Multipath TCP design, and application to data centers

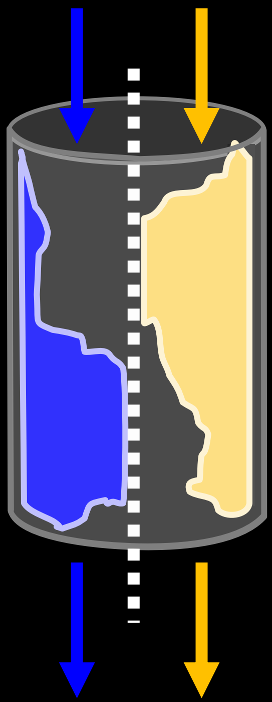
Damon Wischik, Mark Handley, Costin Raiciu, Christopher Pluntke

Packet switching 'pools' circuits.

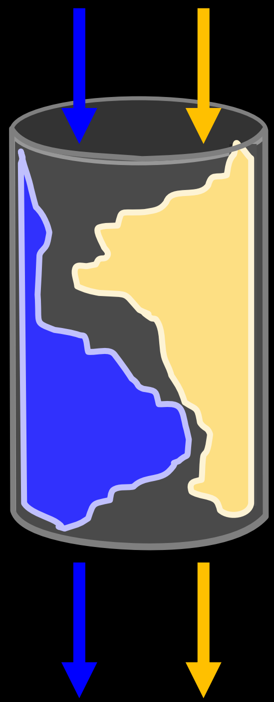
Multipath 'pools' links : it is Packet Switching 2.0.

TCP controls how a link is shared.

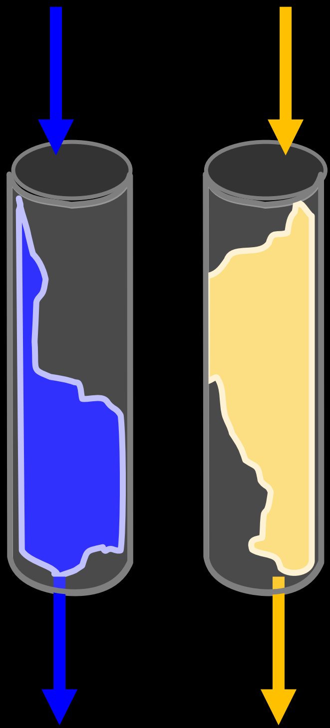
How should a pool be shared? What is TCP 2.0?



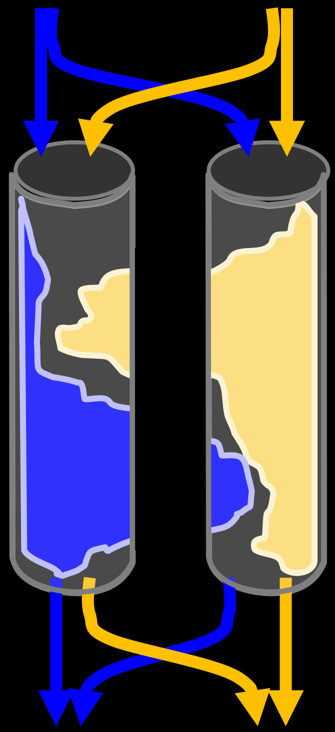
Two circuits



A link

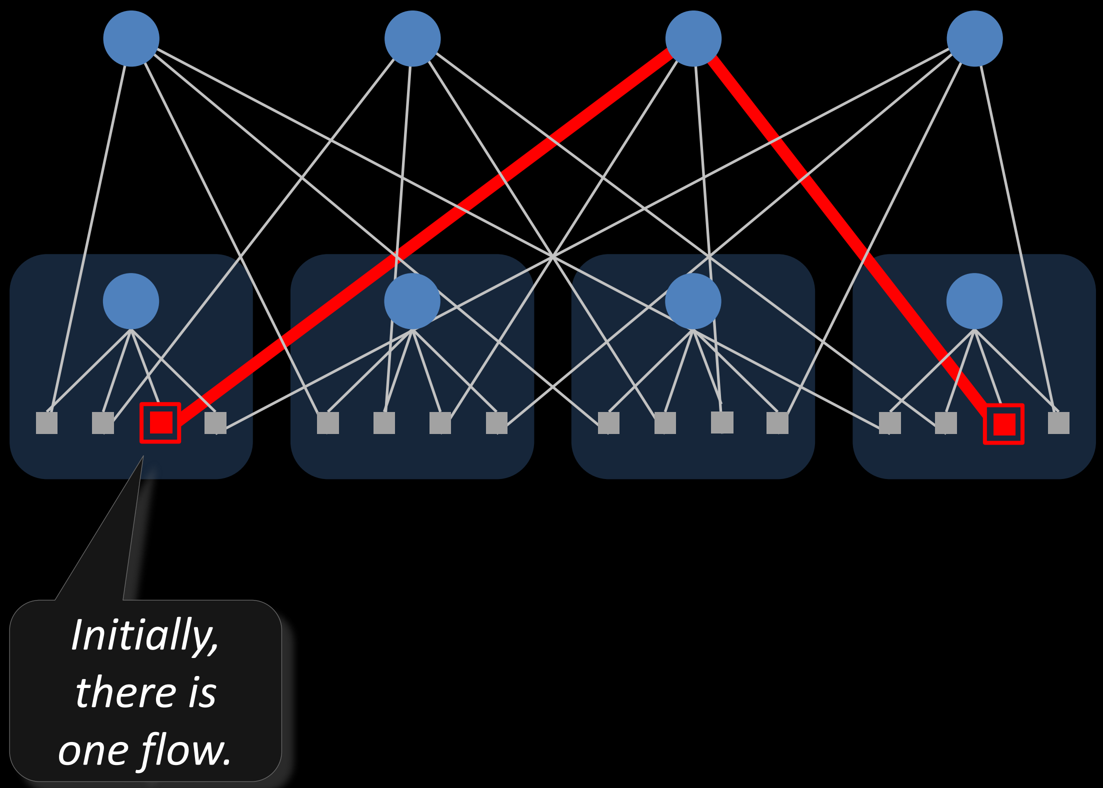


Two separate links

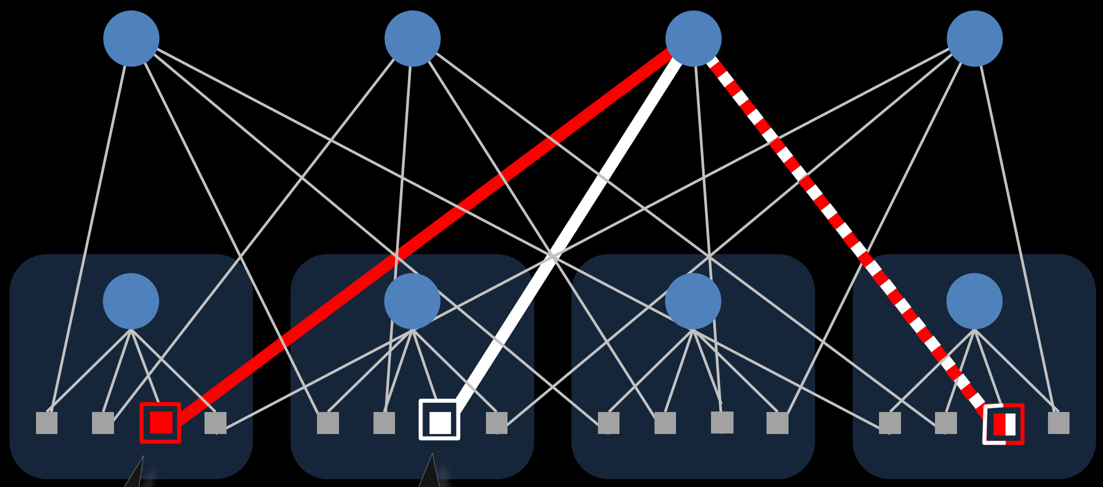


A pool of links

In a data center, can we use multipath to get higher throughput?



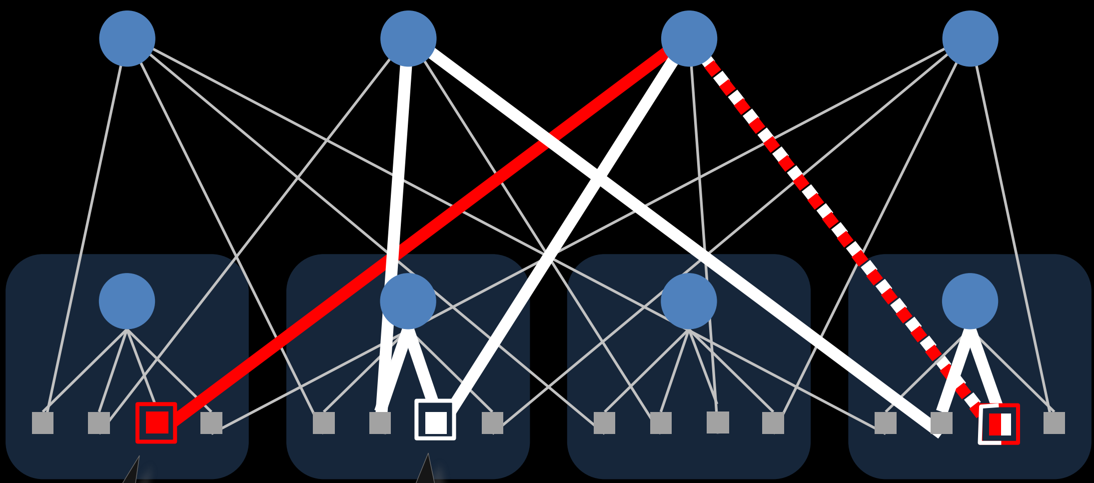
In a data center, can we use multipath to get higher throughput?



Initially, there is one flow.

A new flow starts. Its direct route collides with the first flow.

In a data center, can we use multipath to get higher throughput?

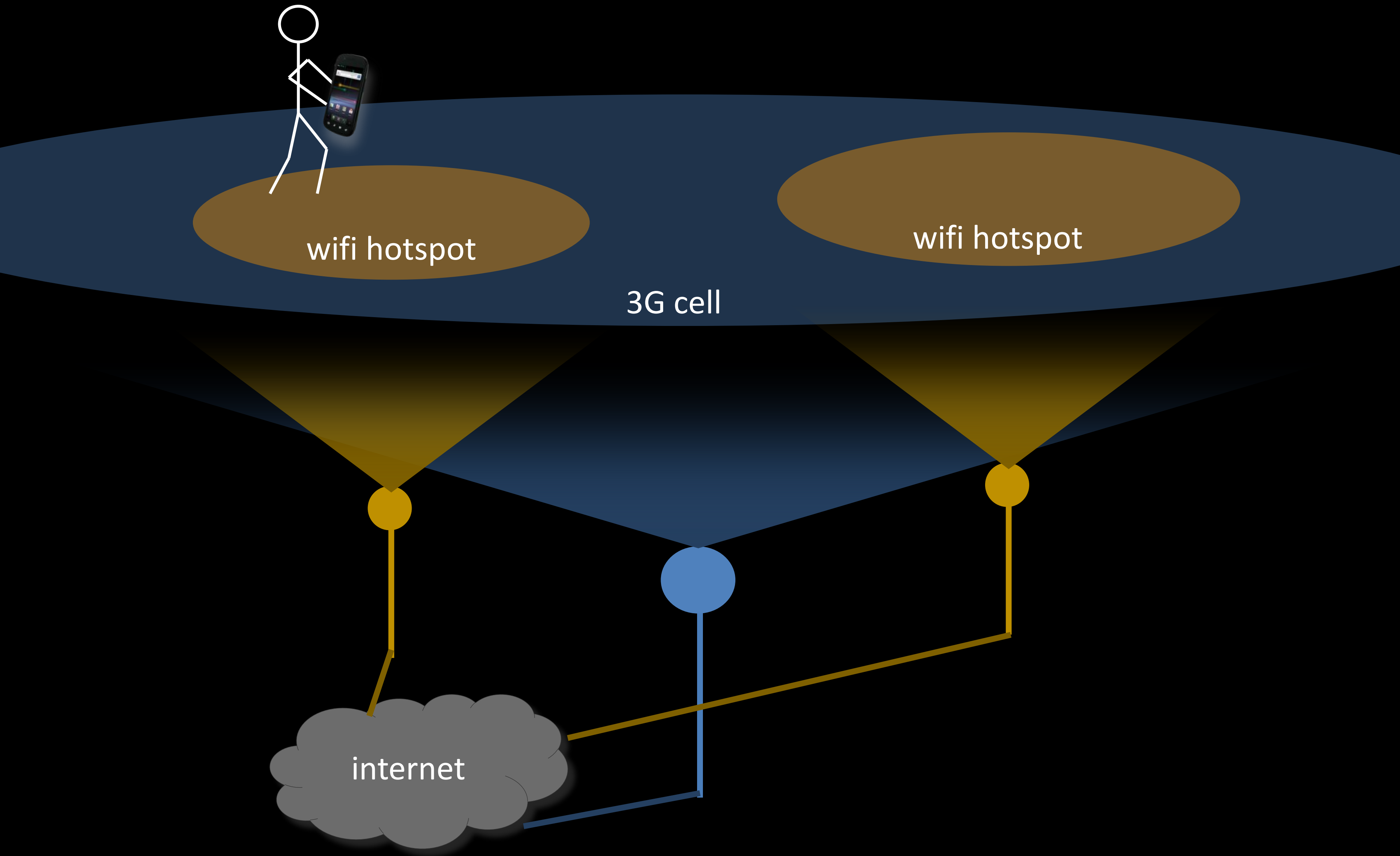


Initially, there is one flow.

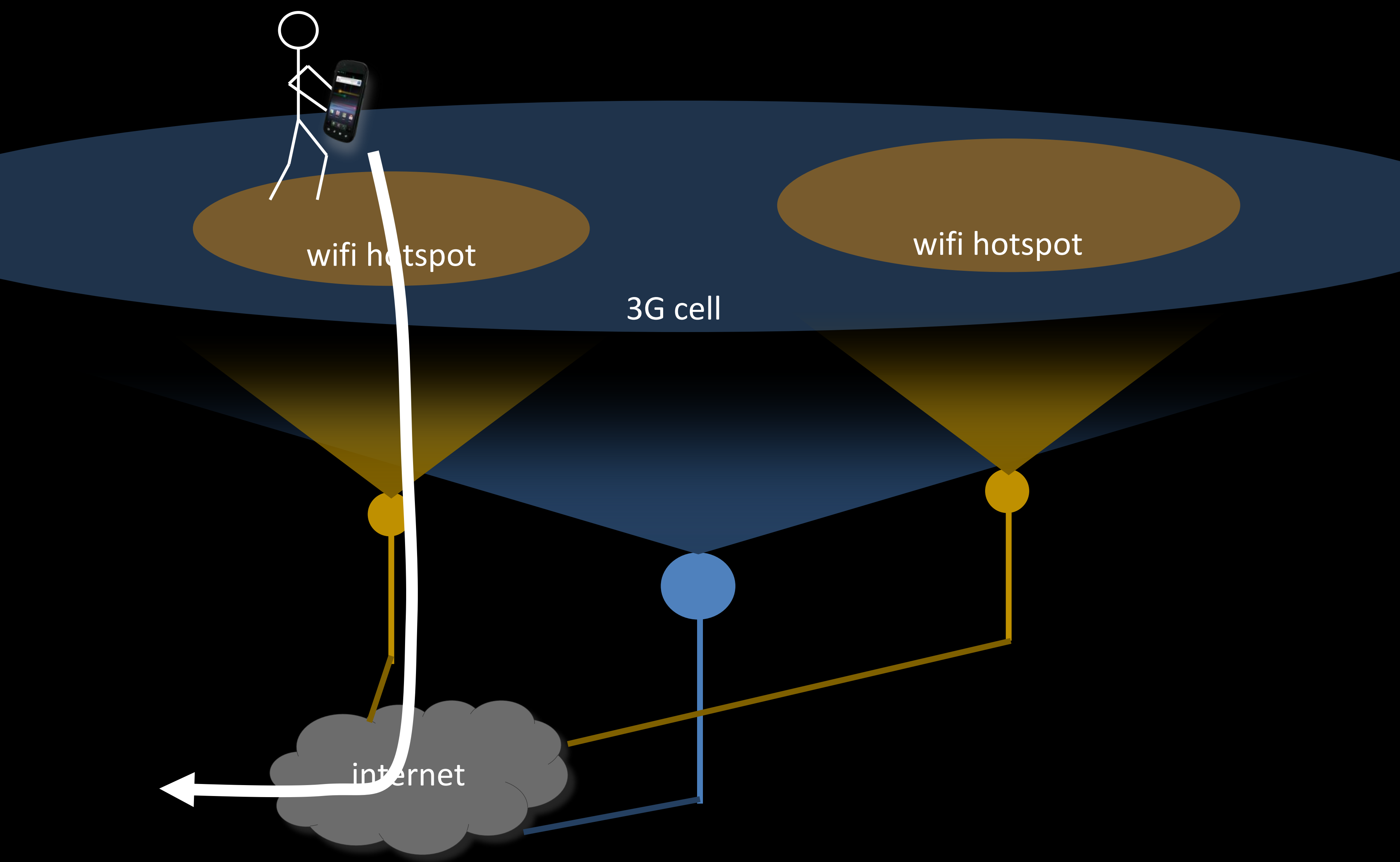
A new flow starts. Its direct route collides with the first flow.

But it also has longer routes available, which don't collide.

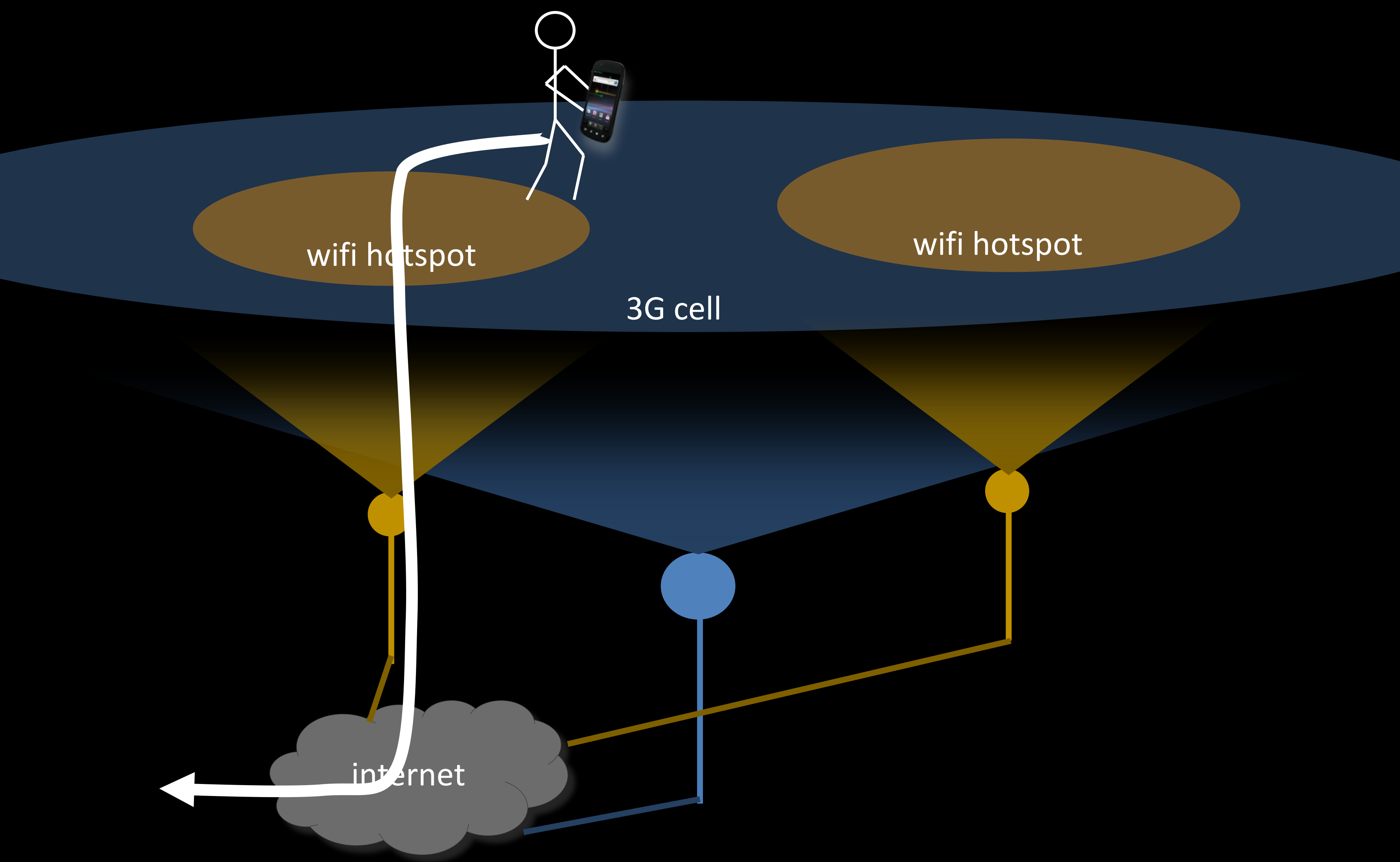
Can multipath help with mobile hand-offs?



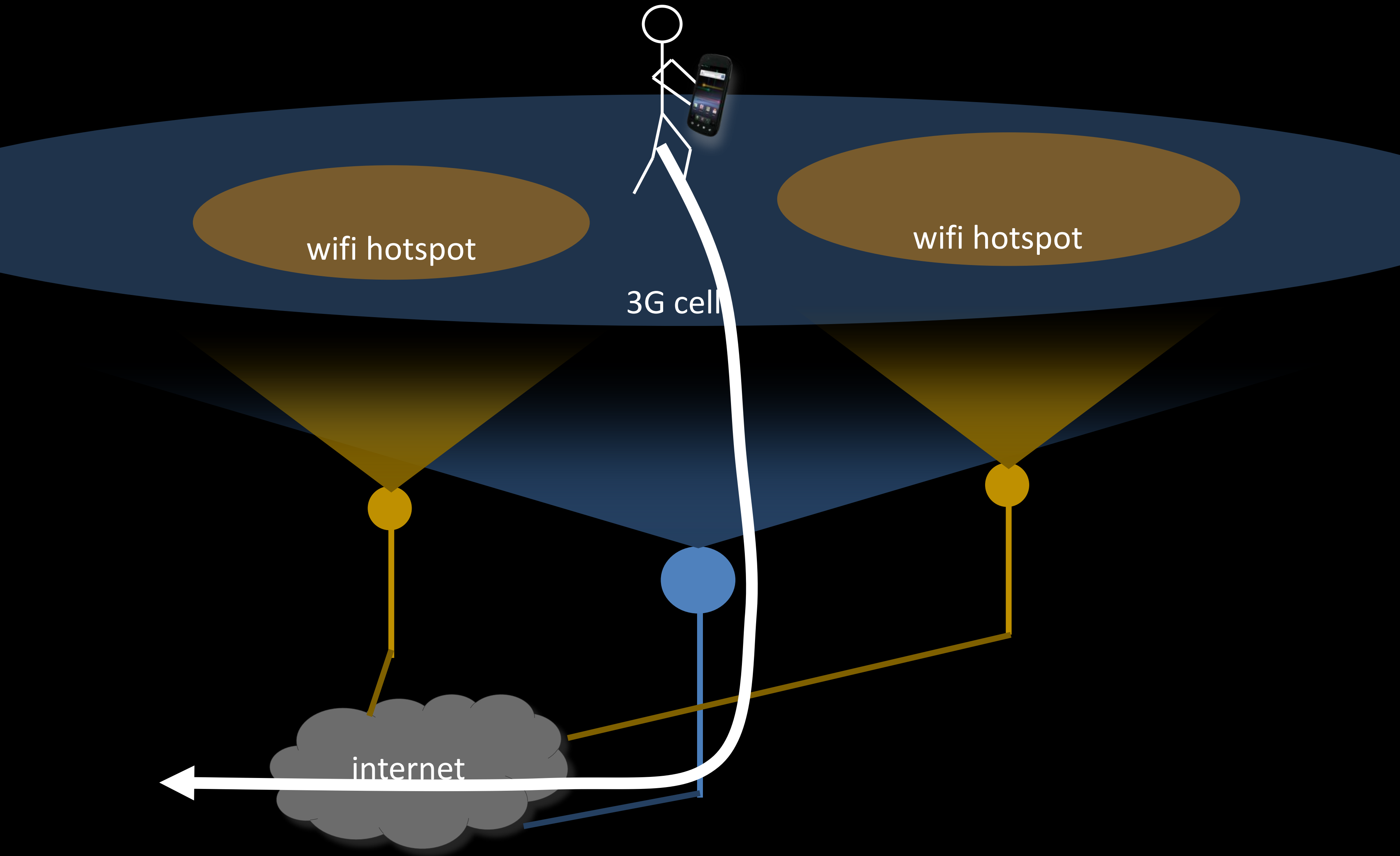
Can multipath help with mobile hand-offs?



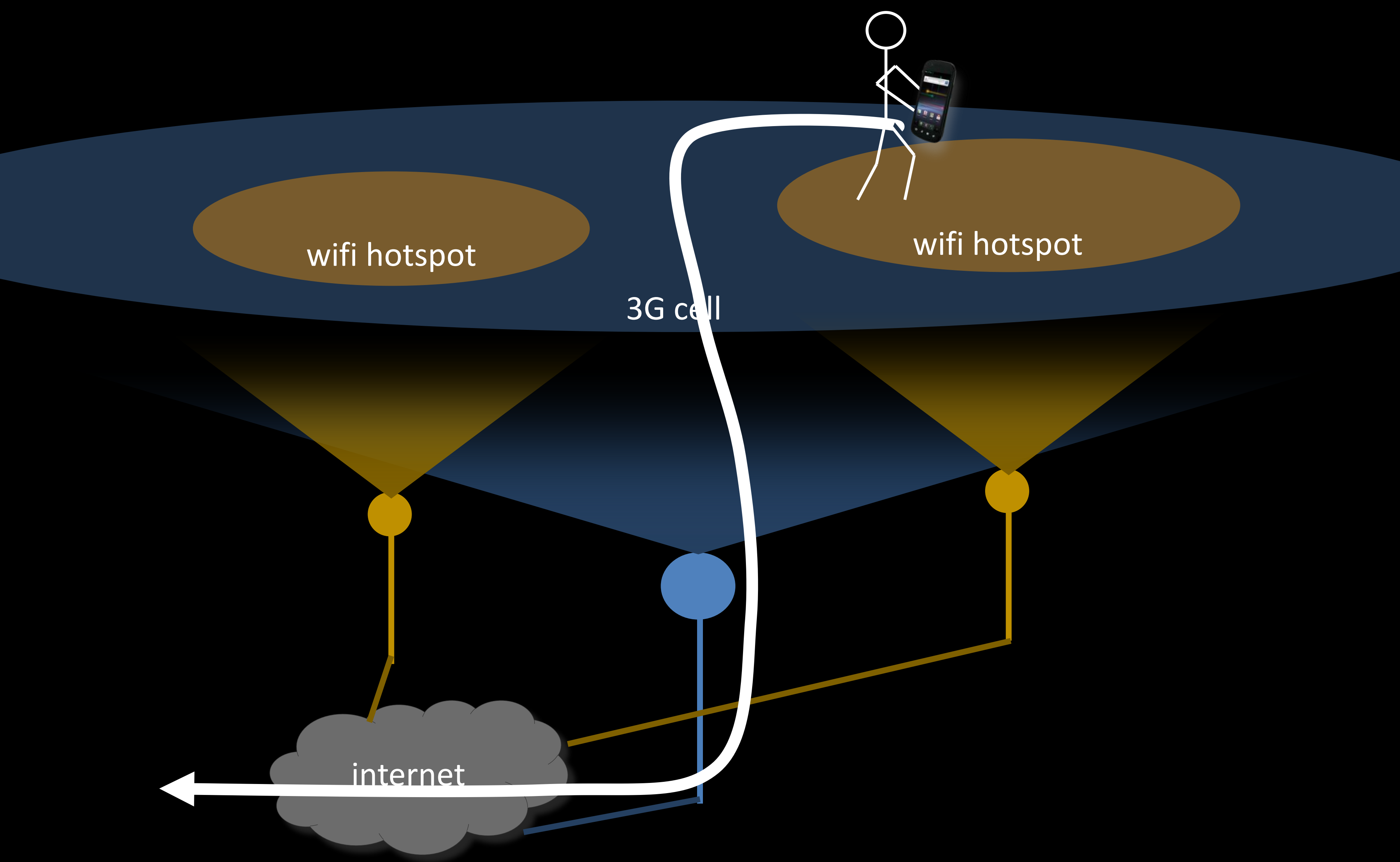
Can multipath help with mobile hand-offs?



Can multipath help with mobile hand-offs?

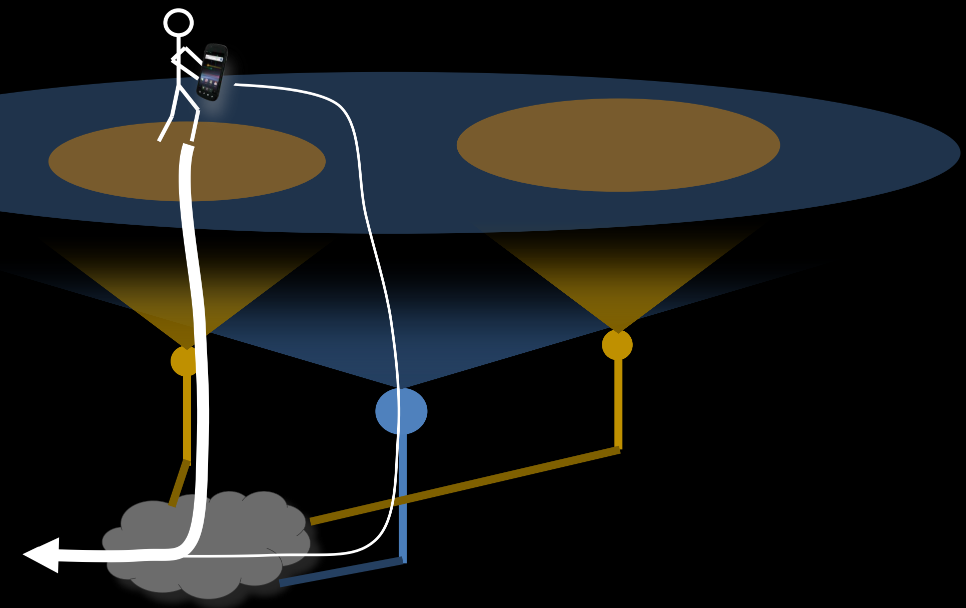


Can multipath help with mobile hand-offs?



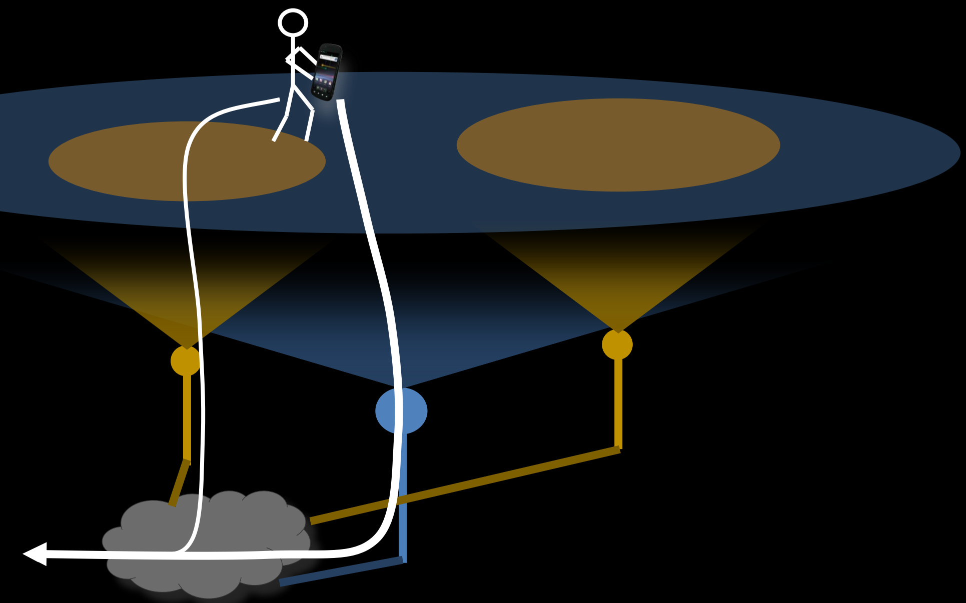
Can multipath help with mobile hand-offs?

If your phone uses both radios simultaneously, you needn’t experience any interruption.



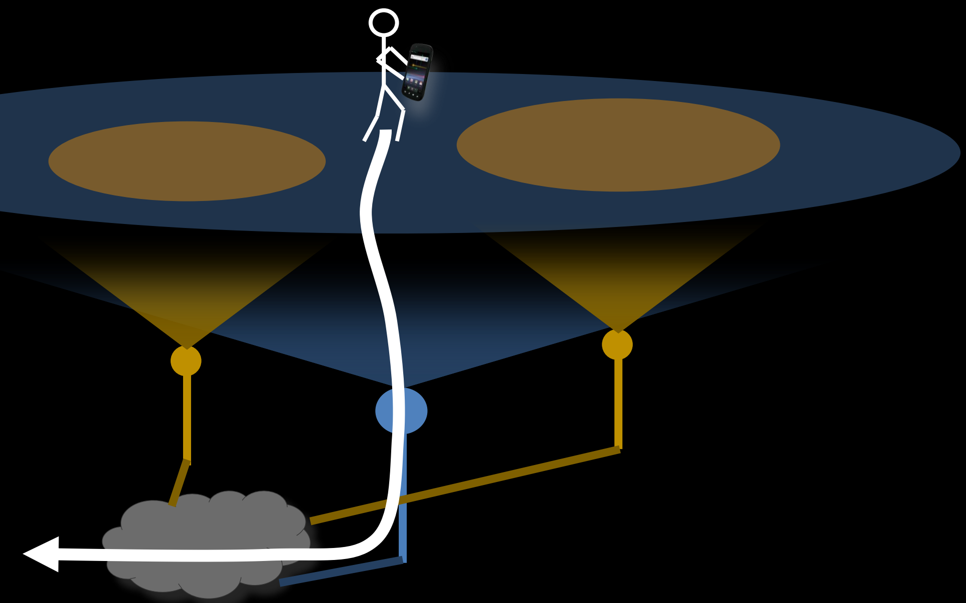
Can multipath help with mobile hand-offs?

If your phone uses both radios simultaneously, you needn’t experience any interruption.



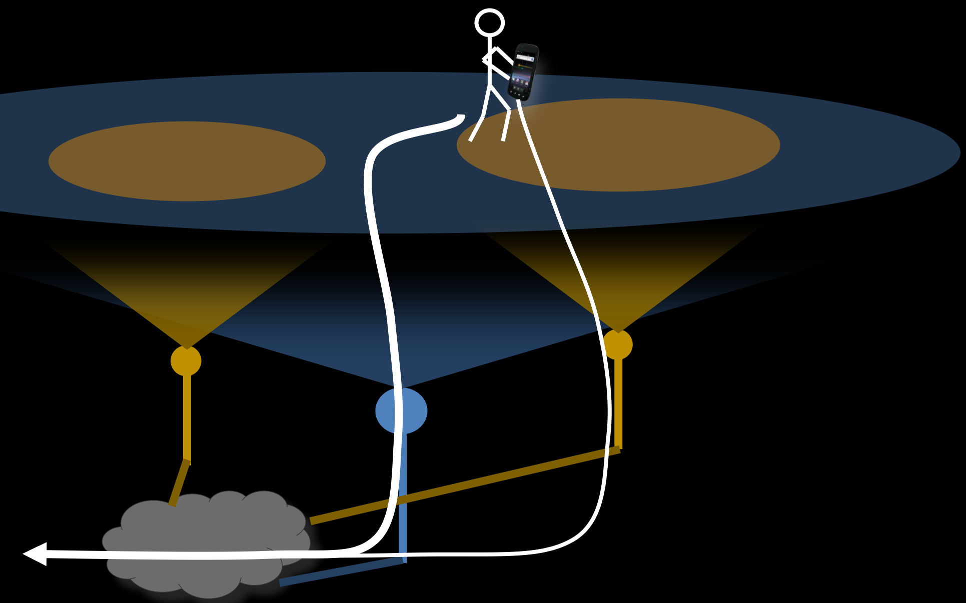
Can multipath help with mobile hand-offs?

If your phone uses both radios simultaneously, you needn’t experience any interruption.



Can multipath help with mobile hand-offs?

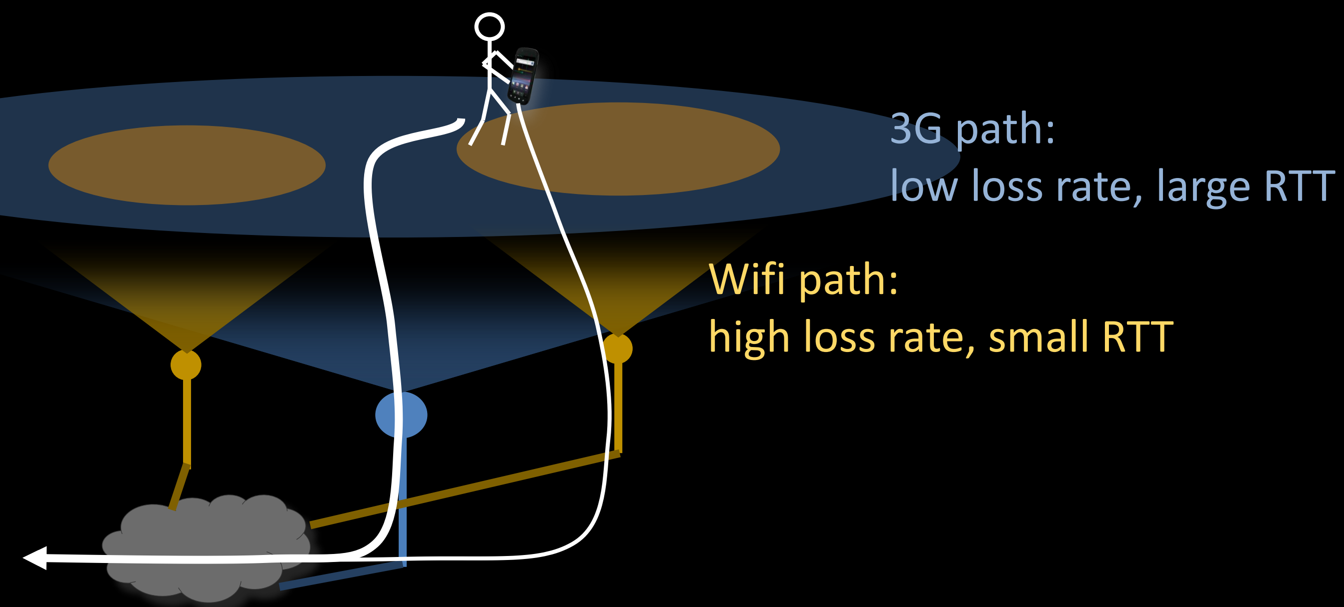
If your phone uses both radios simultaneously, you needn't experience any interruption.



Can multipath help with mobile hand-offs?

If your phone uses both radios simultaneously, you needn't experience any interruption.

How should it balance traffic across dissimilar paths?



We designed the MPTCP protocol to be a general-purpose multipath replacement for TCP.

- I will describe our design process behind MPTCP's congestion control algorithm.

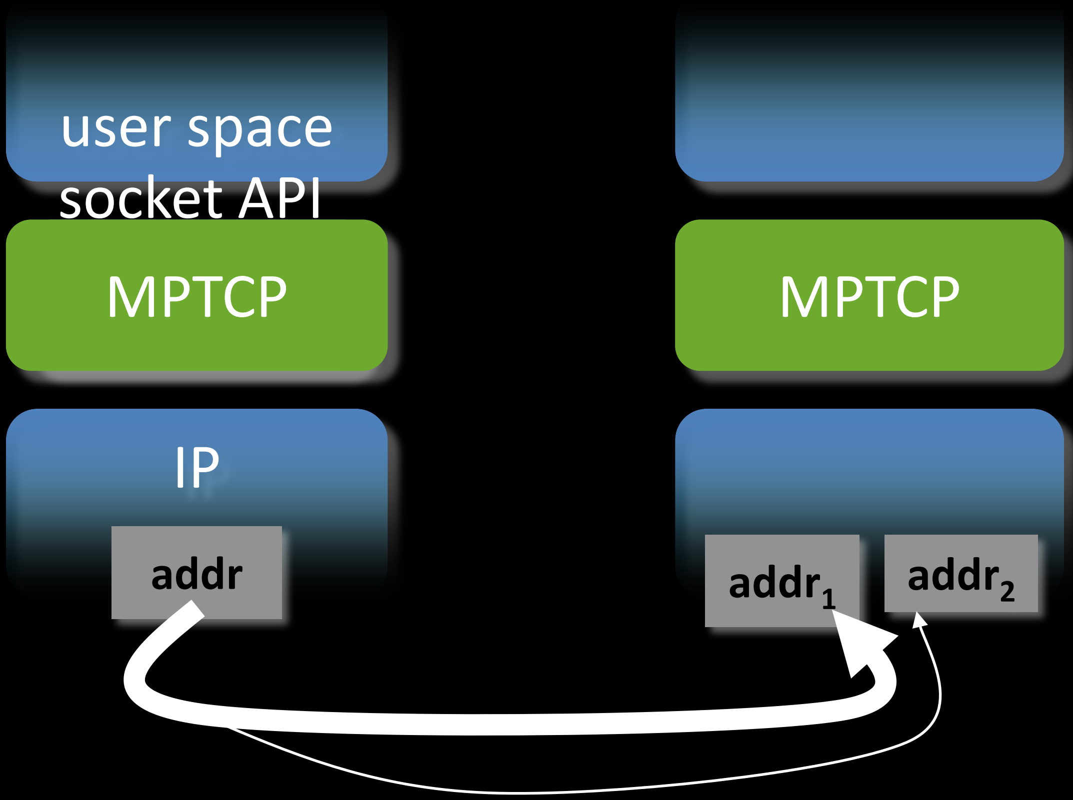
MPTCP should be beneficial in data centers.

- I will describe experimental & simulation results.

What is the MPTCP protocol?

MPTCP is a replacement for TCP which lets you use multiple paths simultaneously.

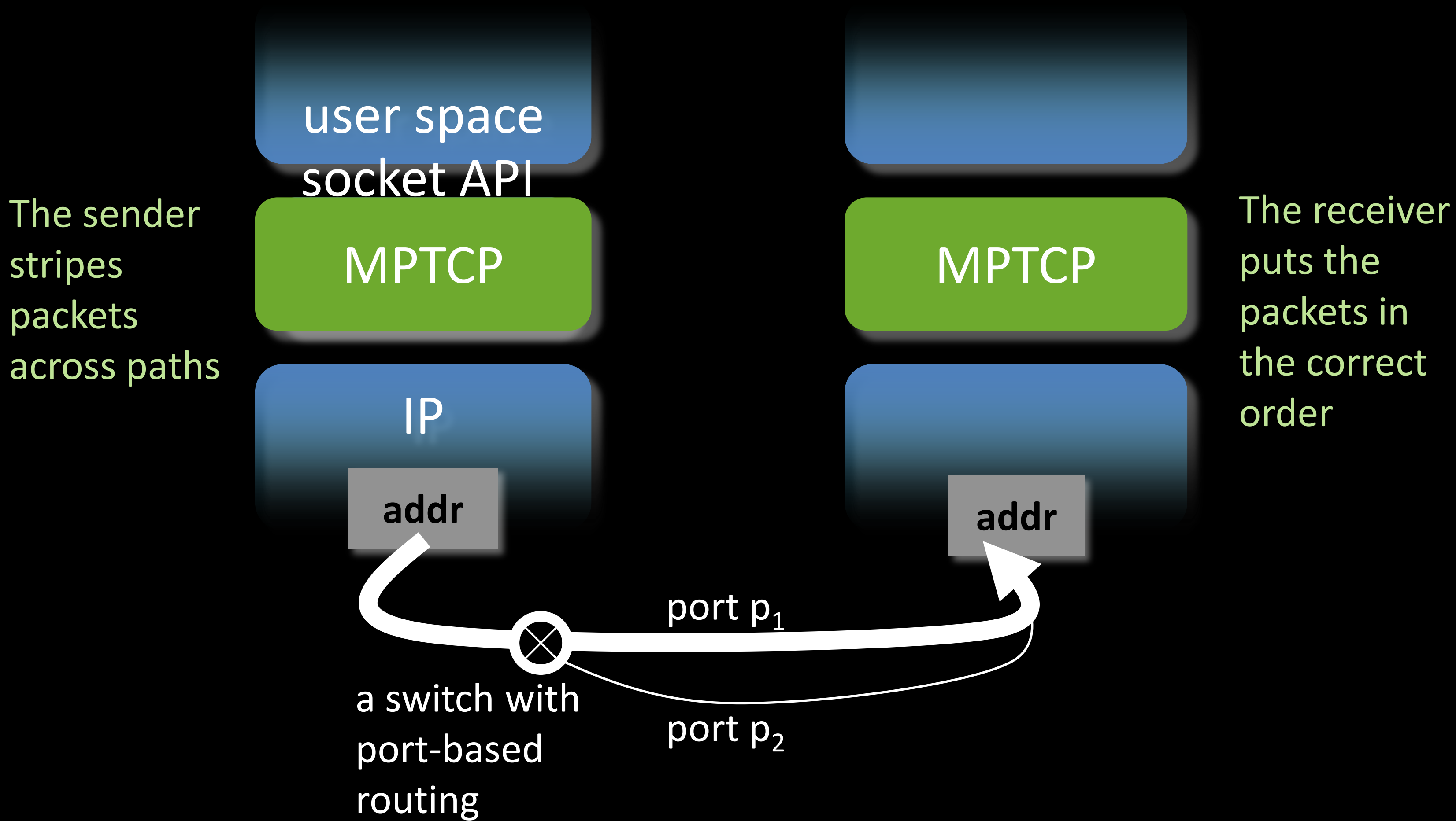
The sender stripes packets across paths



The receiver puts the packets in the correct order

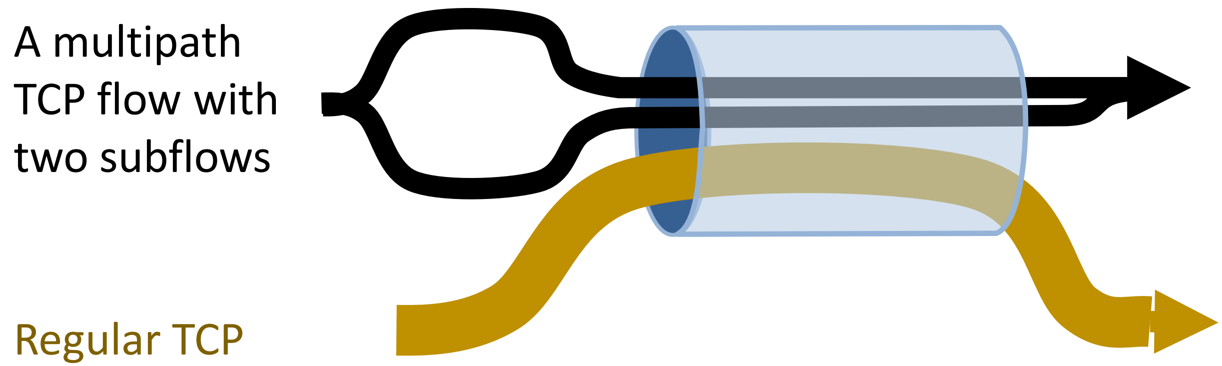
What is the MPTCP protocol?

MPTCP is a replacement for TCP which lets you use multiple paths simultaneously.



Design goal 1:

Multipath TCP should be fair to regular TCP at shared bottlenecks



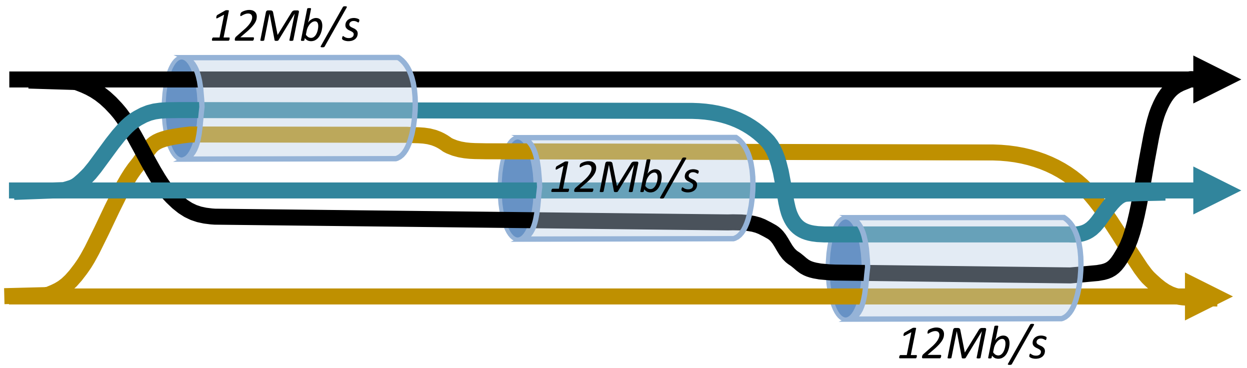
To be fair, Multipath TCP should take as much capacity as TCP at a bottleneck link, no matter how many paths it is using.

Strawman solution:

Run “½ TCP” on each path

Design goal 2:

MPTCP should use efficient paths

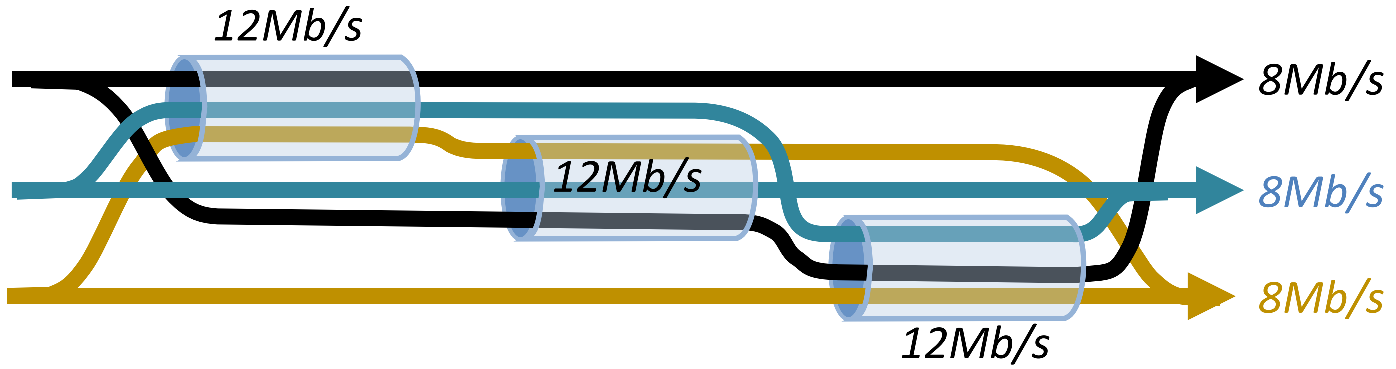


Each flow has a choice of a 1-hop and a 2-hop path.

How should split its traffic?

Design goal 2:

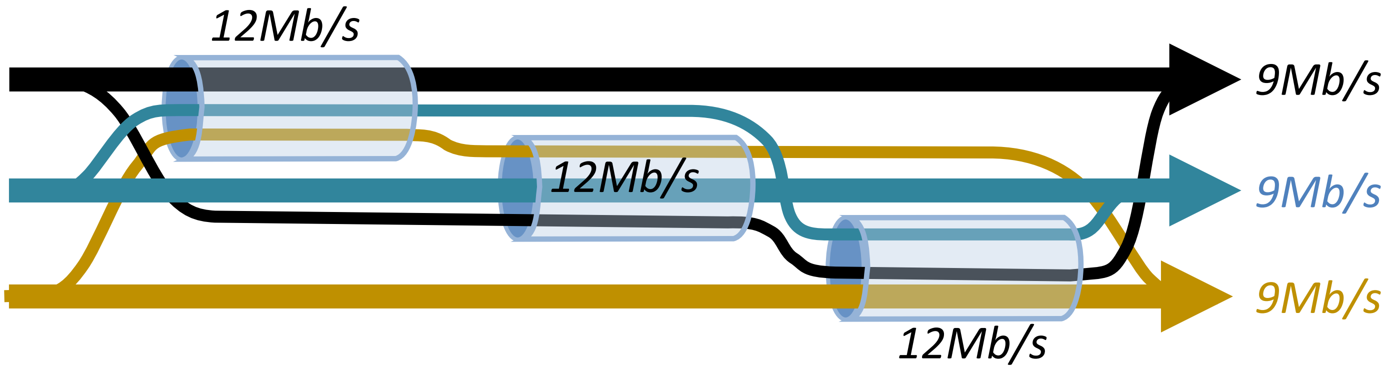
MPTCP should use efficient paths



If each flow split its traffic 1:1 ...

Design goal 2:

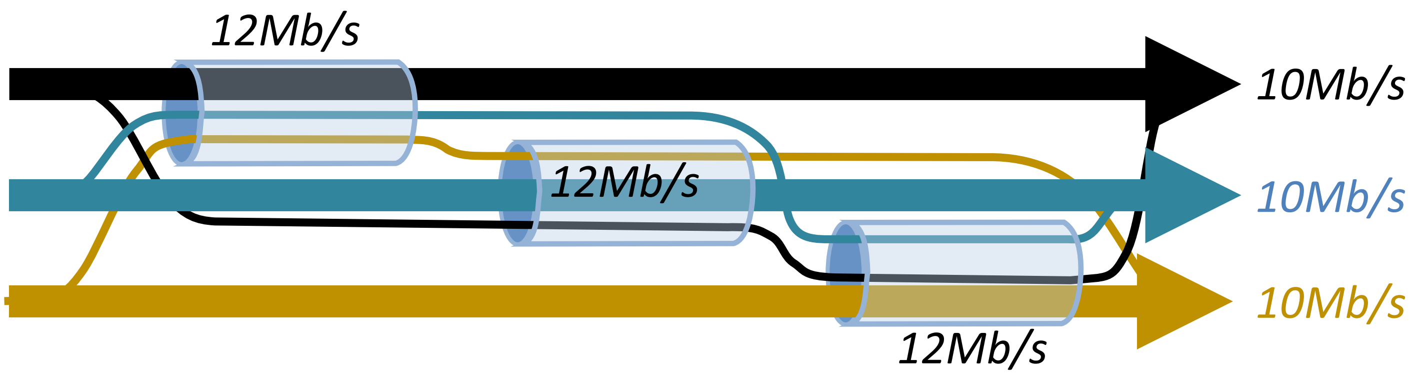
MPTCP should use efficient paths



If each flow split its traffic 2:1 ...

Design goal 2:

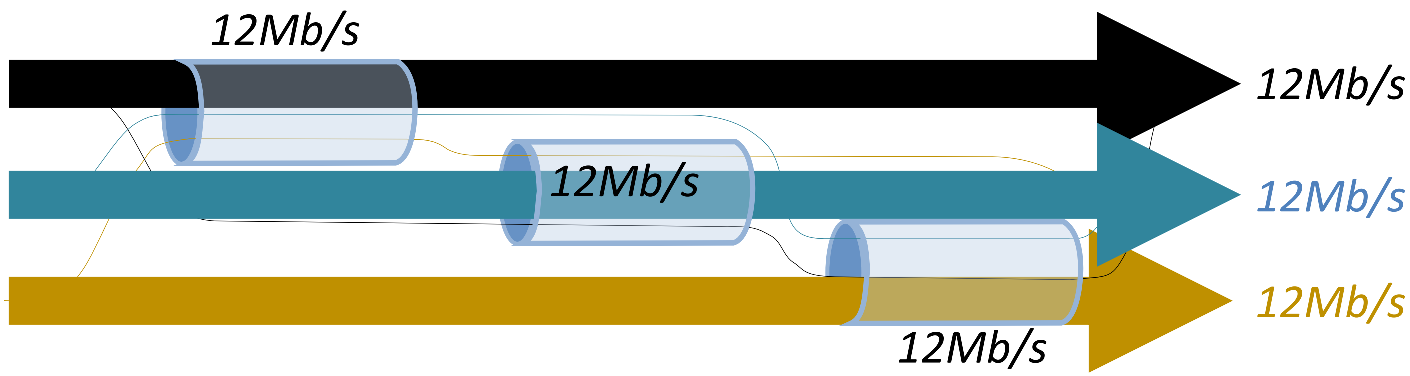
MPTCP should use efficient paths



If each flow split its traffic 4:1 ...

Design goal 2:

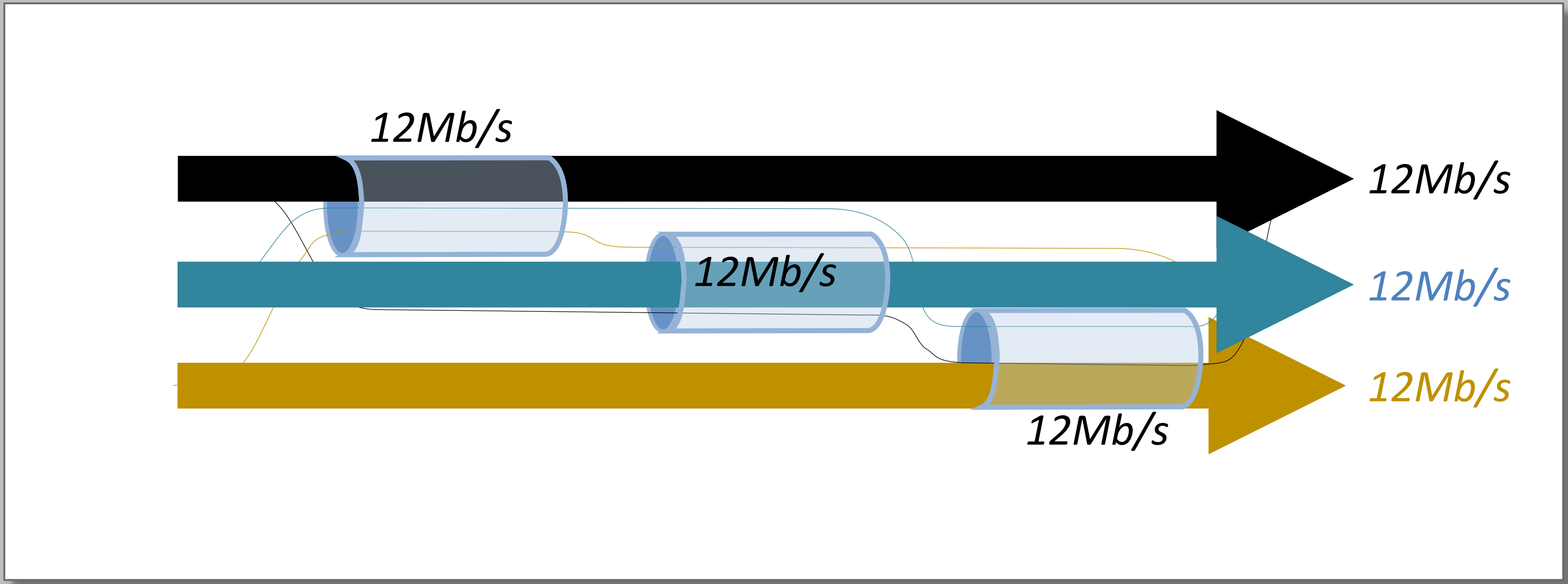
MPTCP should use efficient paths



If each flow split its traffic ∞:1 ...

Design goal 2:

MPTCP should use efficient paths

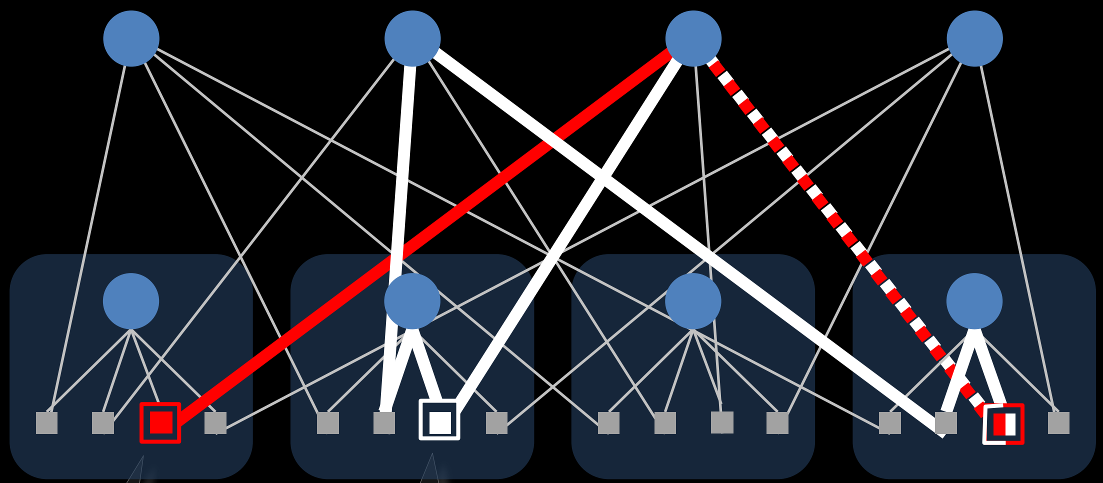


Theoretical solution (Kelly+Voice 2005; Han, Towsley et al. 2006)

MPTCP should send all its traffic on its least-congested paths.

Theorem. This will lead to the most efficient allocation possible, given a network topology and a set of available paths.

MPTCP chooses efficient paths in a BCube data center, hence it gets high throughput.



Initially, there is one flow.

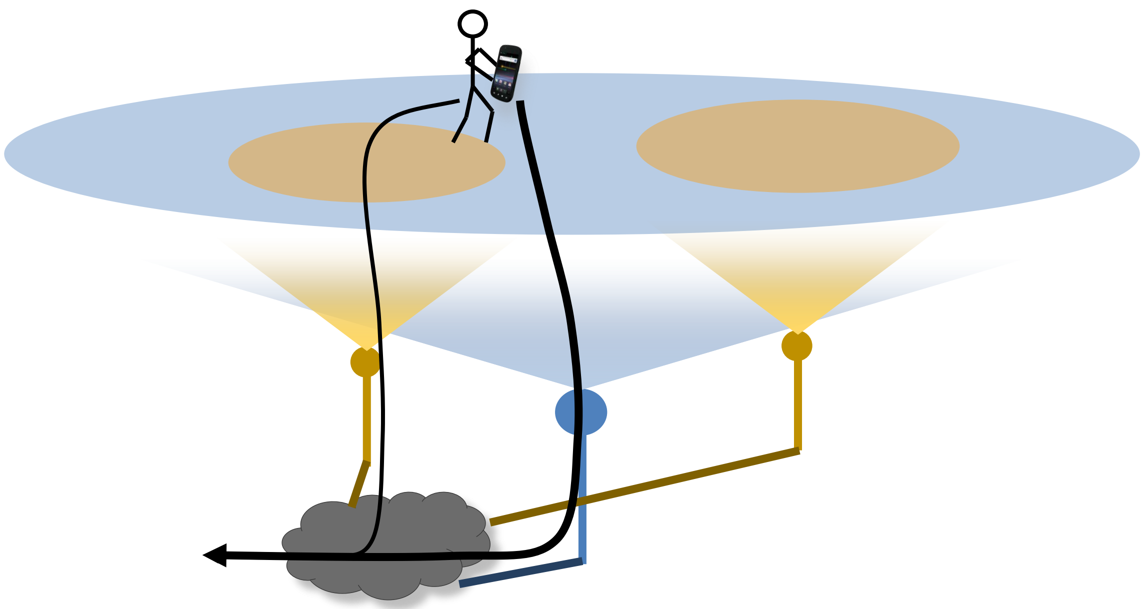
A new flow starts. Its direct route collides with the first flow.

But it also has longer routes available, which don't collide.

MPTCP shifts its traffic away from the congested link.

Design goal 3:

MPTCP should be fair compared to TCP



Design goal 2 says to put all your traffic on the least congested path.

Here, 3G is always the least congested path. But if we only used 3G, we'd get worse throughput.

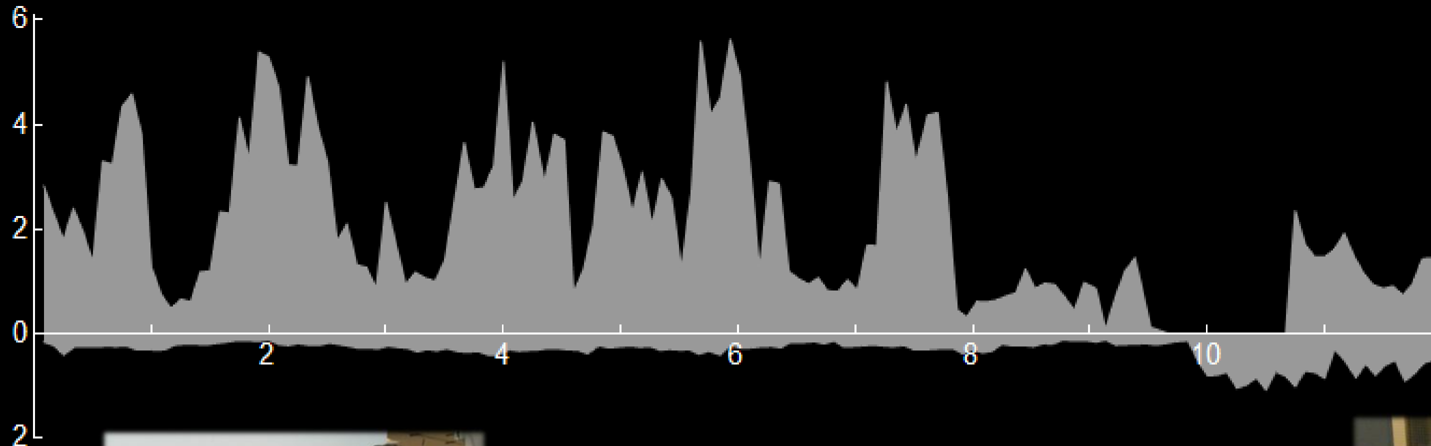
Goal 3a. A Multipath TCP user should get at least as much throughput as a single-path TCP would on the best of the available paths.

Goal 3b. A Multipath TCP flow should take no more capacity on any link than a single-path TCP would.

MPTCP gives fair throughput.

wifi through-put [Mb/s]

3G through-put [Mb/s]



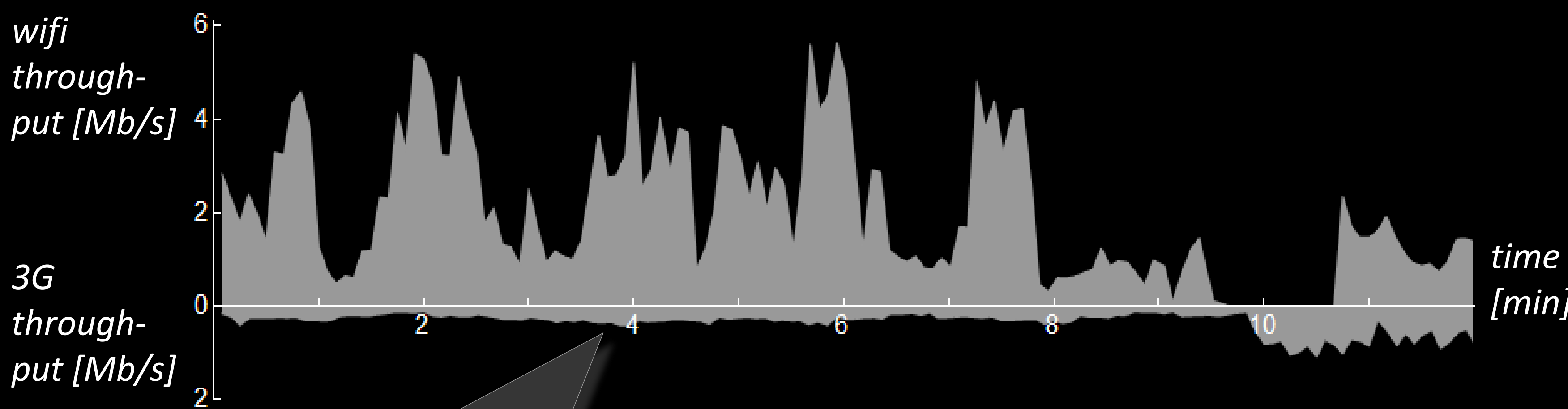
time [min]

User in his office, using wifi and 3G

Going downstairs

In the kitchen

MPTCP gives fair throughput.



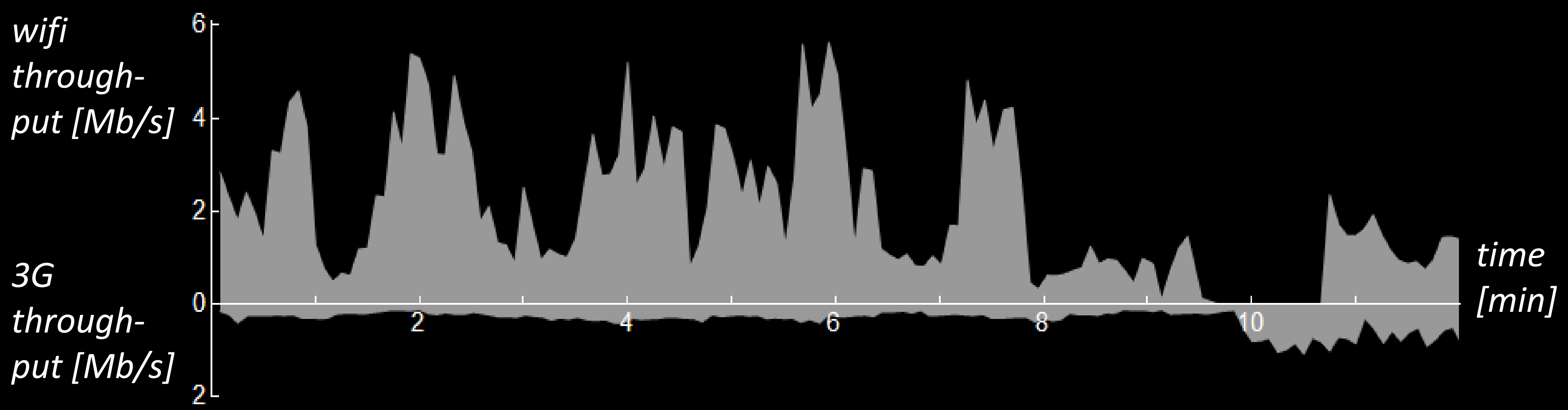
3G has lower loss rate. *Design Goal 2* says to shift traffic onto 3G ...

But, today, TCP over 3G was only getting 0.4Mb/s, so don't take more than that ...

But, today, TCP over wifi was getting 2.2Mb/s, so user is entitled to this much...

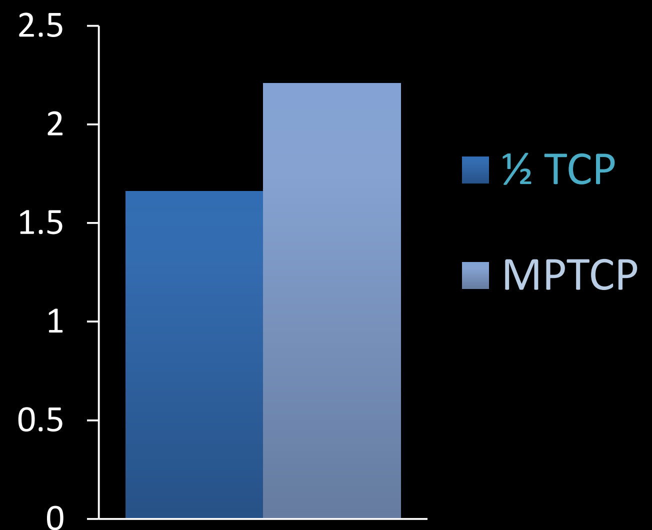
MPTCP sends 0.4Mb/s over 3G, and the remaining 1.8Mb/s over wifi.

MPTCP gives fair throughput.



We measured throughput , for both ½ TCP (strawman) and MPTCP, in the office.

½ TCP is unfair to the user, and its throughput is 25% worse than MPTCP.



Design goals

~~Goal 1. Be fair to TCP at bottleneck links~~ redundant

Goal 2. Use efficient paths ...

Goal 3. as much as we can, while being fair to TCP

Goal 4. Adapt quickly when congestion changes

Goal 5. Don’t oscillate

How does MPTCP achieve all this?

How does TCP congestion control work?

Maintain a congestion window w.

- Increase w for each ACK, by $1/w$
- Decrease w for each drop, by $w/2$

How does MPTCP congestion control work?

Maintain a congestion window w_r, one window for each path, where $r \in R$ ranges over the set of available paths.

- Increase w_r for each ACK on path r, by

$$\min_{S \subseteq R\,:\, r \in S} \frac{\max_{s \in S} w_s / \mathsf{RTT}_s^2}{\left(\sum_{s \in S} w_s / \mathsf{RTT}_s\right)^2}$$

- Decrease w_r for each drop on path r, by $w_r/2$

How does MPTCP congestion control work?

Maintain a congestion window w_r, one window for each path, where $r \in R$ ranges over the set of available paths.

Design goal 3:
At any potential bottleneck S that path r might be in, look at the best that a single-path TCP could get, and compare to what I'm getting.

- Increase w_r for each ACK on path r, by

$$\min_{S \subseteq R : r \in S} \frac{\max_{s \in S} w_s / \mathrm{RTT}_s^2}{\left(\sum_{s \in S} w_s / \mathrm{RTT}_s\right)^2}$$

- Decrease w_r for each drop on path r, by $w_r/2$

How does MPTCP congestion control work?

Maintain a congestion window w_r, **one window for each path**, where $r \in R$ ranges over the set of available paths.

Design goal 2:
We want to shift traffic away from congestion.

To achieve this, we increase windows in proportion to their size.

- Increase w_r for **each ACK on path r**, by

$$\min_{S \subseteq R : r \in S} \frac{\max_{s \in S} w_s / \mathrm{RTT}_s^2}{\left(\sum_{s \in S} w_s / \mathrm{RTT}_s\right)^2}$$

- Decrease w_r for **each drop on path r**, by $w_r/2$

Related work

on multipath congestion control

pTCP , CMT over SCTP, and M/TCP

that meets goal 1 (fairness at shared bottleneck)

mTCP, ≈ R-MTP

and goal 2 (choosing efficient paths)

Honda et al. (2009), ≈ Tsao and Sivakumar (2009)

and goal 5 (non-oscillation)

Kelly and Voice (2005), Han et al. (2006)

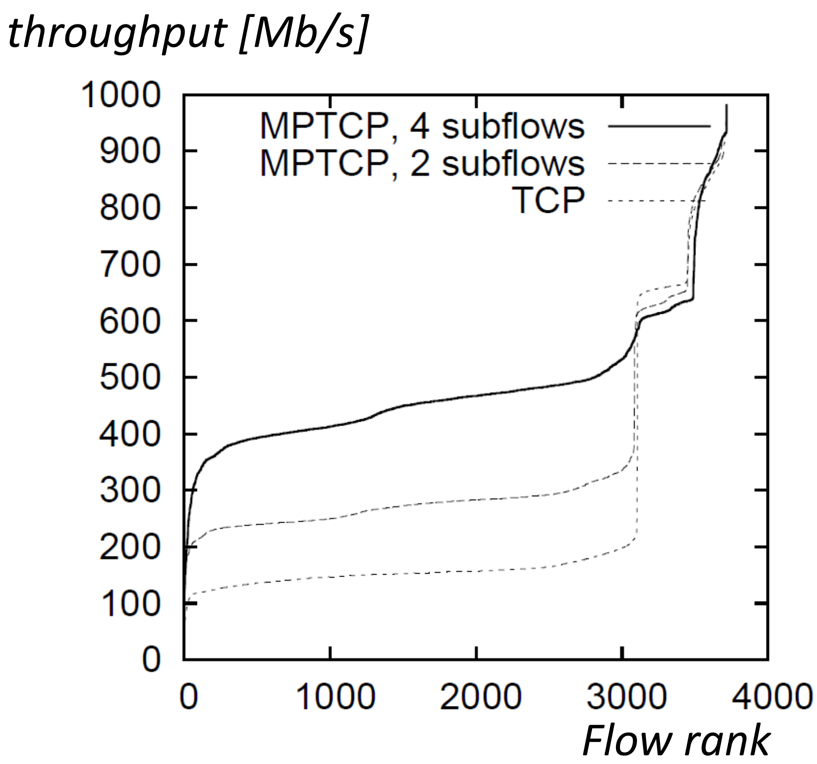
and goal 3 (fairness)

and goal 4 (rapid adjustment)

(none)

Why we like MPTCP for data centers, and why ECMP and per-packet scattering have problems.

MPTCP is a simple way to automatically pick the best of several available paths.



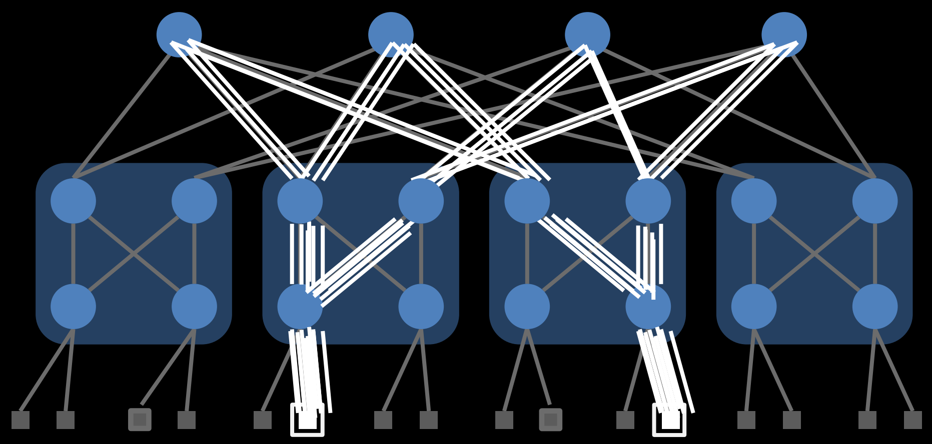
Amazon EC2 (USEast1d).
24 hours.
10 hosts, each sending to the 9 others in turn, over and over again.

Background traffic levels vary.

Traceroute shows 26% of paths were local (i.e. no alternative paths), 74% were four- or five-hop.

MPTCP discovers available capacity

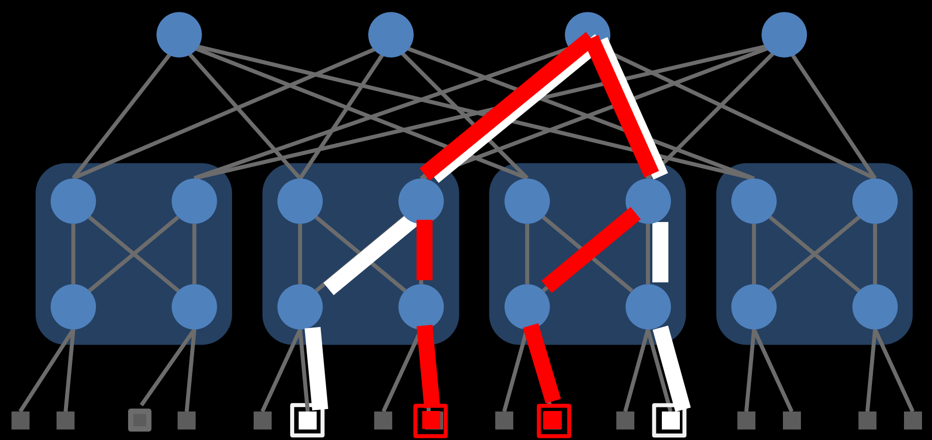
An obvious way to balance load is to use ECMP, i.e. pick randomly from available paths for each TCP flow.



This balances traffic nicely, as long as there are enough flows.

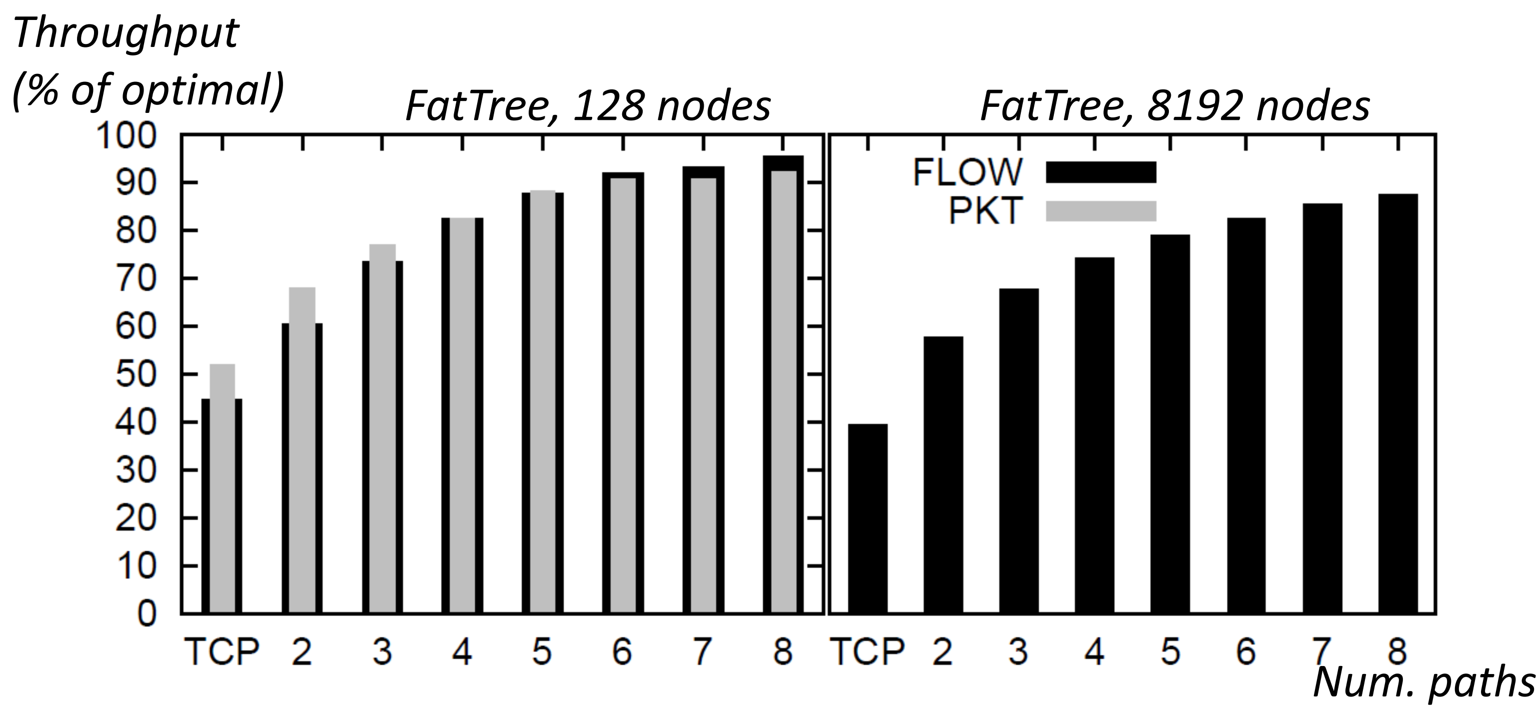
MPTCP discovers available capacity

An obvious way to balance load is to use ECMP, i.e. pick randomly from available paths for each TCP flow.



This balances traffic nicely, as long as there are enough flows. But if there are fewer flows, there may be collisions and wasted capacity.

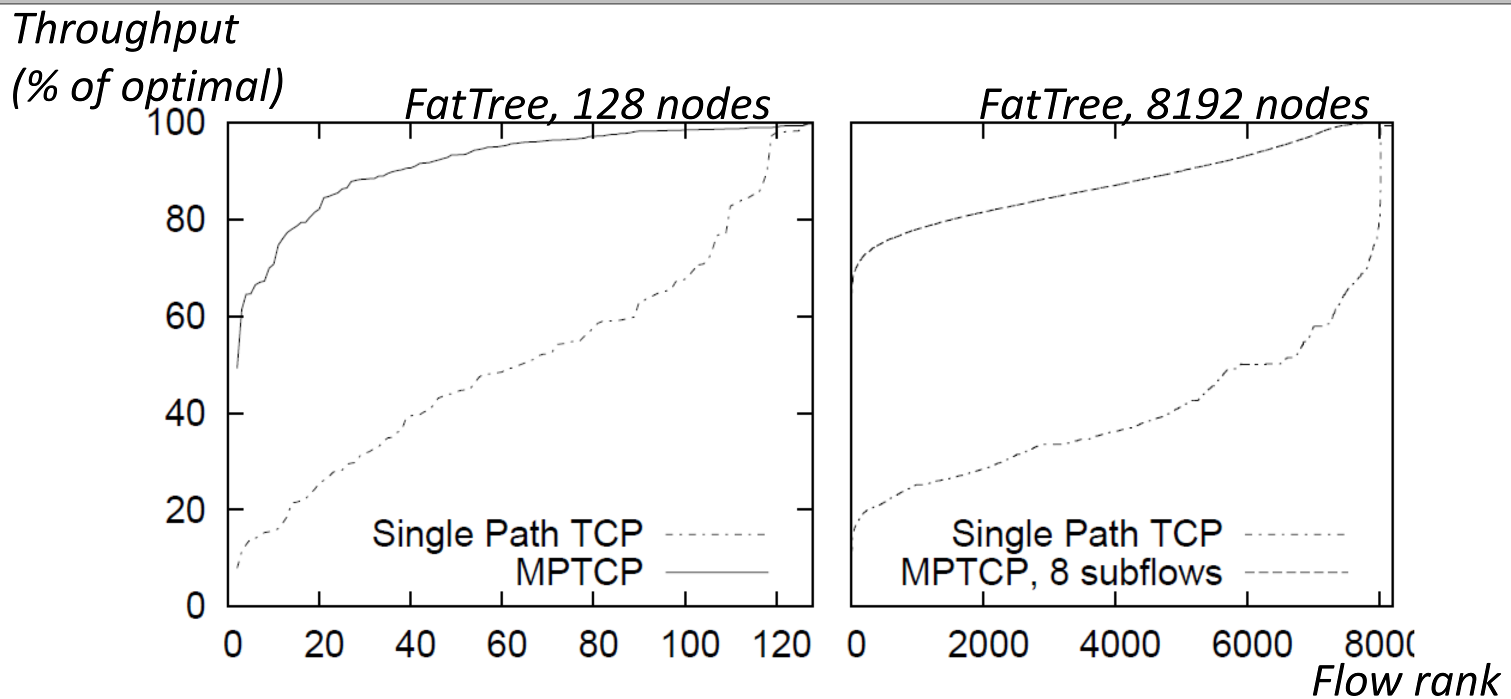
MPTCP discovers available capacity, and it doesn't need much path choice.



Simulations of FatTree, 100Mb/s links, permutation traffic matrix, one flow per host, TCP+ECMP versus MPTCP.

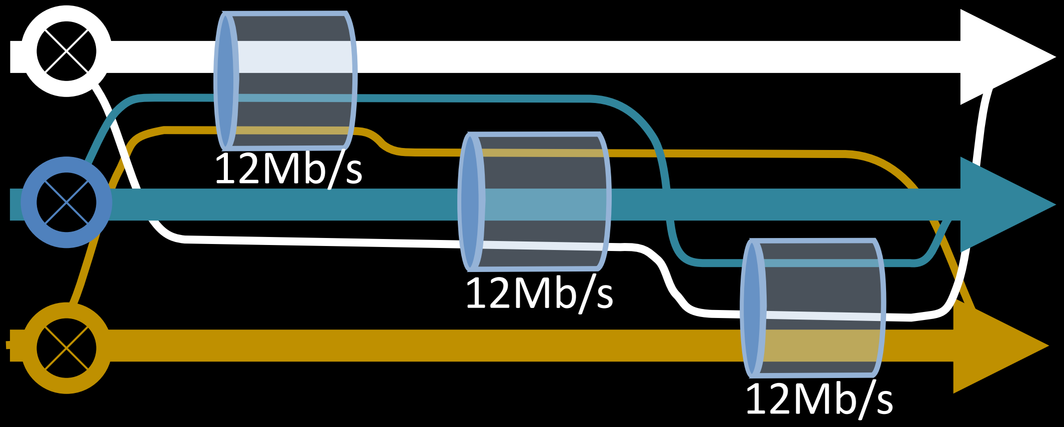
If each node-pair balances its traffic over 8 paths, chosen at random, then utilization is around 90% of optimal.

MPTCP discovers available capacity, and it shares it out more fairly than TCP+ECMP.

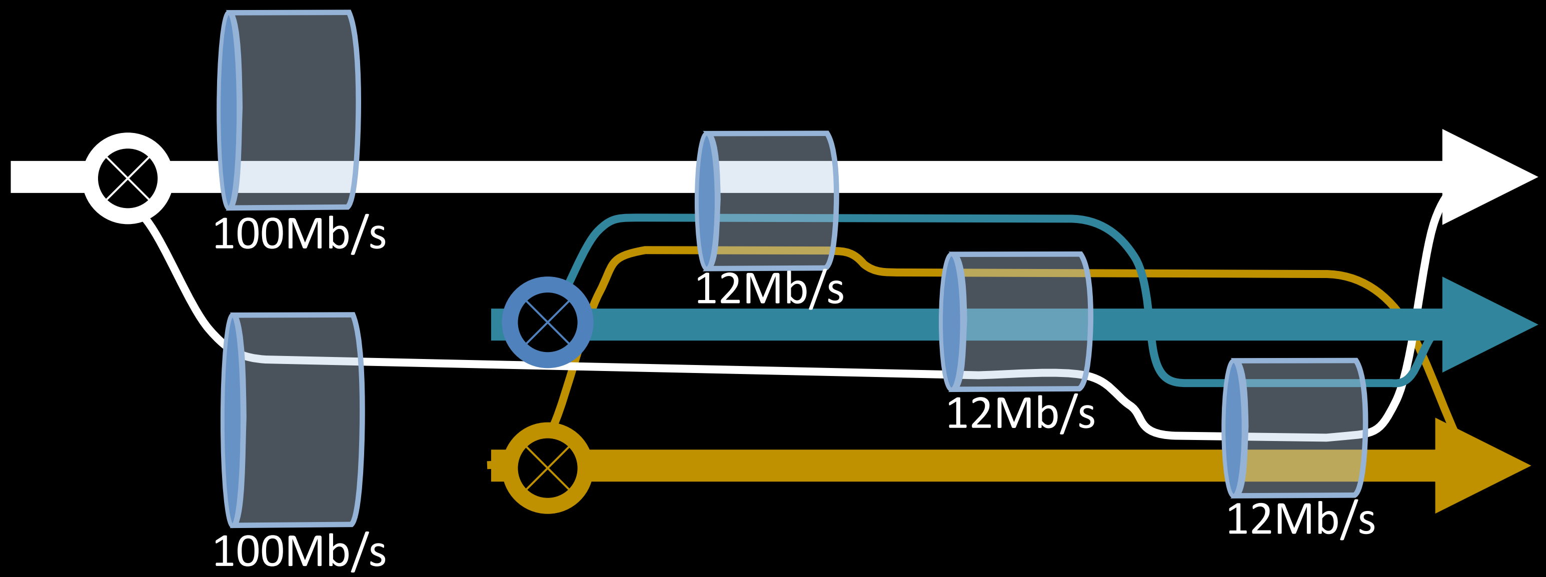


Simulations of FatTree, 100Mb/s links, permutation traffic matrix, one flow per host, TCP+ECMP versus MPTCP.

MPTCP can make good path choices, better than per-packet load balancing.



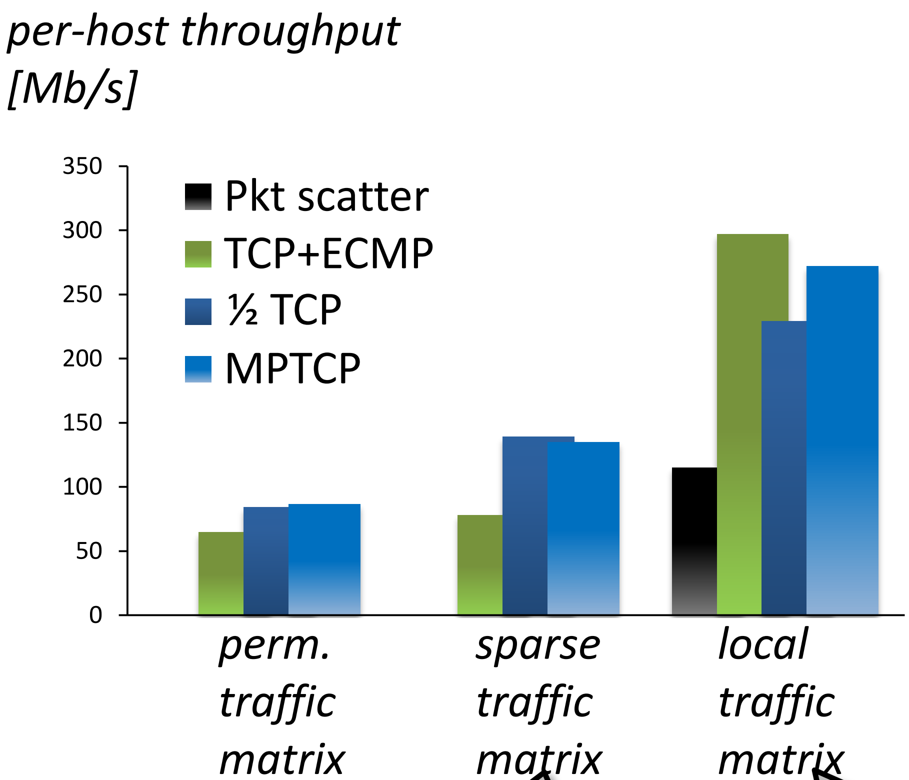
MPTCP can make good path choices, better than per-packet load balancing.



At the point where the switching decision is made, is there enough information to make the right switching choice?

The right choice might even be different for different destinations.

MPTCP can make good path choices, and it's robust against a range of traffic matrices.



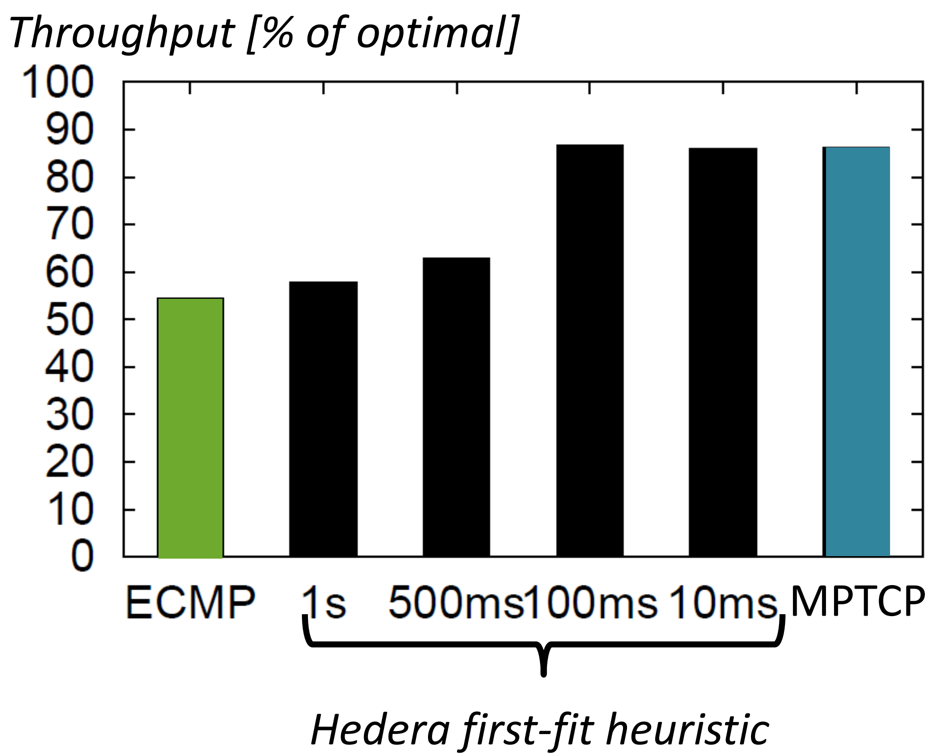
Simulations of BCube, with 125 three-interface hosts, 25 switches, 100Mb/s links.

- Pkt scatter: switches run per-packet load balancing across all available paths; use modified TCP which copes with reordering
- TCP+ECMP: only use shortest-hop paths
- ½ TCP, MPTCP: split traffic across multiple paths

ECMP, using only shortest-hop paths, is wasteful when traffic is light

per-packet scattering might send traffic over longer paths, which is a bad choice when traffic is local

MPTCP can make good path choices, as good as a very fast centralized scheduler.



Simulation of FatTree with 128 hosts.

- Permutation traffic matrix
- Closed-loop flow arrivals (one flow finishes, another starts)
- Flow size distributions from VL2 dataset

MPTCP permits flexible topologies

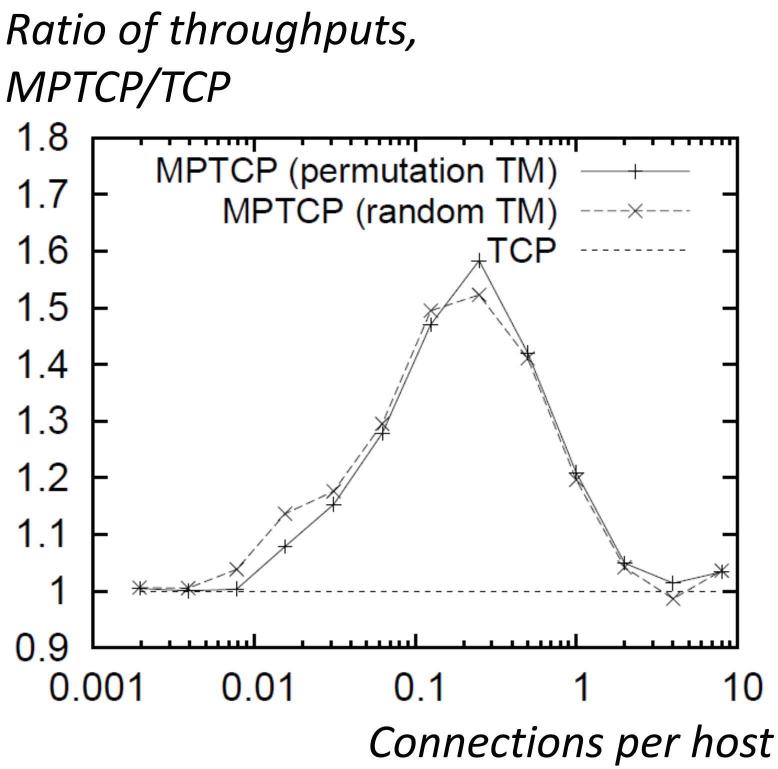
FatTree and VL2 aim to mimic a non-blocking switch, i.e. to support any permutation traffic matrix.

(They achieve this, to varying degrees, depending on the level of statistical multiplexing and/or flow placement.)

But maybe it's the wrong objective.

What if we want a cheaper "right-provisioned" network core, e.g. which lets ¼ of the hosts send at full NIC rate? What if we give them multiple NICs to allow bursts?

MPTCP permits flexible topologies



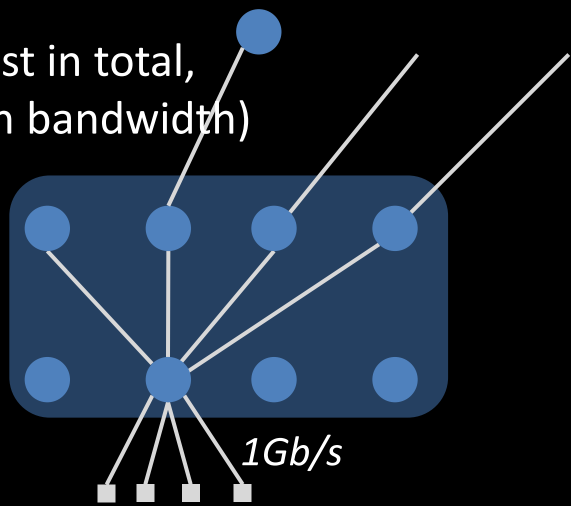
Simulation of a FatTree-like topology with 512 nodes, but with 4 hosts for every up-link from a top-of-rack switch, i.e. the core is oversubscribed 4:1.

- Permutation TM: each host sends to one other, each host receives from one other
- Random TM: each host sends to one other, each host may receive from any number

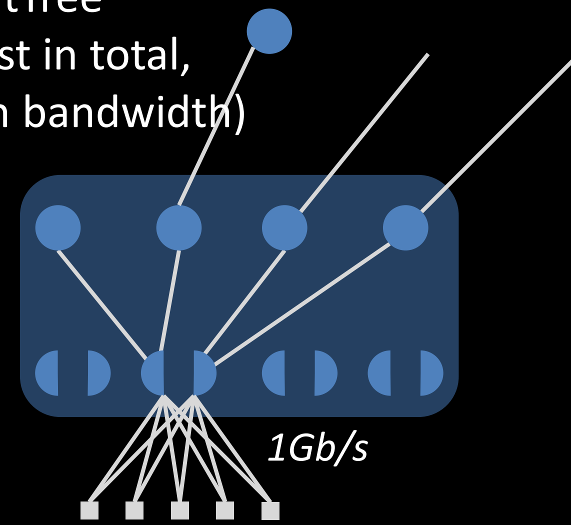
- At low loads, there are few collisions, and NICs are saturated, so TCP ≈ MPTCP
- At high loads, the core is severely congested, and TCP can fully exploit all the core links, so TCP ≈ MPTCP
- When the core is "right-provisioned", i.e. just saturated, MPTCP > TCP

MPTCP permits flexible topologies

FatTree
(5 ports per host in total,
1Gb/s bisection bandwidth)



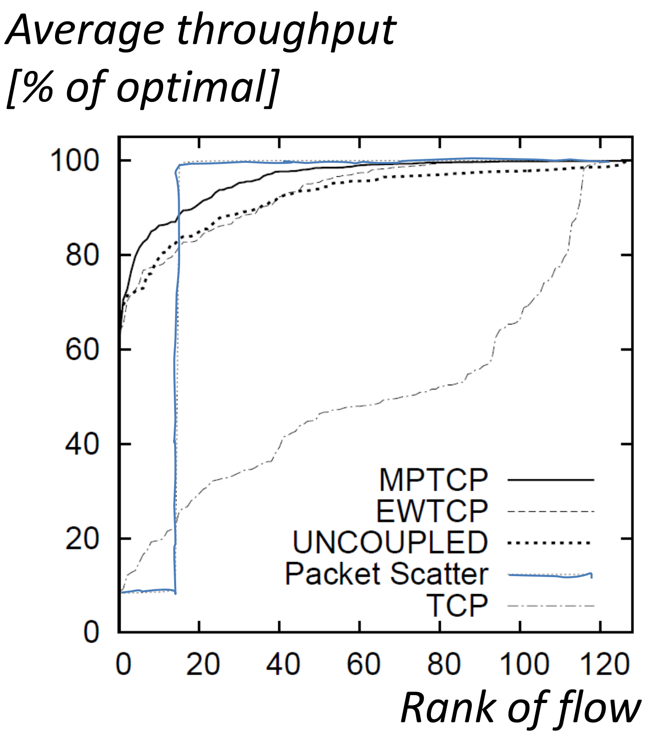
Dual-homed FatTree
(5 ports per host in total,
1Gb/s bisection bandwidth)

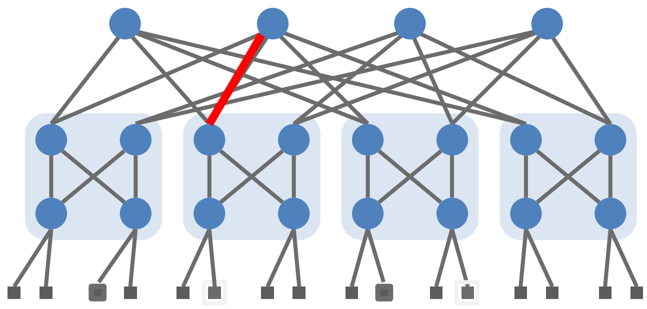


If only 50% of hosts are active, you'd like each host to be able to send at 2Gb/s, faster than one NIC can support.

If a reasonable amount of traffic is local, you'd like to carry more total traffic than 1Gb/s NICs allow.
(And dual-homing increases resilience to ToR failure, which encourages you to localize traffic.)

MPTCP permits flexible topologies

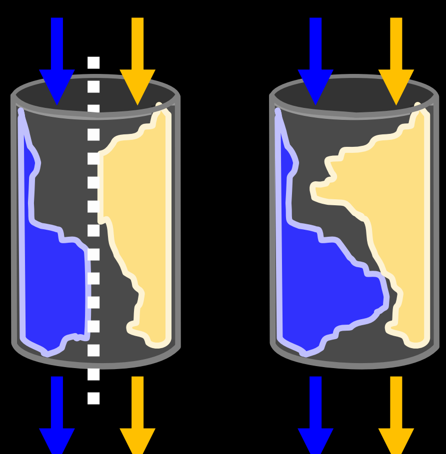


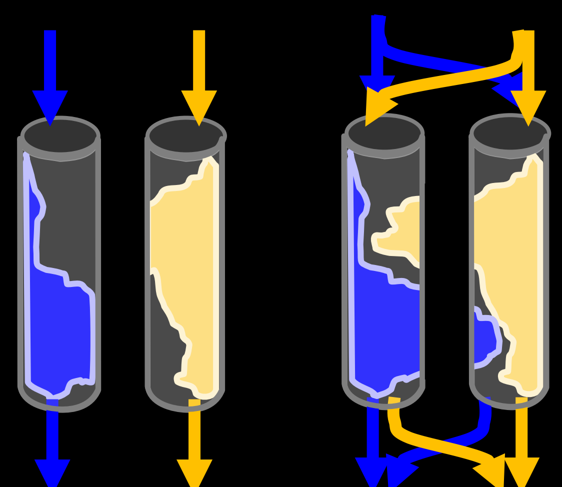


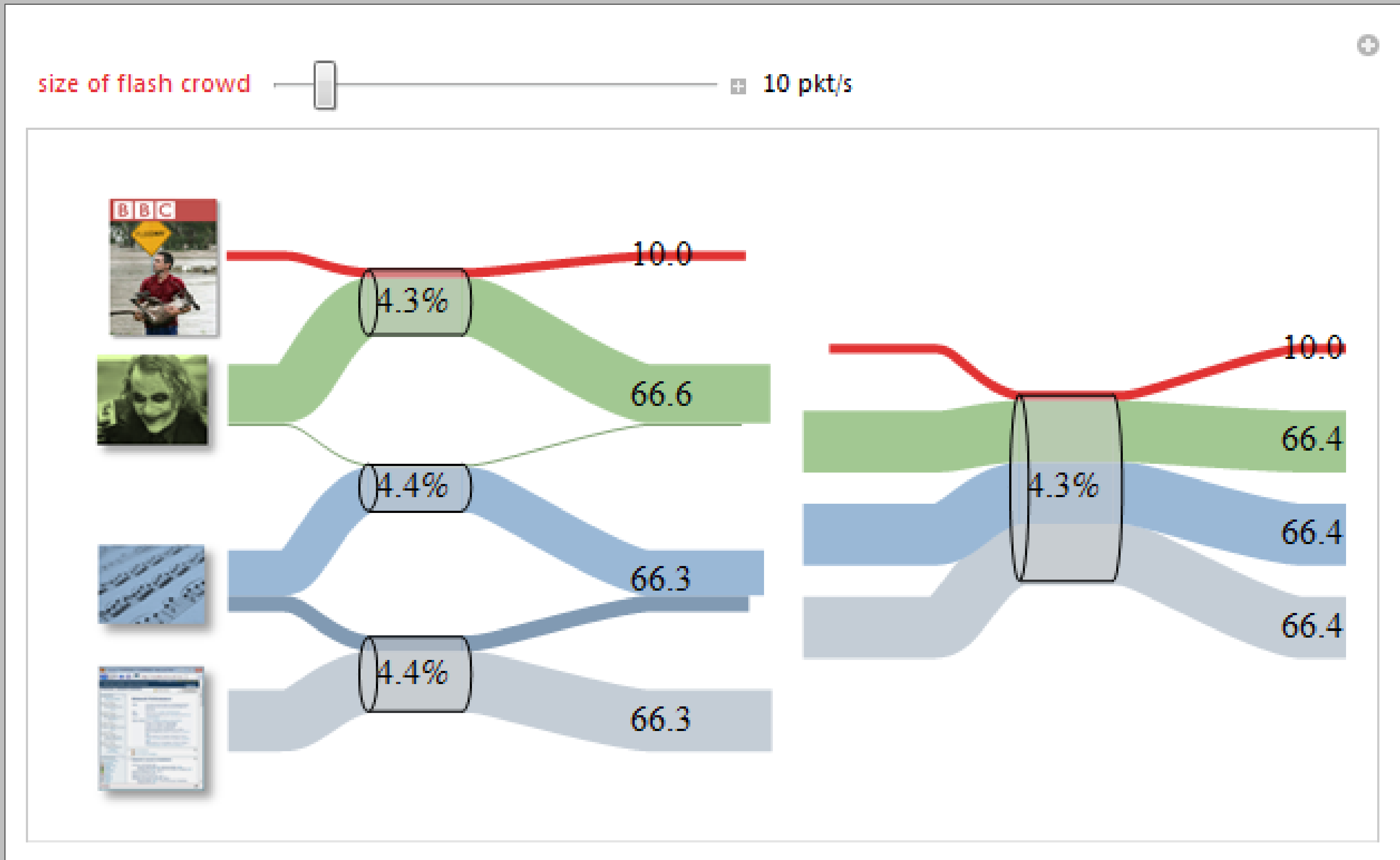
Simulation of 128-node FatTree, when one of the 1Gb/s core links is cut to 100Mb/s

Because an MPTCP flow shifts its traffic onto its least congested paths, congestion hotspots are made to "diffuse" throughout the network. Non-adaptive congestion control, on the other hand, does not cope well with non-homogenous topologies.

What is MPTCP really trying to do?

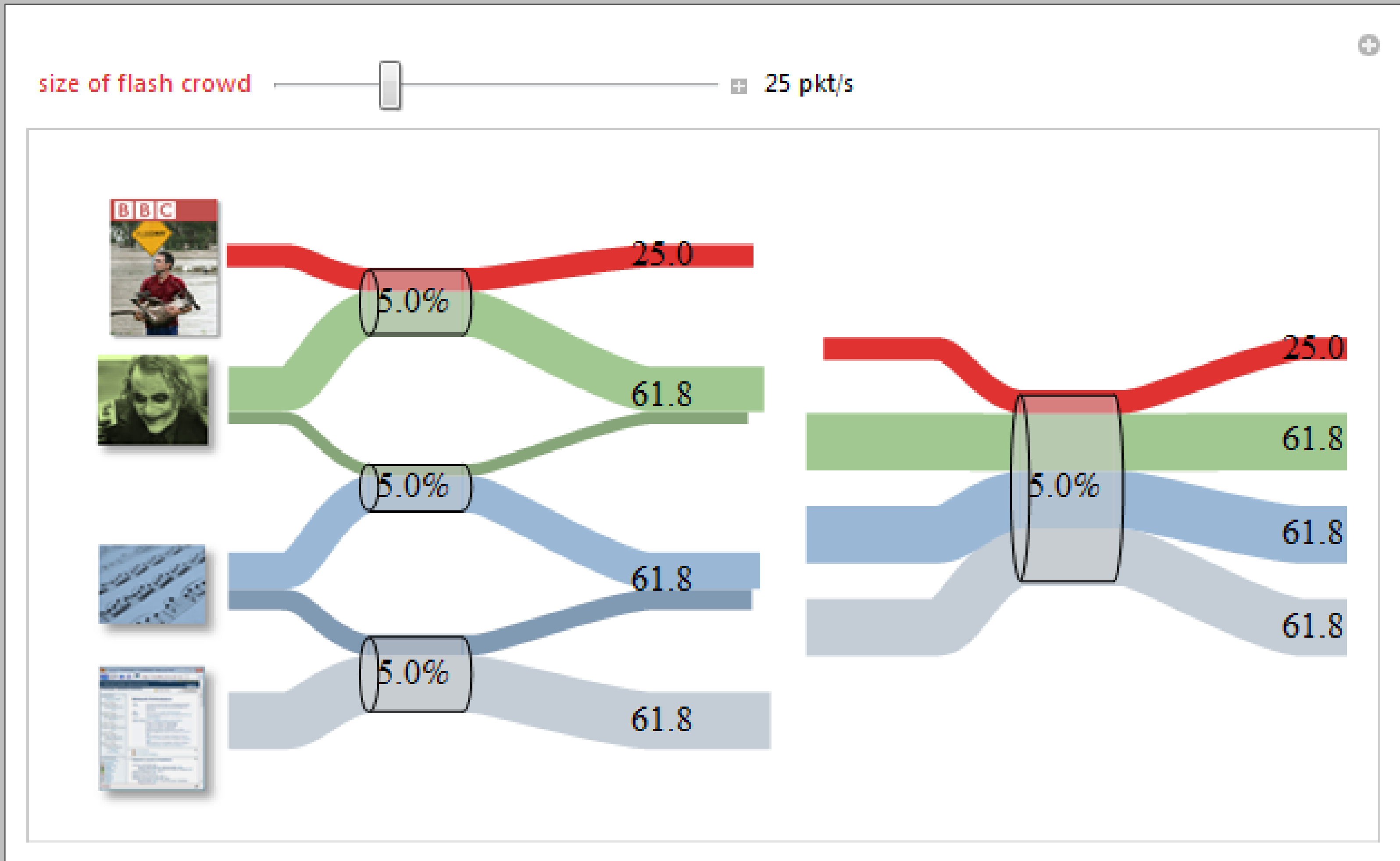






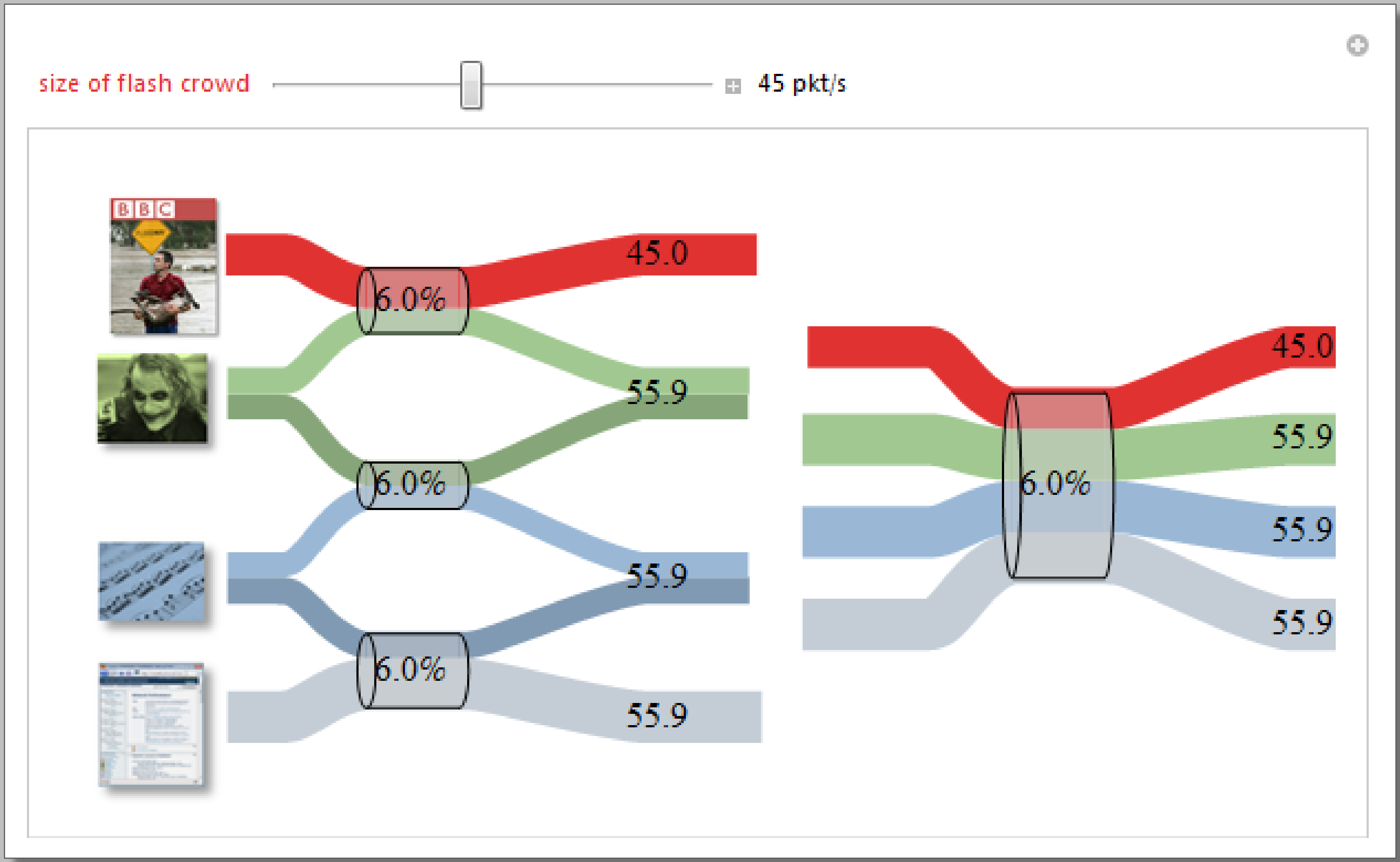
Three links,
80 Mb/s, 50 Mb/s, 60 Mb/s

One link,
190 Mb/s



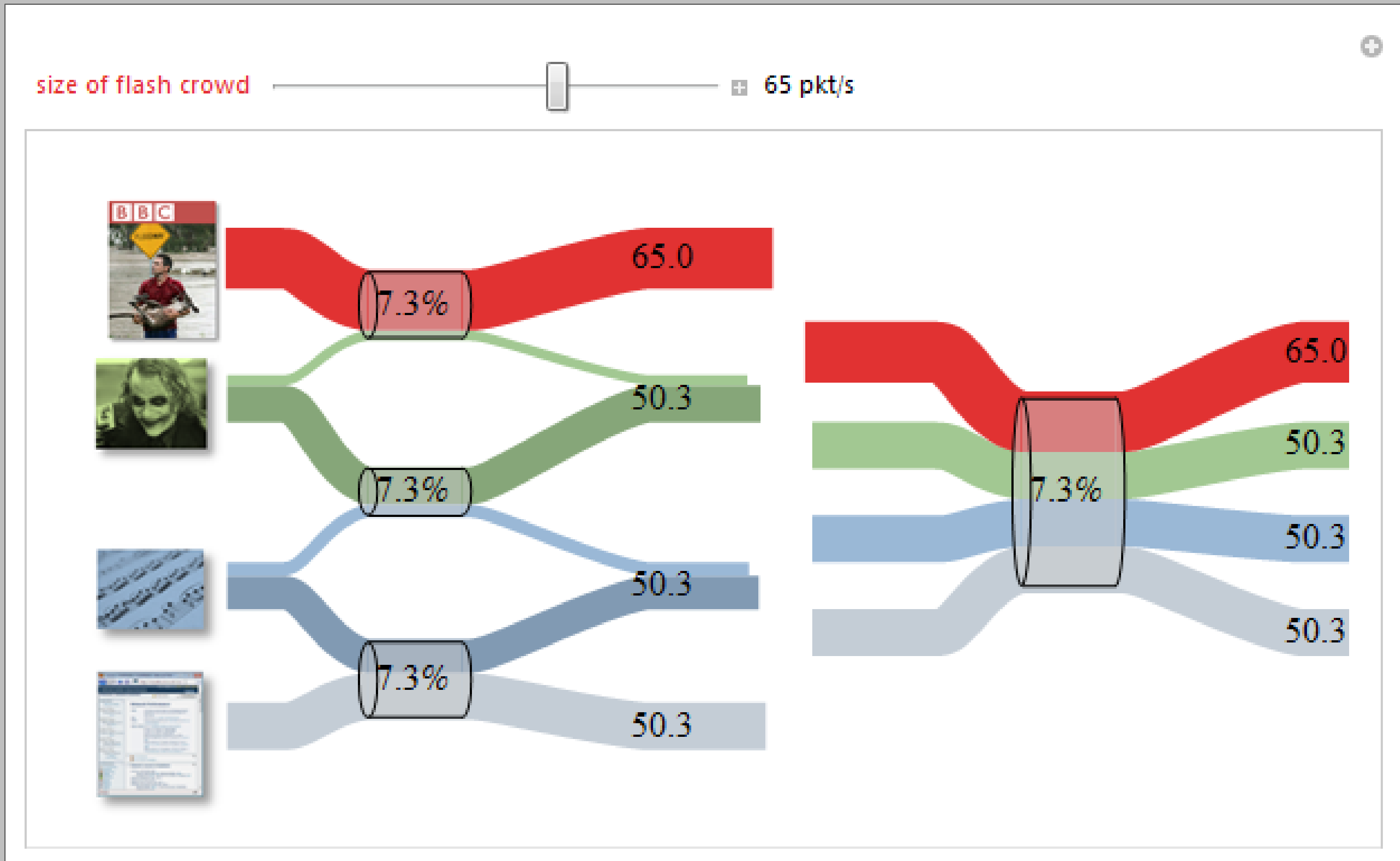
Three links,
80 Mb/s, 50 Mb/s, 60 Mb/s

One link,
190 Mb/s



Three links,
80 Mb/s, 50 Mb/s, 60 Mb/s

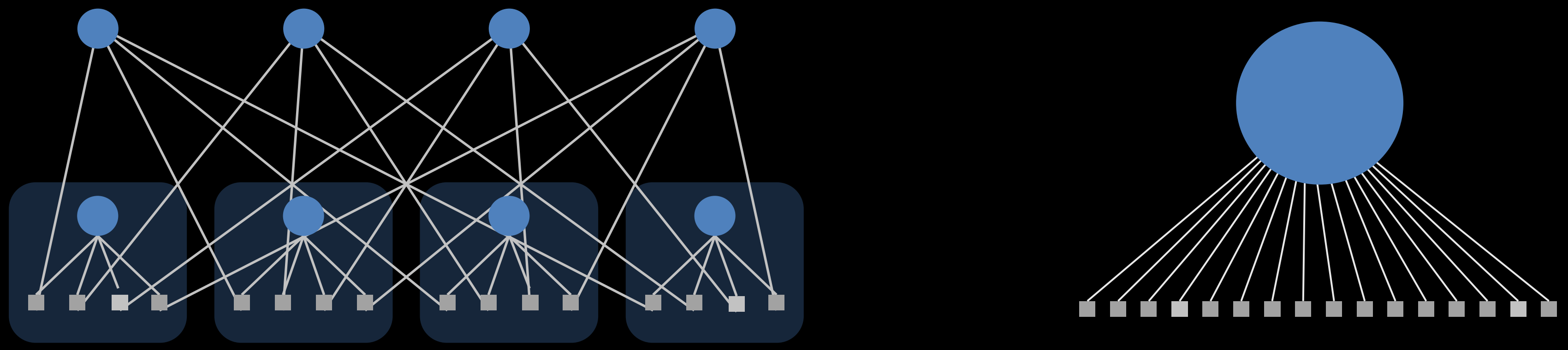
One link,
190 Mb/s



Three links,
80 Mb/s, 50 Mb/s, 60 Mb/s

One link,
190 Mb/s

MPTCP tries to make a network behave like a simple pool of capacity.



Can we design a data center topology that enables MPTCP to achieve its goal, over a good range of traffic matrices?

Further questions

How should we fit multipath congestion control to CompoundTCP or CubicTCP or DCTCP?

Is it worth using multipath for small flows?

Conclusion

- Multipath is Packet Switching 2.0
  It lets you share capacity between links.
- MPTCP is TCP 2.0
  It is a control plane to allocate resource pools.
- MPTCP is traffic engineering, done by end-systems, on a timescale of milliseconds
  which means fairer and more effective use of data center capacity.
- MPTCP means that a given network can support a wider range of traffic matrices, so it gives you more flexibility to customize the network topology.