# Models of multipath resource allocation

Damon Wischik, UCL

- What is the capacity of a network in which traffic flows can be split over multiple paths?
→ Generalized cut constraints

- What sort of rate-control algorithms might make full use of this capacity?
→ Resource pooling; fairness and stability

- What routing support does there need to be, in order to make use of the capacity?
→ Power of two choices

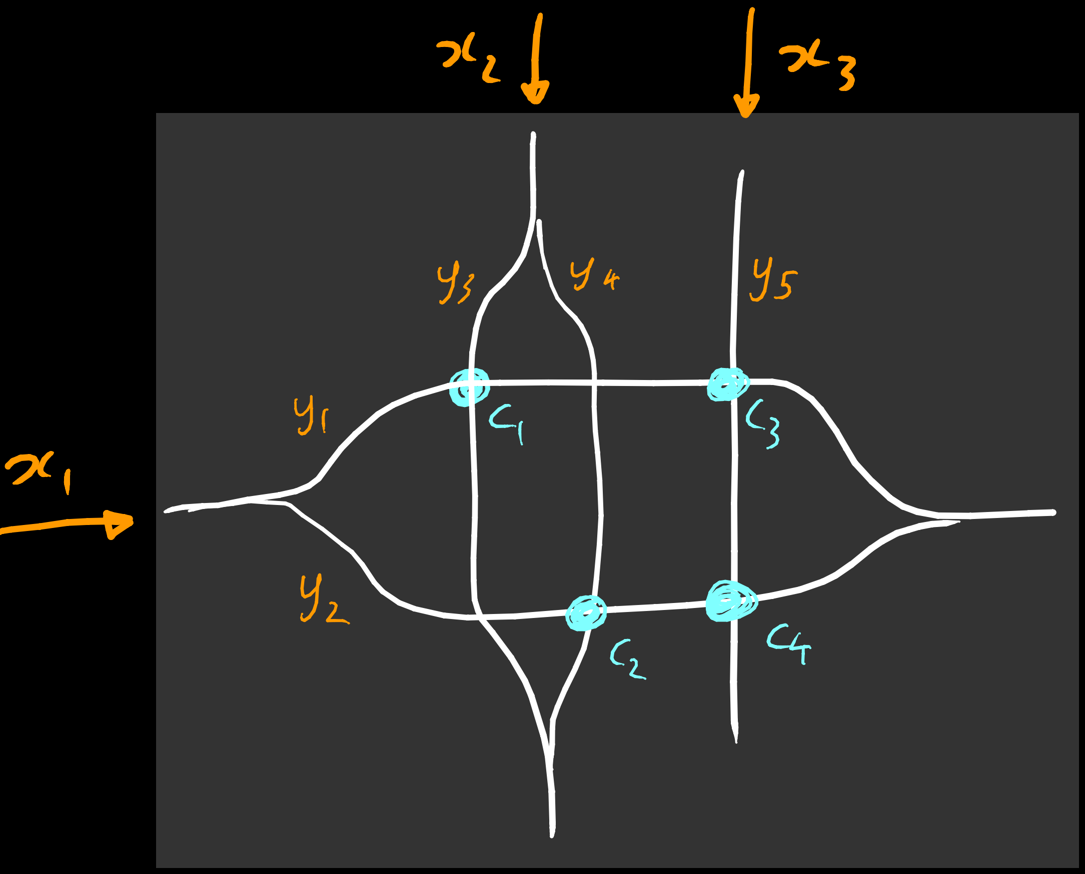
- Is there a conflict between routing at the end-systems, and traffic engineering in the network?
→ Wardrop equilibrium

# I. Generalized cut constraints

“Resource pooling in queueing networks with dynamic routing”, Laws, 1992

“Loss Networks”, Kelly, 1991

# The multipath rate allocation problem



$i \in \{1,2,3\}$ are the flows

$x_i$ is the arrival rate of flow $i$

$j \in \{1,2,3,4\}$ are the resources

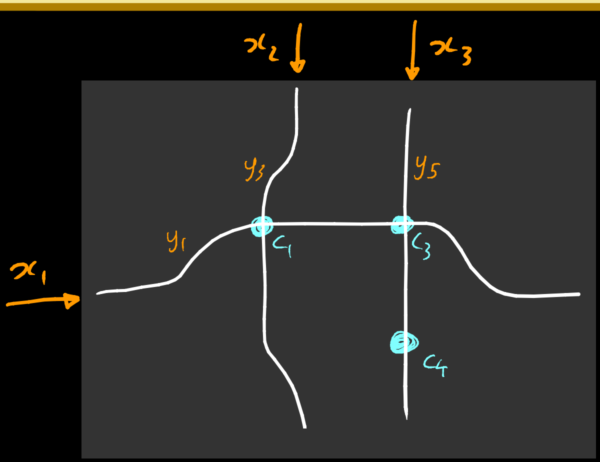
$C_j$ is the capacity at resource $j$

$p \in \{1,\dots,5\}$ are the paths

$Y_p$ is the rate allocated to path $p$

For what arrival rates $x$ is it possible to allocate rates $y$ without overloading the resources?

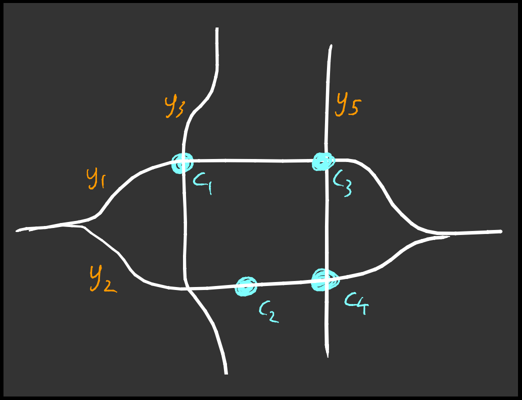
# Generalized cut constraints



$$x_1+x_2 \leq C_1$$

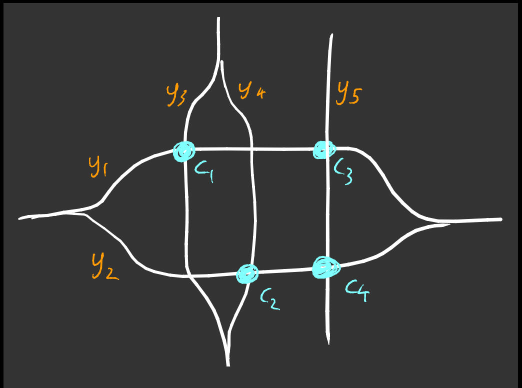
$$x_1+x_3 \leq C_3$$

$$x_3 \leq C_4$$



$$x_1+x_2+x_3 \leq C_1+C_4$$

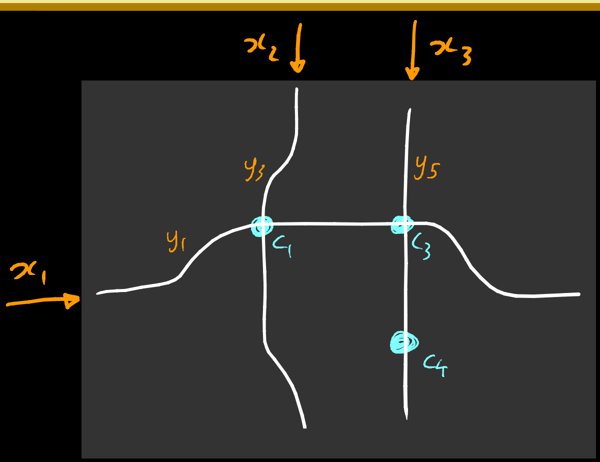
$$x_1+2x_3 \leq C_3+C_4$$



$$x_1+x_2 \leq C_1+C_2$$

$$x_1+2x_3 \leq C_3+C_4$$

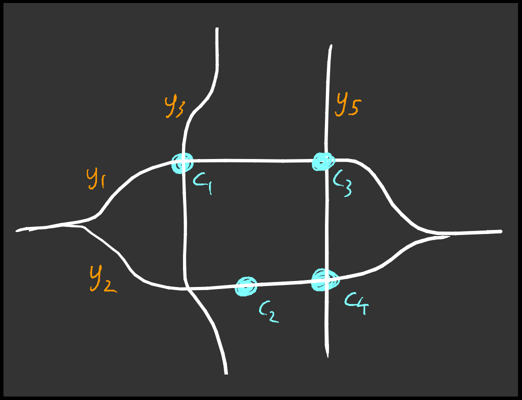
# Generalized cut constraints



$x_1 + x_2 \le C_1$

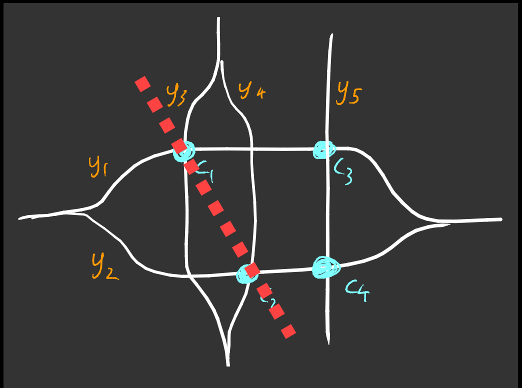
$x_1 + x_3 \le C_3$

$x_3 \le C_4$



$x_1 + x_2 + x_3 \le C_1 + C_4$

$x_1 + 2x_3 \le C_3 + C_4$



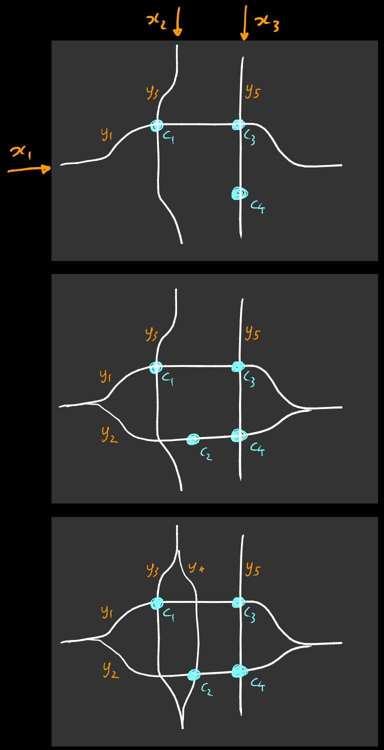
a cut constraint

$x_1 + x_2 \le C_1 + C_2$

$x_1 + 2x_3 \le C_3 + C_4$

a generalized cut constraint

# Generalized cut constraints



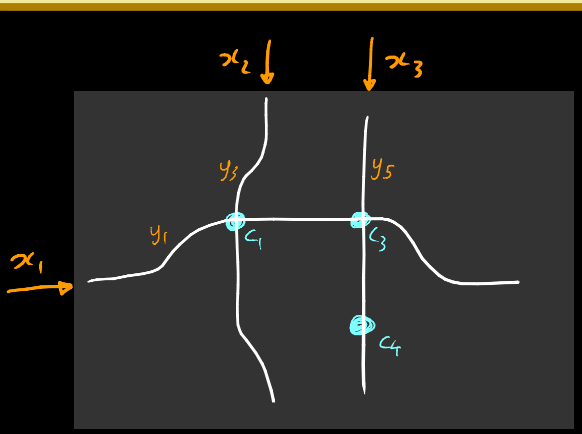
- **Theorem.** The set $\mathcal{X}$ of admissible arrival rates can always be written

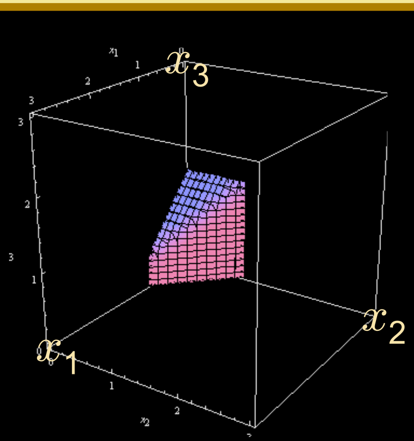
$$\mathcal{X} = \{x : \lambda(k)\cdot x \le \mu(k)\cdot C \text{ for all } k\}$$

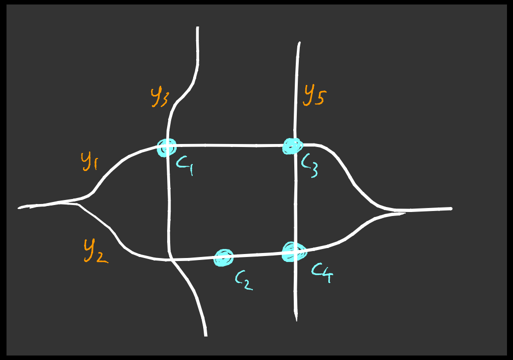
for a suitable set of variables $\lambda(k)$ and $\mu(k)$

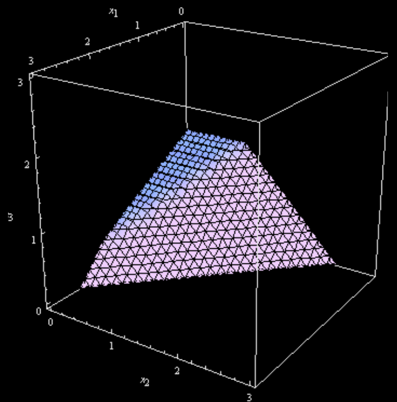
- These constraints are called generalized cut constraints, corresponding to virtual resources
- Consider the linear program: “what is the least amount of extra capacity I need to add to be able to serve $x$?” The generalized cut constraints are the extreme points of the dual feasible set.
- There are algorithms for computing the generalized cut constraints

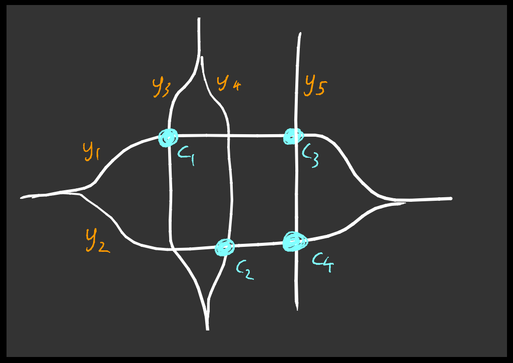
# Capacity region

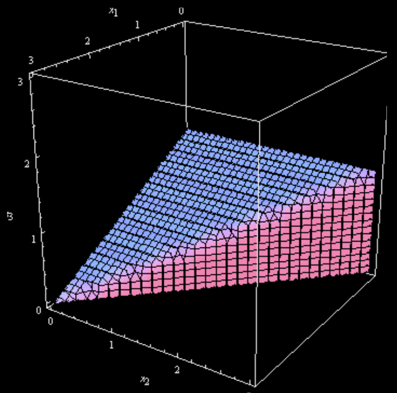












These plots show the capacity region, i.e. the set of admissible arrival rates.

Multipath will increase the capacity region. By how much? For which flows does multipath bring the greatest benefit?

# Questions

Q

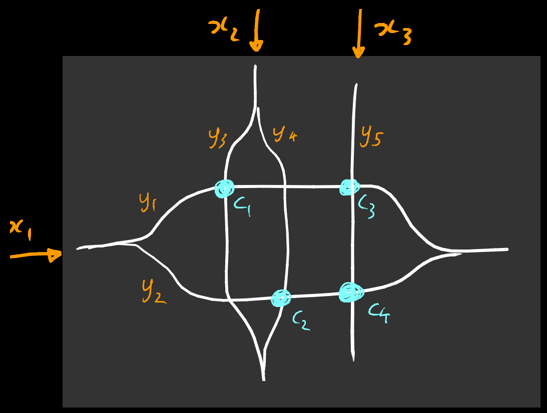
- Given Internet topology, what multipaths do we need to ensure that the generalized cut constraints are as broad as possible?
  - e.g. if each flow could have only two paths, what should they be?
  - if you had complete flexibility, how big is the admissible region?

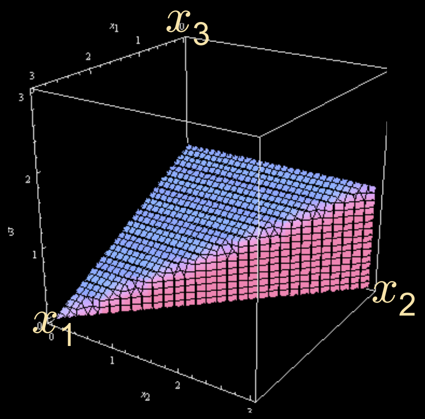
# II. Resource pooling

“State space collapse and diffusion approximation for a network operating under a proportional fair sharing policy”, Kang, Kelly, Lee, Williams

“Heavy traffic analysis of optimal scheduling algorithms for switched networks”, Shah, Wischik

# "No constraints but the network itself"





$x_1 + x_2 \leq C_1 + C_2$

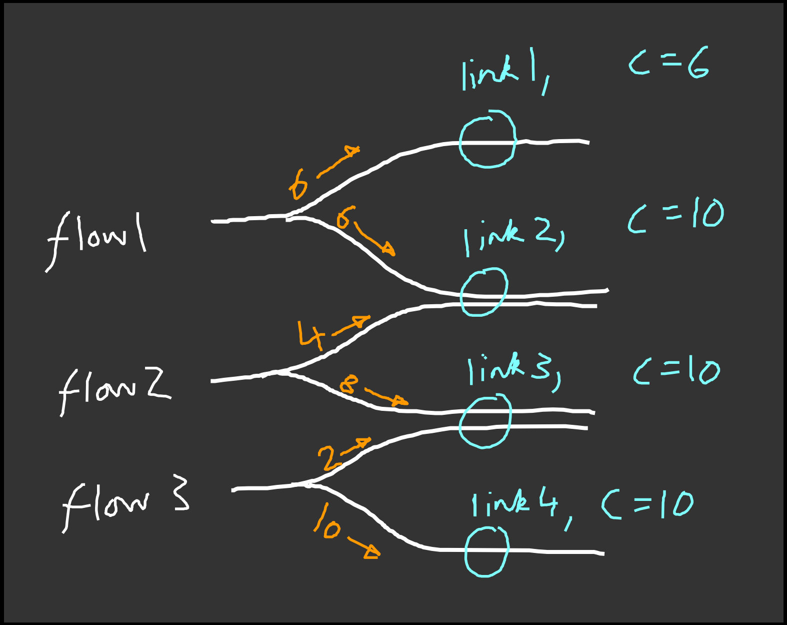
$x_1 + 2x_3 \leq C_3 + C_4$

What are the characteristics of a good rate allocation algorithm?

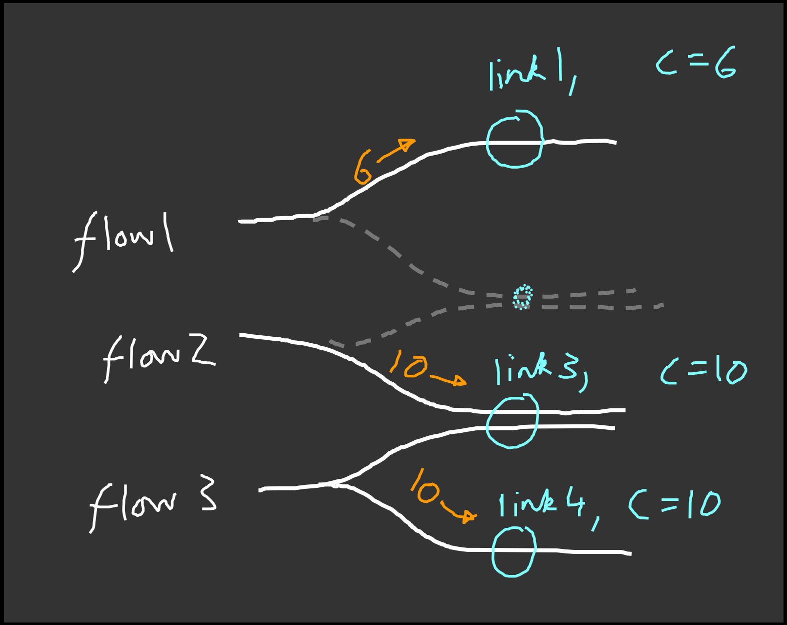
A good algorithm will have decent performance whenever the arrival rates are admissible.

It will not add extra constraints. It will only 'see' the generalized cut constraints.

# Example of resource pooling



*The total capacity of the four links is 36, and it can be evenly shared between the flows, even though none of the flows can balance its load across all four links. This is called **resource pooling**.*

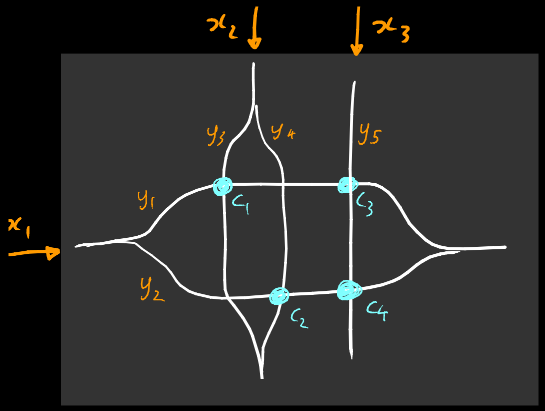


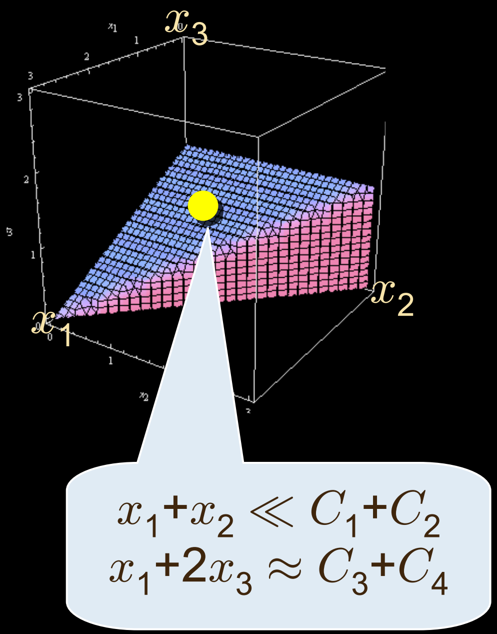
*When a link fails, the flows should automatically re-balance themselves so that service is maintained. Link failures will mean there is less path diversity, and this may break resource pooling.*

# Revision of basic queueing theory

- Consider a M/G/1 processor sharing queue
- Jobs arise as a Poisson process, and job size distribution is arbitrary
- Let the utilization be $\rho$
- Then the number of active jobs has a geometric distribution with mean $\rho/(1-\rho)$

# A critically loaded network



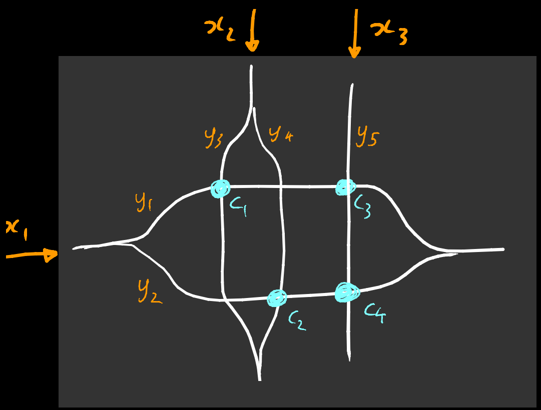


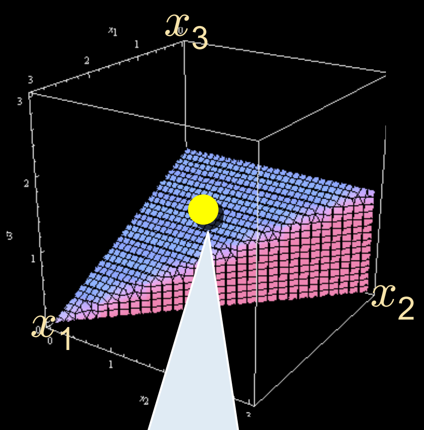
$x_1$+$x_2$ ≪ $C_1$+$C_2$
$x_1$+2$x_3$ ≈ $C_3$+$C_4$

Let $x_1$, $x_2$, $x_3$ be arrival rates for the arrival of TCP sessions, assumed to be Poisson arrivals. Let $C_1$,…,$C_4$ be link speeds divided by mean flow size.

Suppose that one generalized cut constraint is tight, while the other is not.

# Virtual queues at virtual resources





$x_1$+$x_2 \ll C_1$+$C_2$

$x_1$+2$x_3 \approx C_3$+$C_4$

Define a virtual queue $V$ —

- For arrivals of type 1, put in one token
- Every arrival of type 3, put in two tokens
- Serve this queue at rate $C_3$+$C_4$

Define the workload $W$ for the critical constraint —

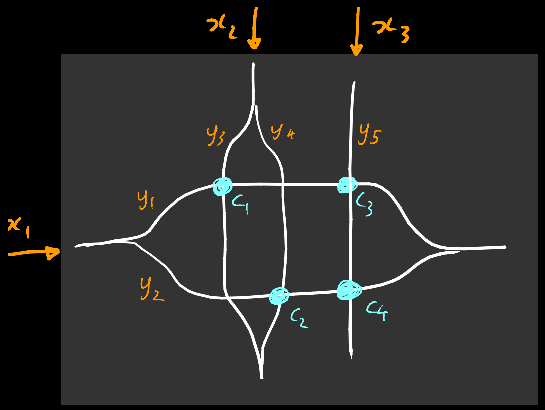
- the number of jobs of type 1,
- + 2 x number of jobs of type 3

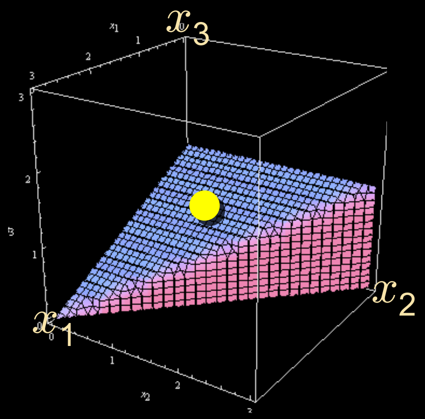
Then $W \geq V$. A perfect algorithm would achieve $W$=$V$, by ensuring that neither resource 3 nor resource 4 is underutilized whenever $V$ is non-empty. If either of these becomes underutilized then $W$>$V$.

A modified TCP achieves $W$=$V$. In this case $V$ has a geometric distribution, and there are

- $\approx$ [$x_1$/($x_1$+2$x_3$)] $W$ jobs on route 1
- $\approx$ 0 jobs on route 2
- $\approx$ [$x_1$/($x_1$+2$x_3$)] $W$ jobs on route 3

# Resource pooling





In effect, the system only ‘sees’ the virtual resources, and each of them behaves like a standard processor-sharing link. This is as good as possible.

To achieve this, whenever a virtual resource is non-empty none of the real resources that comprise it should go idle.

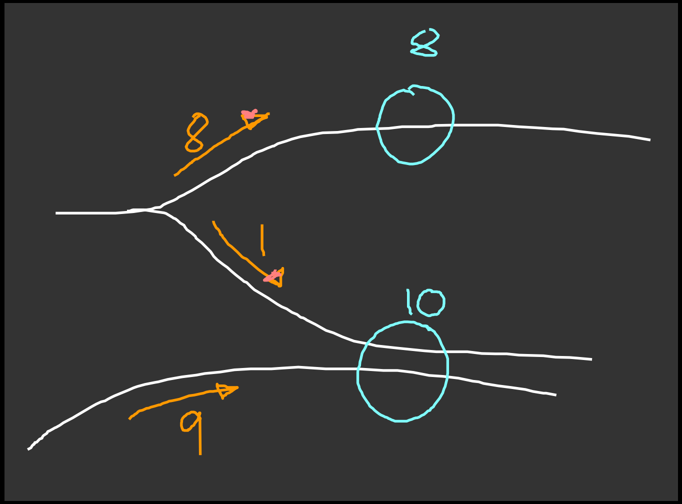
This is called resource pooling.

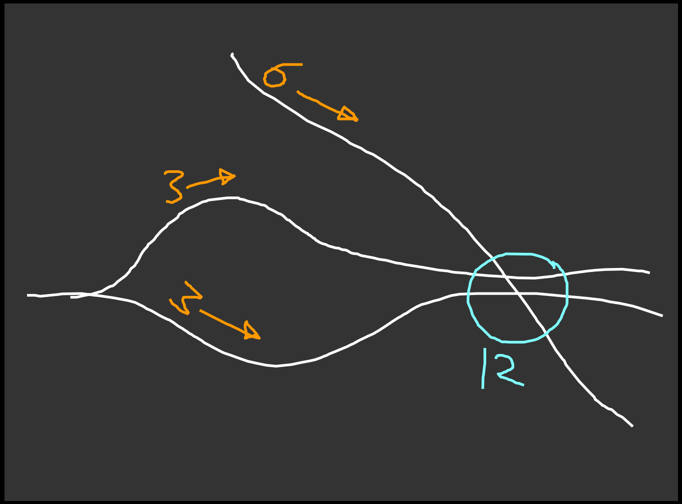
(In this example, resource pooling is a consequence of the fact that one virtual resource is heavily loaded. This is *heavy traffic theory*.)

# III. Fairness and stability of congestion control

“Stability of end-to-end algorithms for joint routing and rate control”, Kelly, Voice, 2005

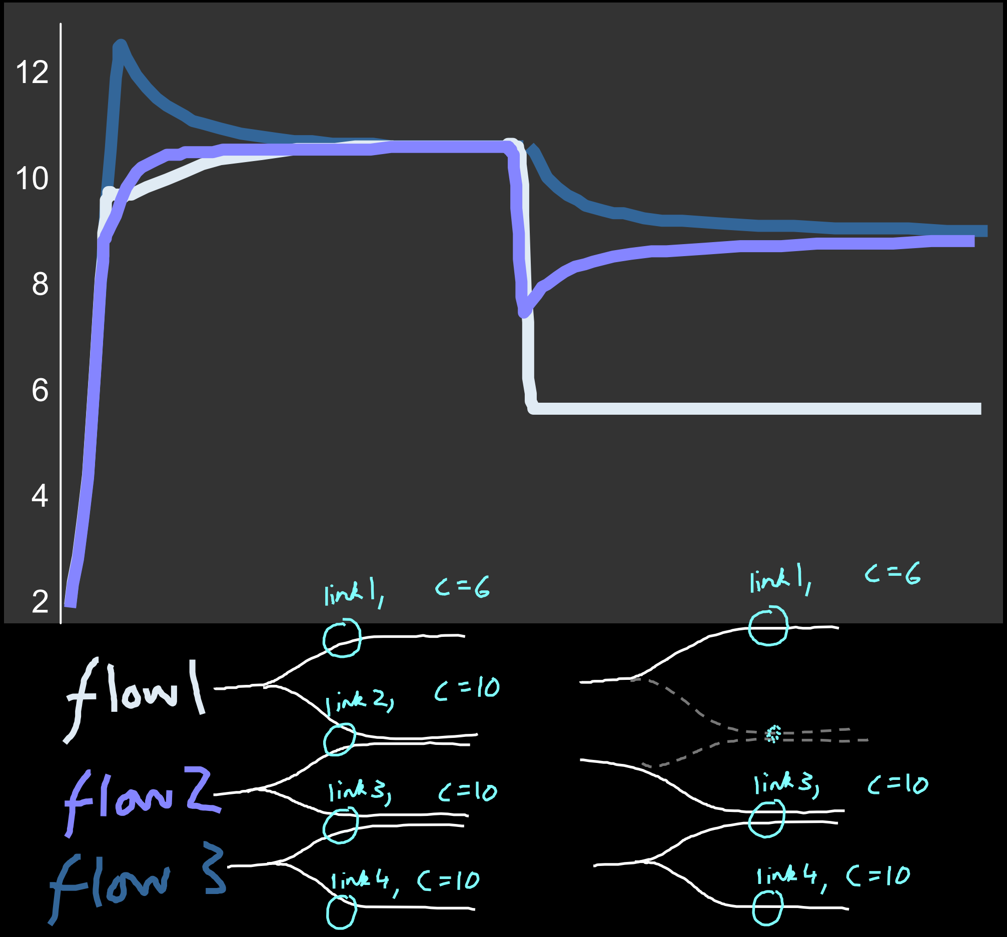
# Unfairness of per-path congestion control





- When some flows have several paths available, and other flows do not, it is fairer to allocate capacity to *users* rather than to *paths*
- Therefore we should not use standalone TCP congestion control on each path individually
- In maths: we want the congestion control to solve a user-centric utility maximization problem

# Stability and responsiveness



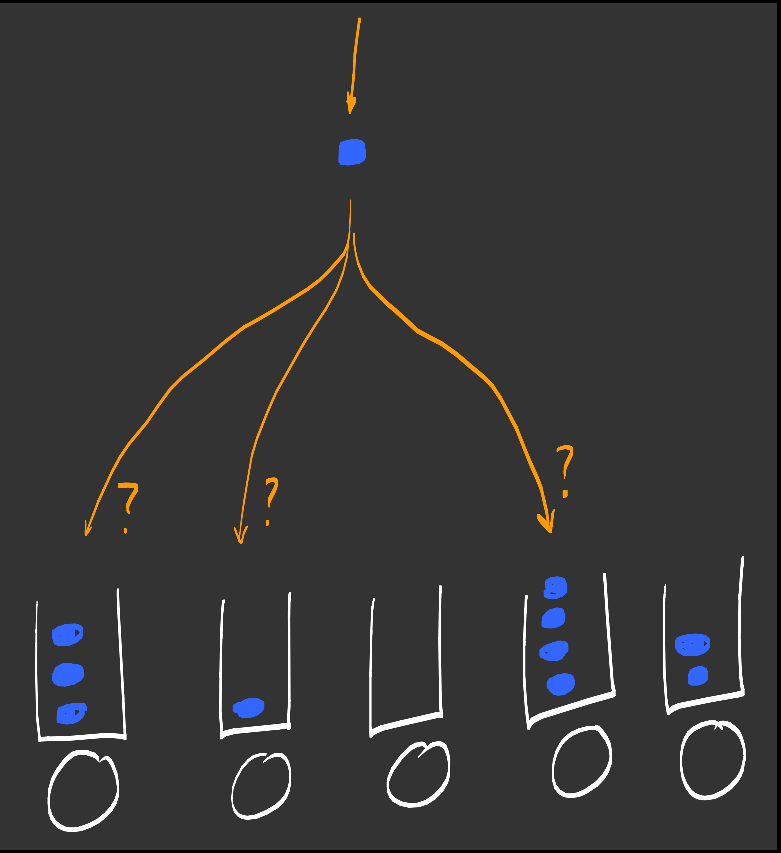
- When a link fails, or when flows come and go, rates should rebalance gradually
- Otherwise the network will be susceptible to route flap / oscillations
- In maths: the dynamical system describing the algorithm should be stable

# IV. Power of two choices

“Queueing system with selection of the shortest of two queues: an asymptotic approach”, Vvedenskaya, Dobrushin, Karpelevich, 1996

“The power of two choices in randomized load balancing”, Mitzenmacher, 1996

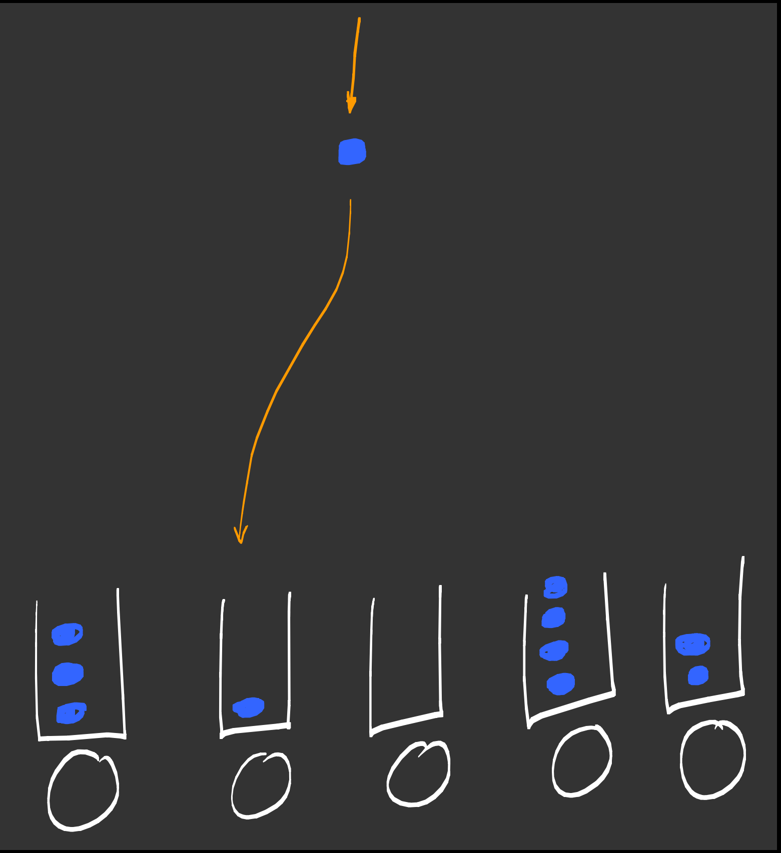
# A simple example of load-balancing



Packets arrive as a random process. When they arrive they look at $m$ queues, selected at random, and join the shorter queue

What is the resulting distribution of queue sizes?

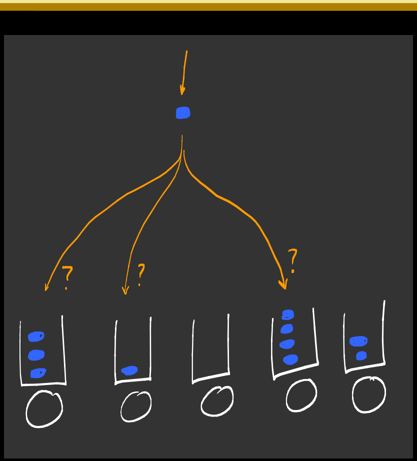
# A simple example of load-balancing



Packets arrive as a random process. When they arrive they look at $m$ queues, selected at random, and join the shorter queue

What is the resulting distribution of queue sizes?

# A simple example of load-balancing



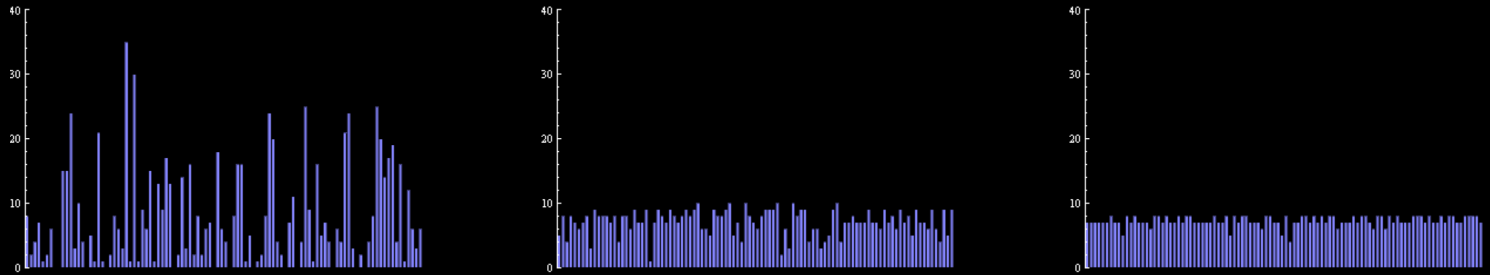
The larger $m$ is, the more uniform the queue sizes should be. Surprisingly, $m$=2 is enough to get most of the benefit.

If $m$=1 then Prob($Q$>$q$)=$\rho^q$

If $m\geq$2 then Prob($Q$>$q$)=$\rho^{(m^q\text{-}1)/(m\text{-}1)}$

(Here $\rho$ is the average load, arrivals are Poisson, service is exponential.)

A sample of queue sizes, for different values of $m$

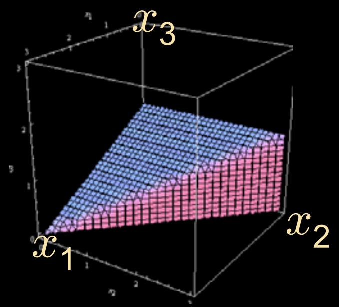


$m$=1

$m$=2

$m$=5

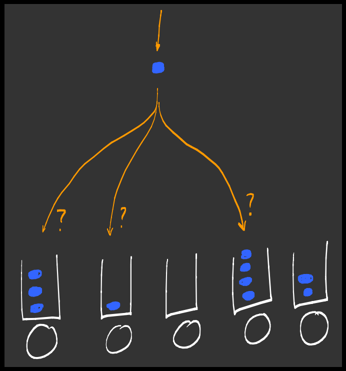
# Two paths good



$x_1+x_2 \leq C_1+C_2$

$x_1+2x_3 \leq C_3+C_4$

- Recall resource pooling:
  This is what happens when all the real resources that make up a virtual resource are kept fully occupied as long as possible. It means that the system is operating at peak efficiency.



- Perhaps, if each flow could balance itself across two real resources in every virtual resource, then we will achieve resource pooling.

# Questions

Q

- What traffic mix / routing will give us load balancing?
  - Maybe, if 50% of flows using a given virtual resource have two choices of which real resource to use, this will be enough
  - Is BitTorrent doing all this already?
  - How can we formulate this question properly?

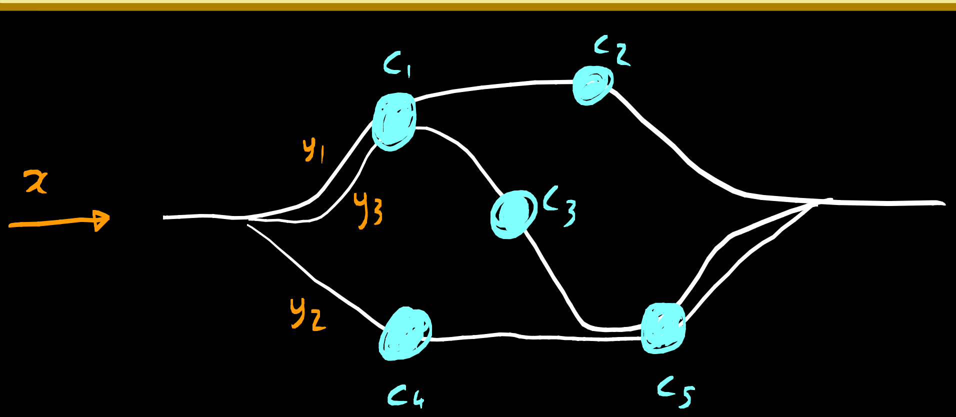
- What is the relationship between heavy traffic resource pooling and the example of ‘power of two choices’?
  - How does ‘power of two choices’ work in networks, i.e. not just an array of homogeneous resources?
  - One theory looks at the traffic matrix that leads to resource pooling, the other looks at the routing choice that leads to resource pooling. How can we integrate these two?

# V. Wardrop equilibrium

“Partially optimal routing”, Acemoglu, Johari, Ozdaglar, 2007

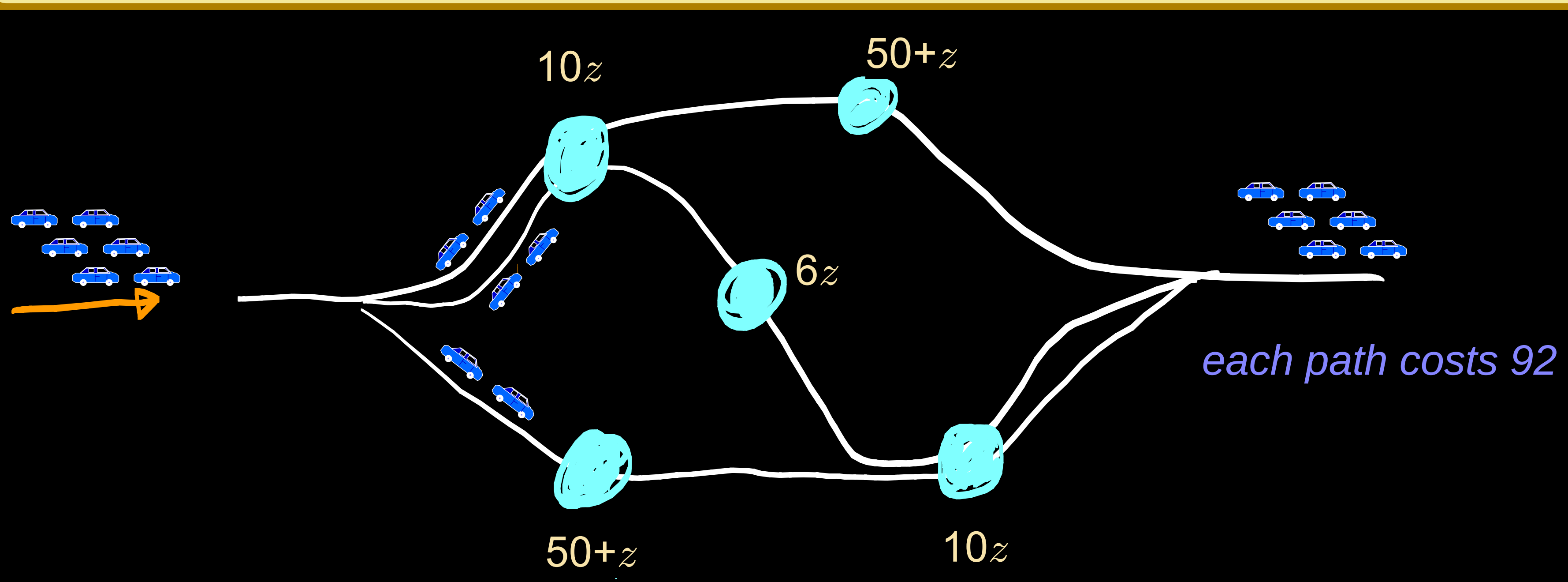
“How bad is selfish routing?”, Roughgarden, Tardos, 2002

# Wardrop equilibrium



- Suppose the total traffic $x$ is made up of many users who each can choose how to split their traffic, and they make their choice selfishly so as to minimize their congestion cost
- Let the congestion cost at resource 1 be $C_1(y_1+y_3)$ etc. Let the congestion cost of a path be the sum of the congestion costs along the route
- Then, if some path has greater congestion cost than others, no one will send any traffic over that path
- The resulting allocation is called a Wardrop equilibrium

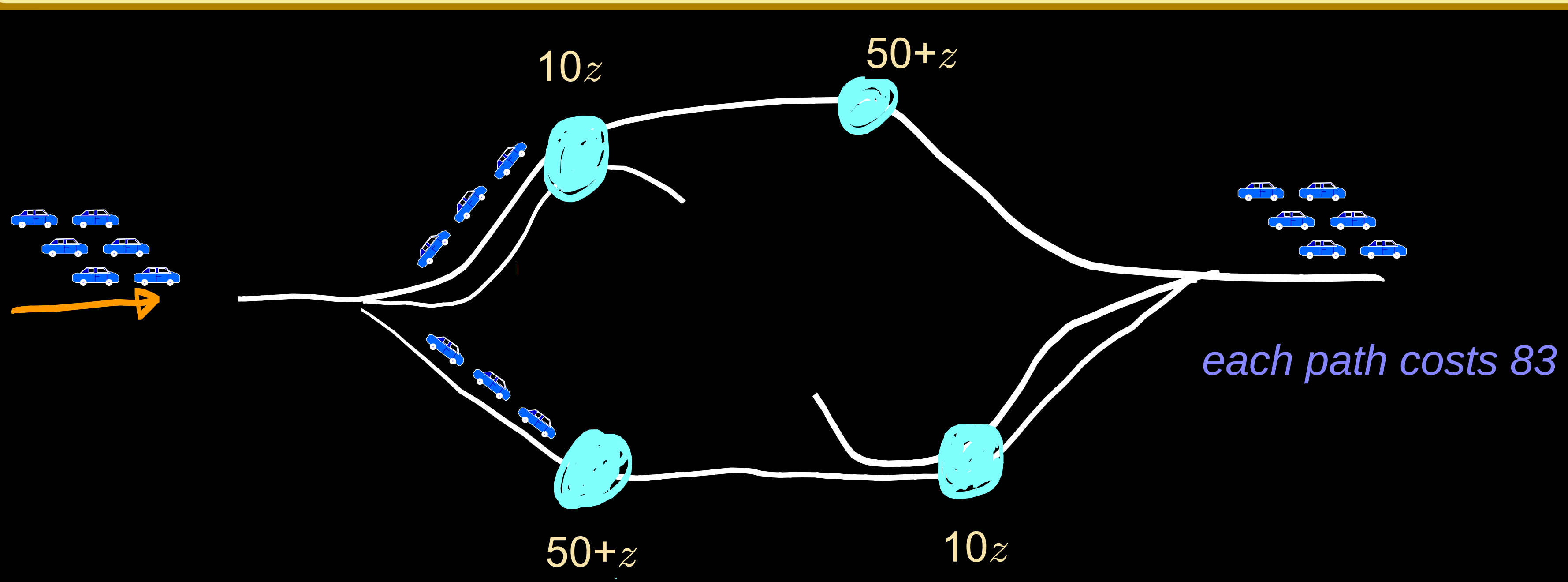
# Braess's paradox



- Braess discovered a situation where by removing resource 3 we end up with lower congestion costs for all users
- This is called Braess's paradox

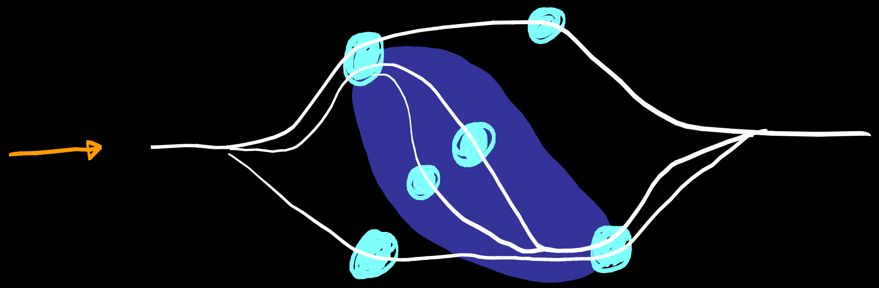
- It is not an uncommon problem!

# Braess's paradox



- Braess discovered a situation where by removing resource 3 we end up with lower congestion costs for all users
- This is called Braess's paradox
- It is not an uncommon problem!

# Generalized Braess's paradox



- Suppose an AS owns the middle of the network, and does load-balancing
- This will decrease the net congestion cost of traversing the middle of the network
- This may INCREASE the average congestion costs to users, by an extension of Braess's paradox
- The relative cost

  $C$(selfish users & load-balancing)/$C$(selfish users)

  can be arbitrarily large
- Plus other perverse outcomes...

# Questions

Q

- Why do these problems not show up in the earlier model of capacity constraints? Is it because the model of congestion costs is fishy?
- The difference seems to be to do with congestion signals: the signals to which users respond are in this case not in alignment with the costs.
- It may be that the network has much more routing flexibility than do individual users. Does this lead to a different answer? How can we fit it into the model?

# More questions

Q

- Are there appropriate ways of signalling congestion, such that user and network incentives align? For example, if BitTorrent responded appropriately, could it do a much better job of traffic engineering?
- Multipath should lead to more competition, which should force prices closer to costs. What are appropriate models of this? Is P2P already enough to encourage competition?