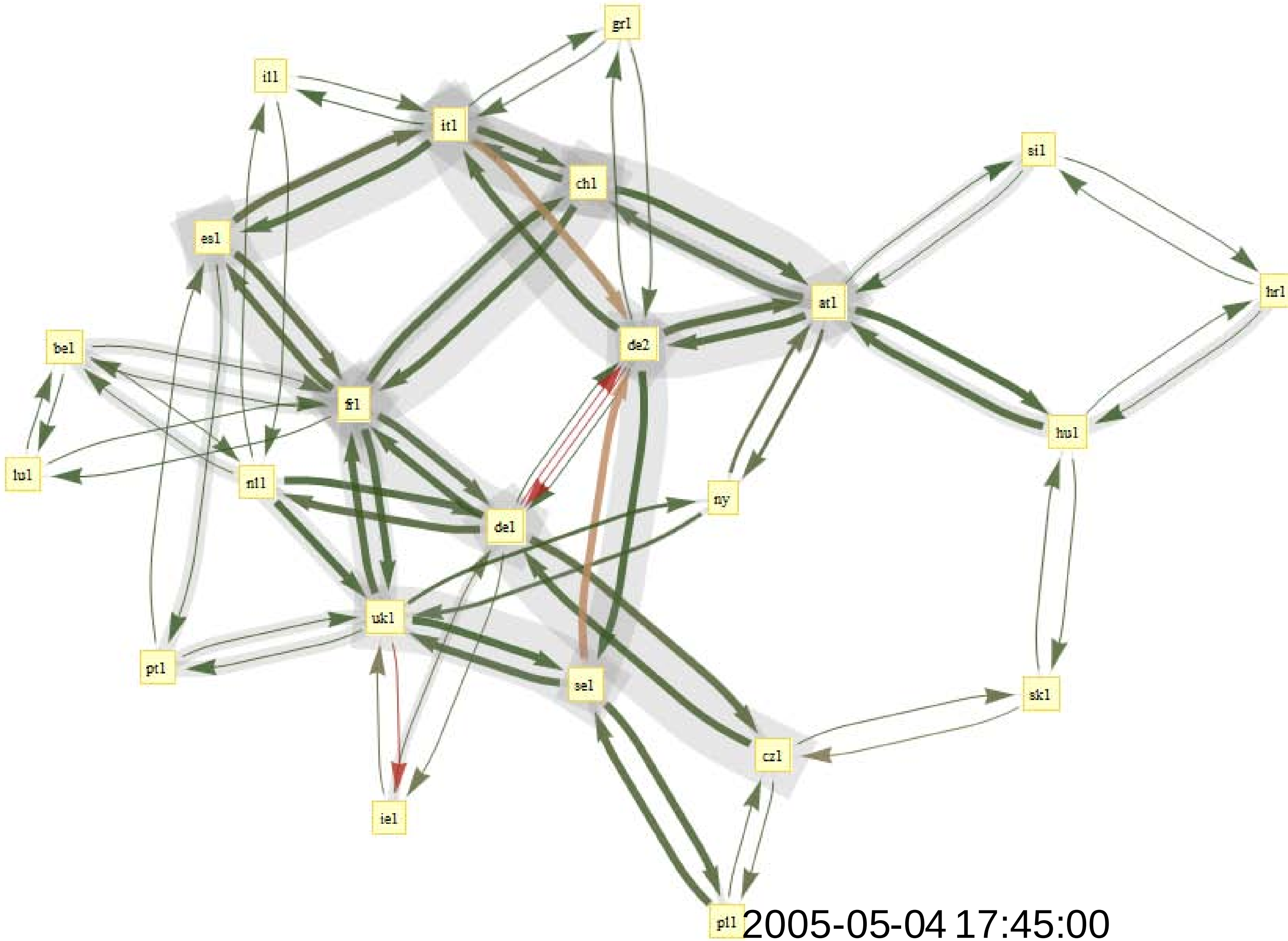
Other configurations are also possible.

GEANT data provided by UCL Belgium
multipath routes, link capacities, and traffic matrices

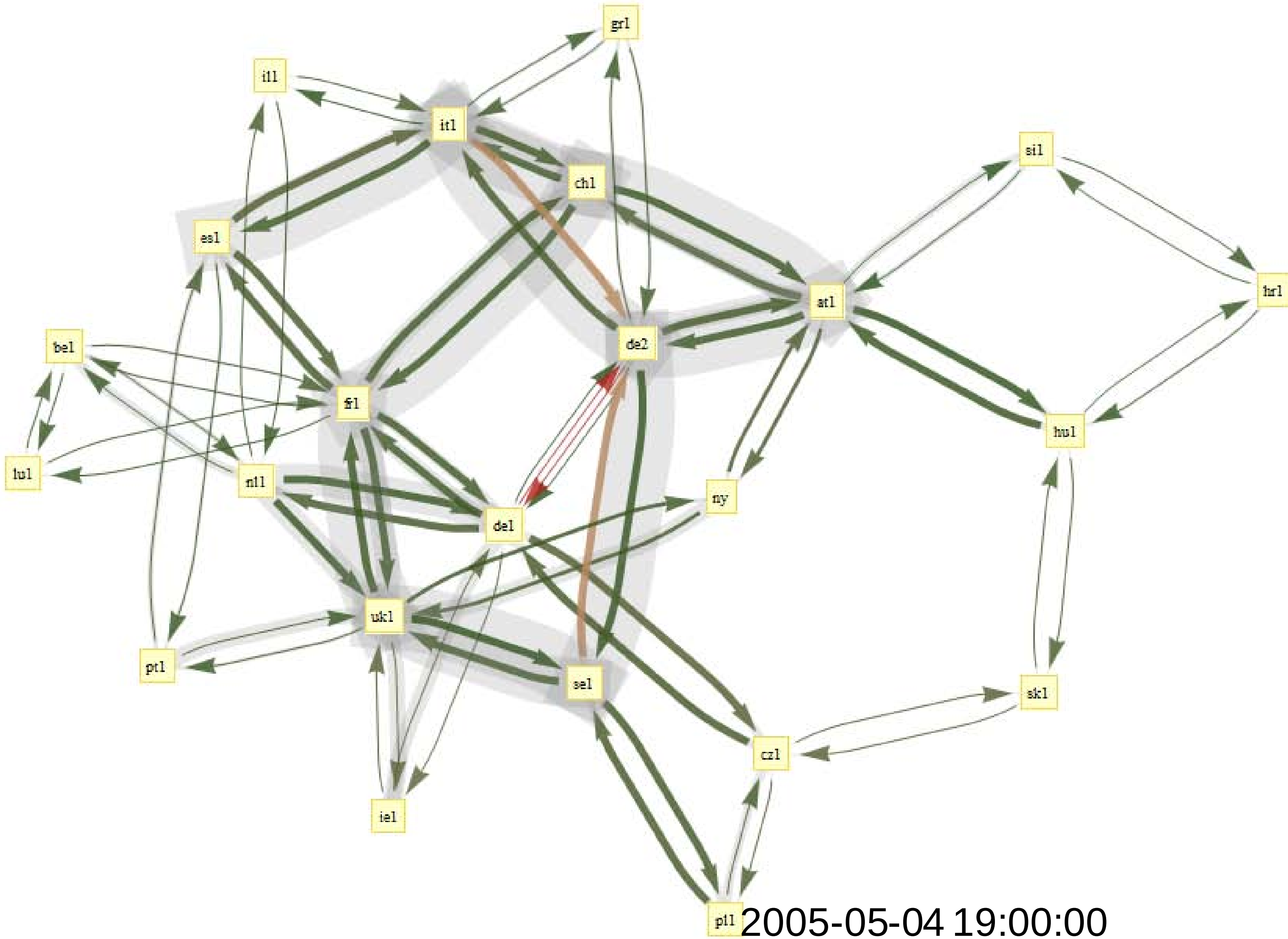




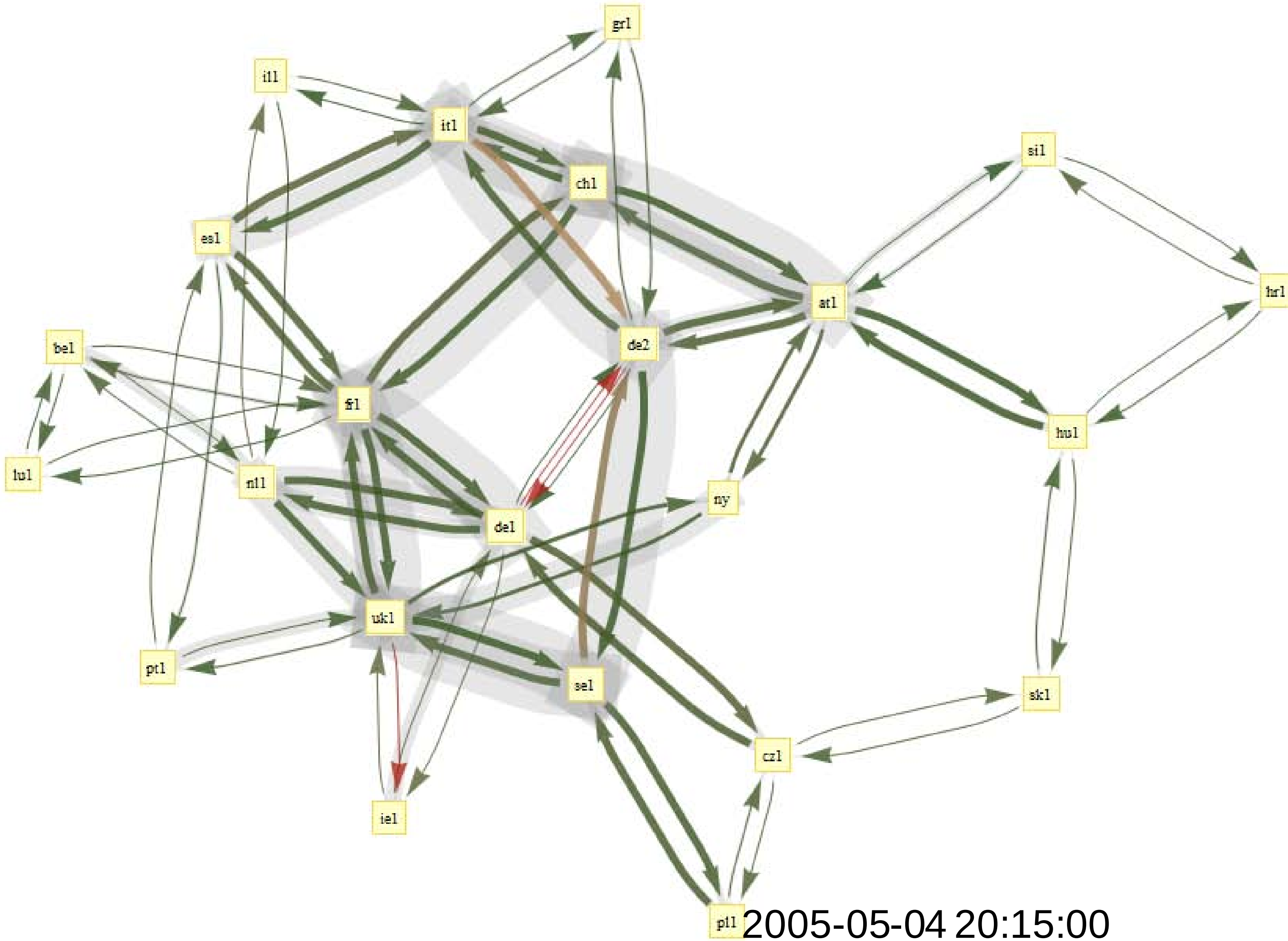
2005-05-04 16:30:00
Colours show utilization
Grey shows effective pooled capacity



2005-05-04 17:45:00
Colours show utilization
Grey shows effective pooled capacity



2005-05-04 19:00:00
Colours show utilization
Grey shows effective pooled capacity



2005-05-04 20:15:00
Colours show utilization
Grey shows effective pooled capacity

Topic III. Can we design a congestion controller such that users react in the right way to achieve resource pooling?

In the analysis of resource pooling, I assumed an idealized congestion controller: one which knows exactly the level of congestion on each path, and shifts its traffic onto the least congested.

To achieve this, we thought it would be a simple matter of taking a published "fluid model" of a load-balancing congestion controller, and implementing it.

[Kelly+Voice, 2005; Han, Shakkottai, Hollot, Srikant, Towsley (2006)]

$$\frac{d}{dt} x_r(t) = \frac{x_r(t-T_r)}{T_r} \Big(a\big(1-\lambda_r(t)\big) - b_r\, y_{s(r)}(t)\, \lambda_r(t) \Big)^{+[x_r(t)=0]}$$

Topic III. Can we design a congestion controller such that users react in the right way to achieve resource pooling?

In the analysis of resource pooling, I assumed an idealized congestion controller: one which knows exactly the level of congestion on each path, and shifts its traffic onto the least congested.

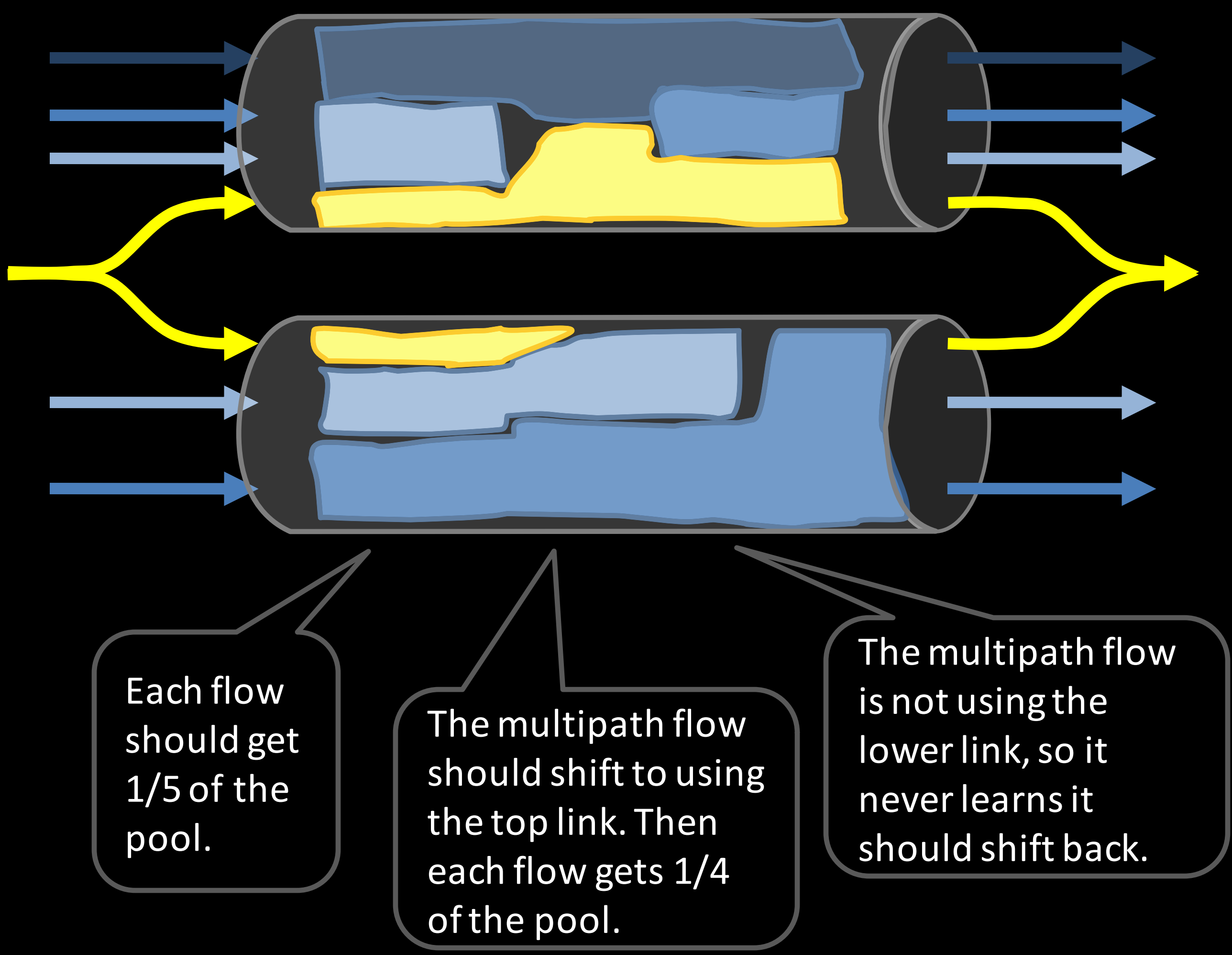
To achieve this, we thought it would be a simple matter of taking a published "fluid model" of a load-balancing congestion controller, and implementing it.

[Kelly+Voice, 2005; Han, Shakkottai, Hollot, Srikant, Towsley (2006)]

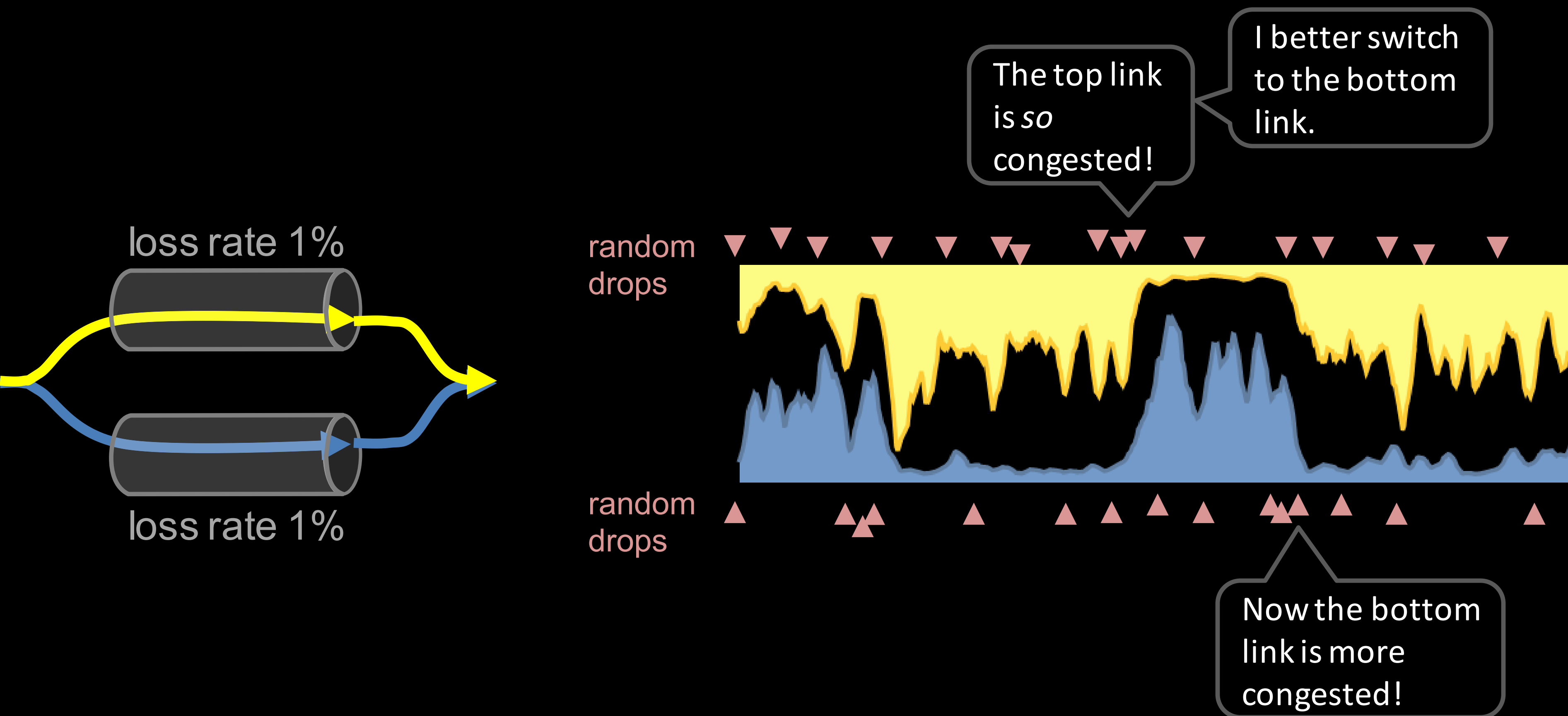
$$\frac{d}{dt} x_r(t) = \frac{x_r(t-T_r)}{T_r} \Big(a\big(1-\lambda_r(t)\big) - b_r\, y_{s(r)}(t)\, \lambda_r(t) \Big)^{+[x_r(t)=0]}$$

We were wrong.

The idealized congestion control algorithm puts *all* its traffic on the least congested path. This can a failure of load balancing, when congestion levels vary.



The noisy nature of congestion feedback makes it difficult to estimate congestion levels.



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

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

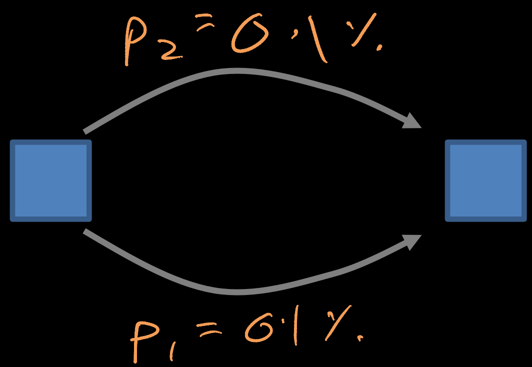
How we expect the fluid model to behave:

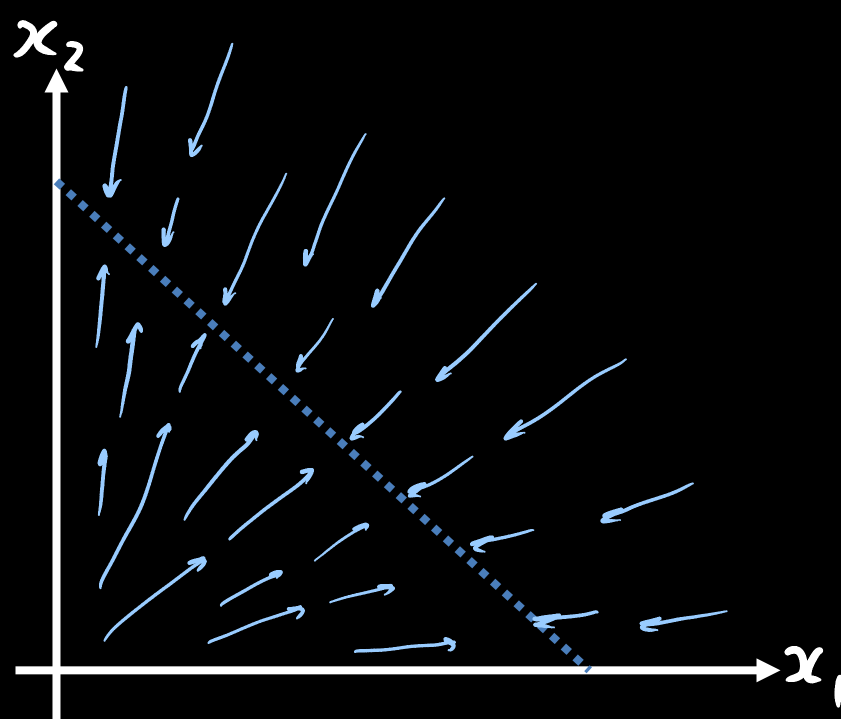
$$\frac{d}{dt} x_r(t) = k_r \left(x_r(t) - x_r(t)\, p_r(t)\, y(t) \right)$$

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

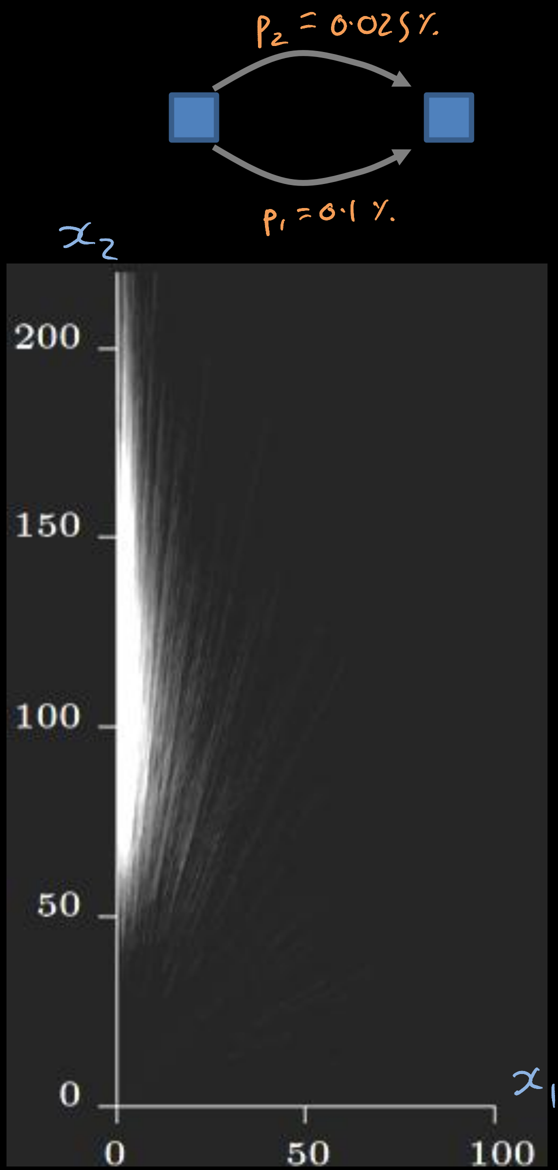
$y(t)$ = total rate over all flows for this user

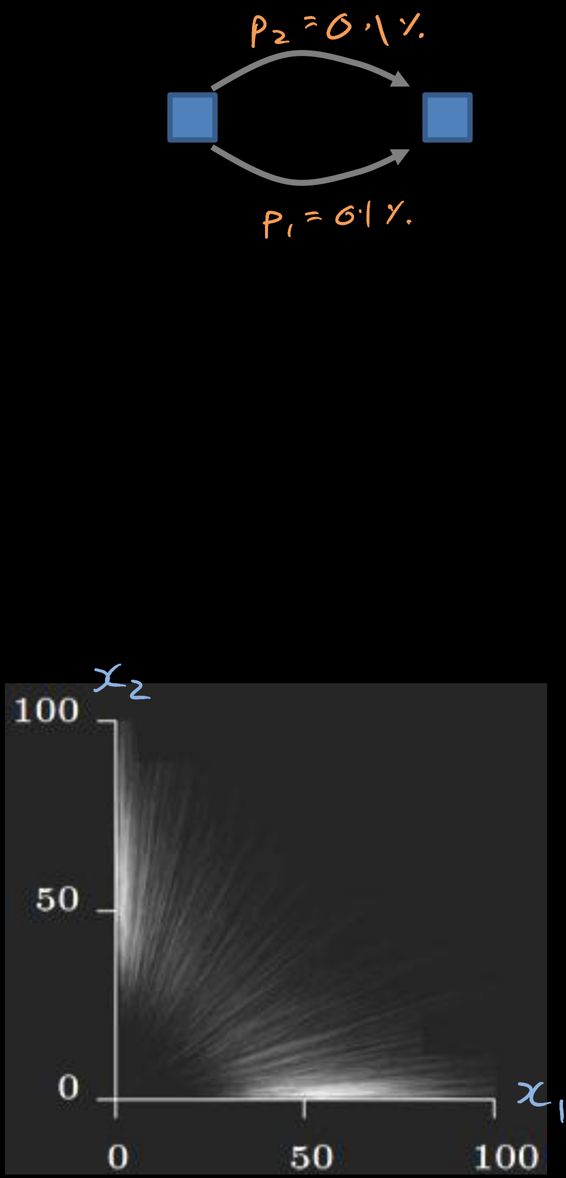
$p_r(t)$ = packet drop probability on path r





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 information feedback stream (packet drops, delays) is noisy. To get a good measure of the true state of the link, we have to average the signal.

But congestion is not static. To react promptly to changes in congestion, we have to look only at recent data about congestion, and we should constantly probe all paths.

The information feedback stream (packet drops, delays) is noisy. To get a good measure of the true state of the link, we have to average the signal.

But congestion is not static. To react promptly to changes in congestion, we have to look only at recent data about congestion, and we should constantly probe all paths.

The Zen of resource pooling

To pool resources effectively, the end-system should not try too hard to pool resources.

Instead, it should maintain **equipoise**, i.e. balance its traffic rate across its paths, *to the extent necessary* to achieve resource pooling.

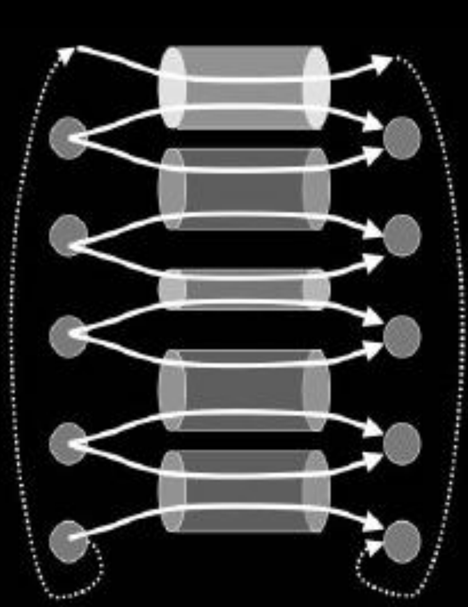
We devised a parameterized family of multipath congestion control algorithms, indexed by $\varphi \in [0,2]$, to investigate the tradeoff between load balancing and equipoise.

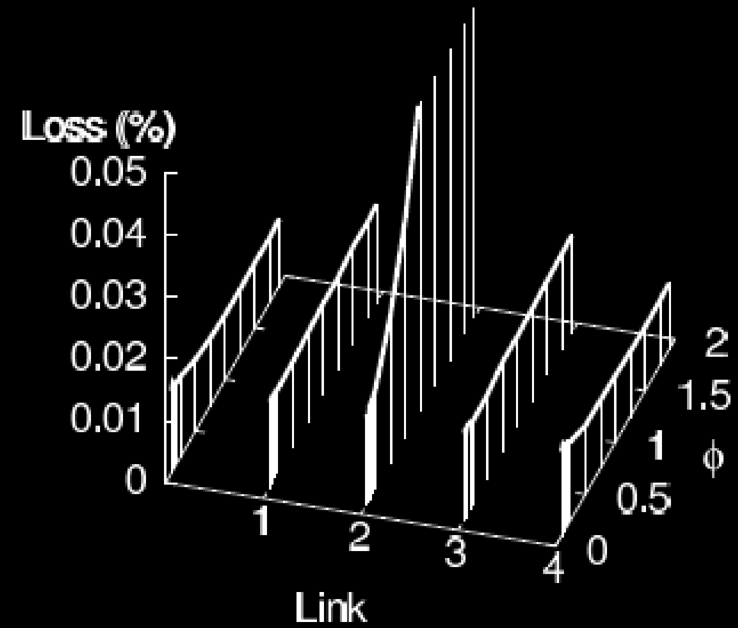
$\varphi=0$ | $\varphi=2$

the idealized congestion controller, inspired by Kelly+Voice

run independent TCP control on each path

How good is this congestion controller at achieving resource pooling, in a static network?



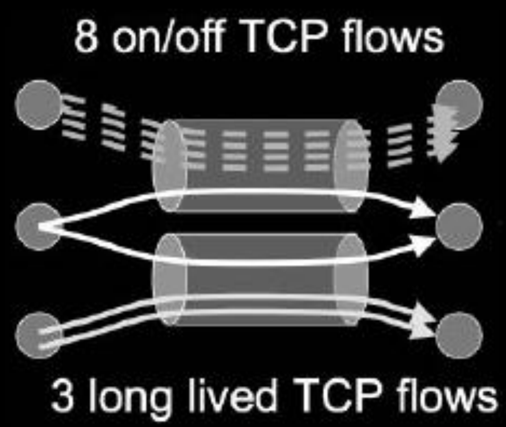


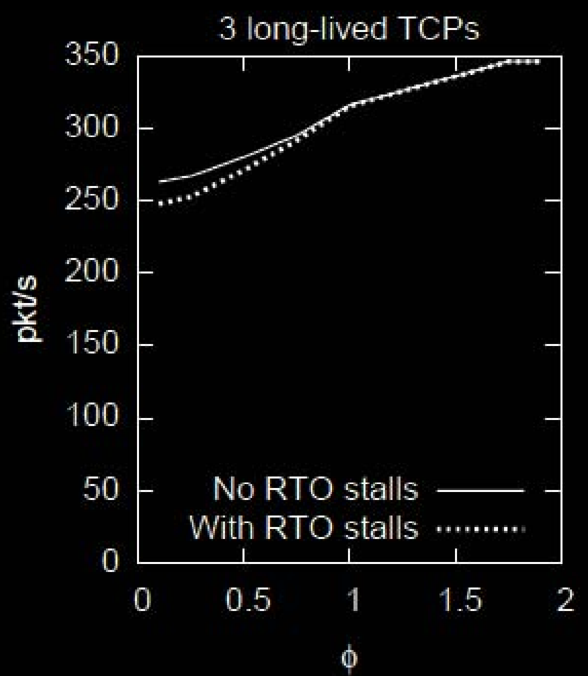
φ=0 φ=2

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

How good is this congestion controller at achieving resource pooling, in a dynamic network?





$\varphi=0$ ⟷ $\varphi=2$

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

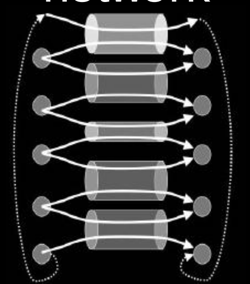
good at resource pooling:
constantly probes both links, so learns quickly when congestion levels change

the naïve coupled congestion controller, inspired by Kelly+Voice

run independent TCP control on each path

φ=0 φ=2

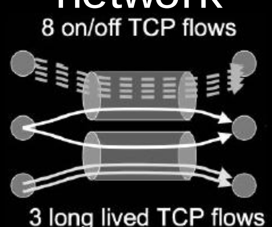
static network



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

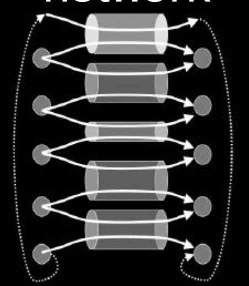
our choice

the naïve coupled congestion controller, inspired by Kelly+Voice

run independent TCP control on each path

φ=0 ⟷ φ=2

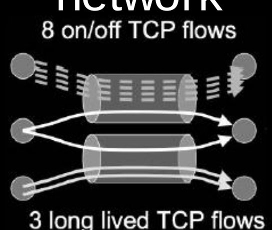
static network



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



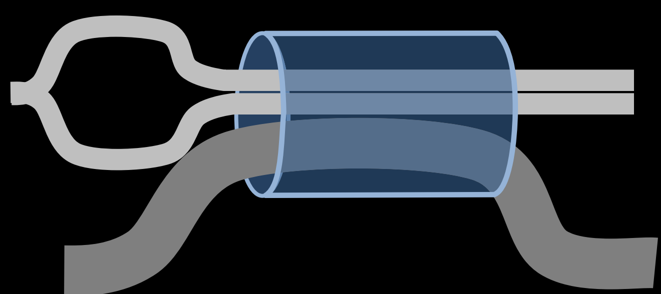
bad at resource pooling: shifts too enthusiastically to the less loaded link, and is slow to learn when the other link improves

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

We tweaked the φ algorithm, to ensure fairness with TCP.

We assign a weight to each link, and run a weighted version of the φ-algorithm. We have an adaptive algorithm for choosing the weights, to guarantee that

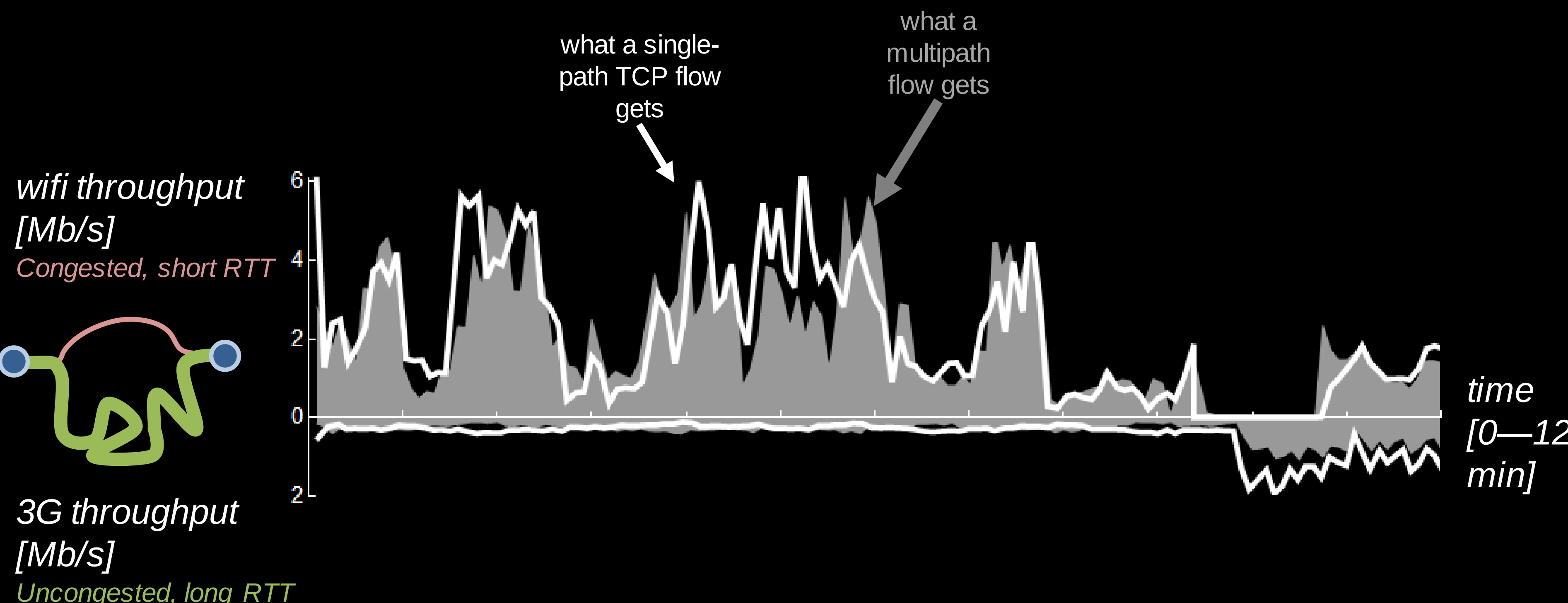
- the multipath user gets as least as much throughput as if he/she used the best single path
- the multipath user takes no more bandwidth on any link than a single-path TCP would.



The 3G link has lower drop probability. We'd prefer to use the 3G link, to get resource pooling.

But the 3G link has a long RTT, so single-path TCP gets low throughput. We shouldn't take any more than single-path TCP would.

Therefore we need to keep some traffic on the wifi link, so that the multipath user gets as good throughput as if he used single-path TCP.



Theorem

Let x_r be the fixed-point throughput on path r of our multipath algorithm, and let x_r^{TCP} be the throughput that a single-path TCP flow on that path. Assume that packet drop probabilities are given. Then

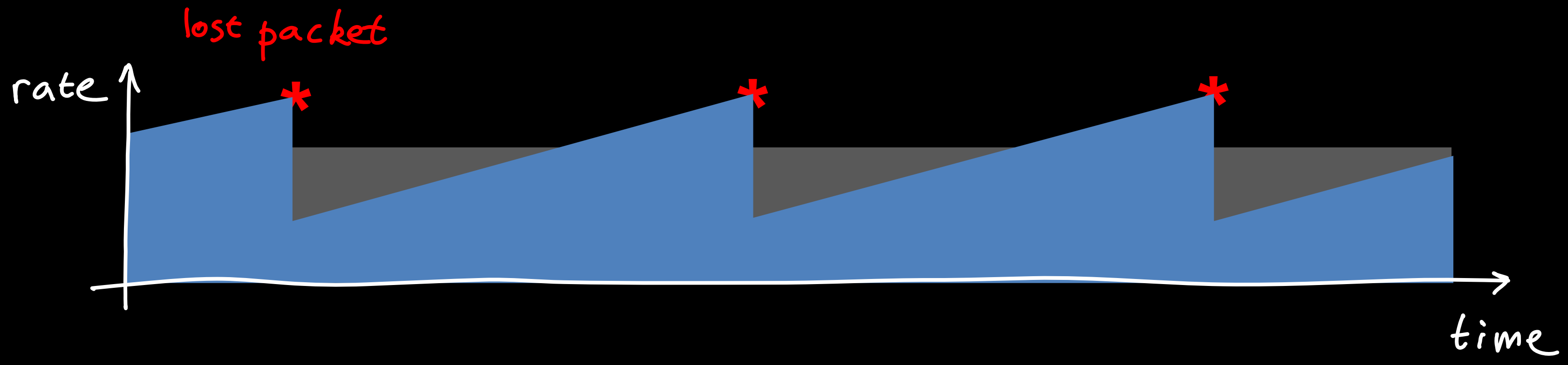
$$\sum_r x_r \geq \max_r x_r^{TCP}$$

$$\sum_{r \in S} x_r \leq \max_{r \in S} x_r^{TCP}$$ for all subsets S of paths

But is there a principled way to think about the congestion control problem?

But is there a principled way to think about the congestion control problem?

In the current Internet, the rate at which a source sends packets is controlled by TCP, the transmission control protocol of the Internet [12], implemented as software on the computers that are the source and destination of the data. The general approach is as follows [3]. When a resource within the network becomes overloaded, one or more packets are lost; loss of a packet is taken as an indication of congestion, the destination informs the source, and the source slows down. The TCP then gradually increases its sending rate until it again receives an indication of congestion. This cycle of increase and decrease serves to discover and utilize whatever bandwidth is available, and to share it between flows.



"Resource pricing and the evolution of congestion control", Gibbens and Kelly, 1999.

But is there a principled way to think about the congestion control problem?

In the current Internet, the rate at which a source sends packets is controlled by TCP, the transmission control protocol of the Internet [12], implemented as software on the computers that are the source and destination of the data. The general approach is as follows [3]. When a resource within the network becomes overloaded, one or more packets are lost; loss of a packet is taken as an indication of congestion, the destination informs the source, and the source slows down. The TCP then gradually increases its sending rate until it again receives an indication of congestion. This cycle of increase and decrease serves to discover and utilize whatever bandwidth is available, and to share it between flows.

dynamic programming!

$$F(n,p) = \lambda + \max_{u} \left\{ \mathbb{E}(u-D) - \gamma \mathbb{E}D + \mathbb{E}F(n',p') \right\}$$

This is the Bellman equation for a long-term average-cost dynamic programming problem.

Control: at what rate the user should send packets

State: the user's current belief about the network

Plant: Bayesian update of user's beliefs, based on acknowledgements and drops, and incorporating a preconceived notion of how quickly congestion levels might fluctuate

! Note: this equation is a toy model for single-path congestion control, not multipath.

$$F(n,p) = \lambda + \max_{u} \left\{ \mathbb{E}(u-D) - \gamma \mathbb{E} D + \mathbb{E} F(n', p') \right\}$$

We consider a model in which, each round trip time (RTT), the user chooses how many packets to send in that RTT. We assume $u \in \{0,1,\ldots,u_{max}\}$.

D is the number of dropped packets. The reward is *u*-*D*, the number of delivered packets. The cost is γD, for some constant $\gamma > 0$.

$$F(n,p) = \lambda + \max_{u} \left\{ \mathbb{E}(u-D) - \gamma\, \mathbb{E} D + \mathbb{E}\, F(n', p') \right\}$$

The distribution of *D* depends on the packet drop probability, *Q*.

The user's current Bayesian belief about *Q* is specified by a Beta distribution, parameterized by *n* and *p*. (Here, *p* is the expected drop probability and *n* is the "amount of evidence" for p.)

The user's belief about *q* is updated every RTT, in two ways:

the user gains information about the distribution of *Q*, from observing *D*

congestion levels may change over an RTT, which adds uncertainty to the distribution of *Q*. That is, the network is a *restless bandit*.

Assume that $D \sim \min(\text{Geom}(Q)-1, u)$

i.e. that packets are sent one by one, the drop probability is Q, and once one packet in an RTT is dropped all subsequent packets are dropped also.

Assume that $Q \sim \text{Beta}(n_0 p_0 + np,\ n_0(1-p_0) + n(1-p))$.

Here, n_0 and p_0 are prior belief about the network; in the absence of any new data, belief reverts to (n_0, p_0).

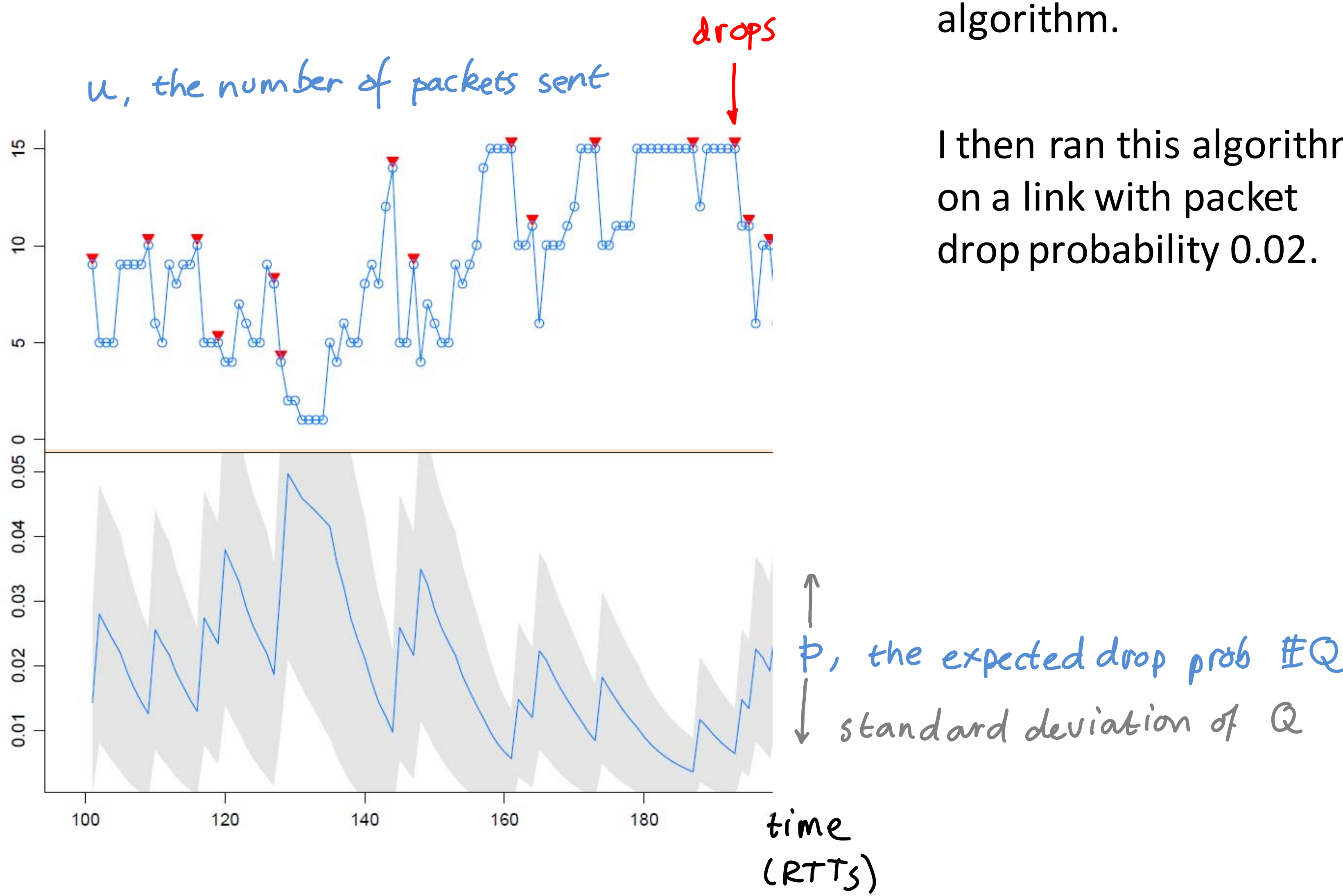
Belief about Q is updated by

$$n' = n\left(1 - \frac{RTT}{\tau}\right) + (u - D + 1) \wedge u$$

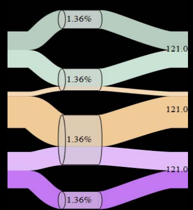
$$p' = \frac{n\left(1 - \frac{RTT}{\tau}\right)p + 1_{D>0}}{n'}.$$

I solved the Bellman equation numerically, and derived an optimal congestion control algorithm.

I then ran this algorithm on a link with packet drop probability 0.02.

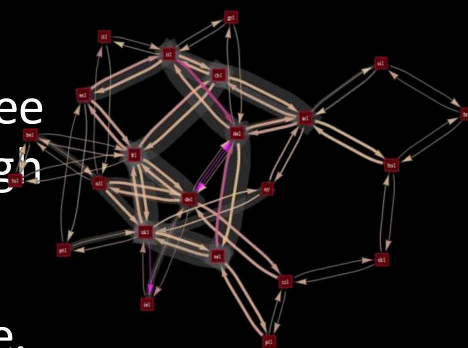


SUMMARY. We have a working implementation of multipath transport.

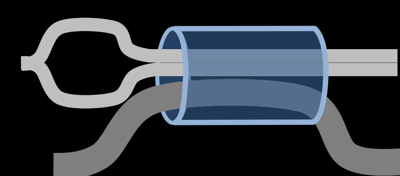


It achieves a reasonable degree of load balancing.

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.

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

Ongoing research topics

How can we use poolability scores to help design a multipath routing algorithm? Is it sufficient to rely on end-host addressing?

Can multipath TCP help achieve resource pooling in data centres?

Can multipath TCP make good routing choices in ad-hoc wireless networks?

Does the dynamic programming approach shed light on CUBIC, Compound TCP etc.? Why has classic TCP worked so well?

What is the impact of resource pooling on competition and pricing? Will it drive network operators to switch to congestion volume pricing?