Multipath TCP and the Resource Pooling Principle

Mark Handley, UCL and XORP, Inc

Also:

Damon Wischik, UCL

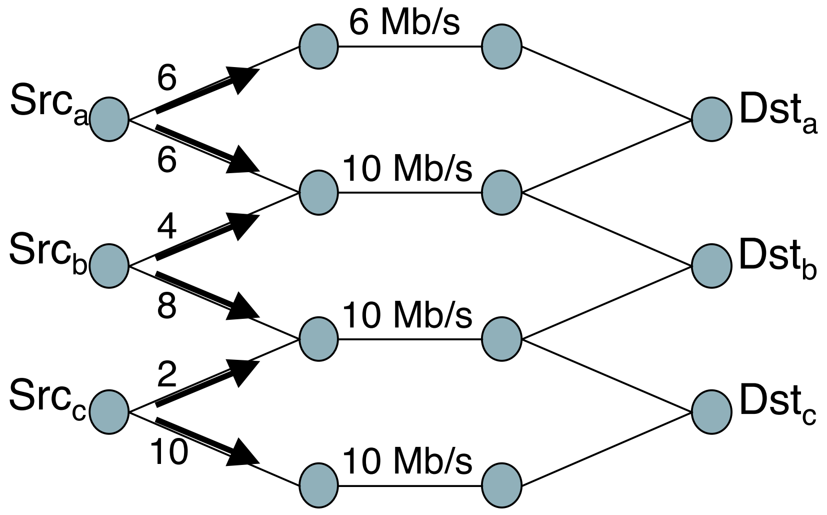
Marcelo Bagnulo Braun, UC3M

The members of **Trilogy** project: www.trilogy-project.org

Resource Pooling?

Make a network's resources behave like a **single pooled resource**.

- Aim is to increase reliability, flexibility and efficiency.
- Method is to build mechanisms for shifting load between the various parts of the network.

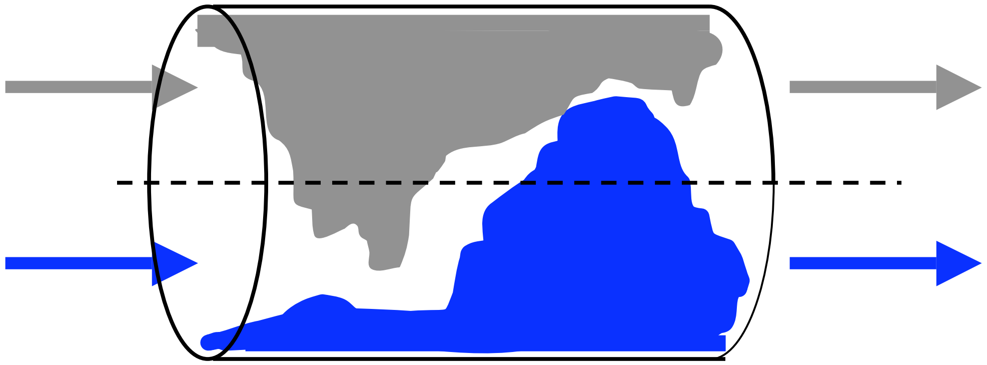


Everyone keeps reinventing resource pooling to solve their own local problems.

Resource Pooling is not new…

Computer communication is bursty, so a virtual circuit-based model with rate allocations gives poor utilization.

- A packet-switched network pools the capacity of a single link.
  - Goal: high utilization
- Router queues pool capacity from one time interval to the next
  - Goal: high utilization, robustness to arrival patterns



We’re doing resource pooling in routing

- **BGP traffic engineering:**
  - Slow manual process to pool resources across peering links.
  - Financial goal - match revenues to costs.
  - Robustness goal - prevent overload.

- **OSPF/MPLS traffic engineering:**
  - Slow mostly manual process to pool resources across internal ISP links.
  - Primary: Robustness - prevent overload.
  - Secondary: Higher utilization.

- **BT, AT&T (and others) dynamic alternative routing**
  - Robustness to overload.
  - Provide higher availability than the availability of the links/switches themselves (pool reliability)

Recent resource pooling trends

- **Multihoming**
  - Primary goal: pool reliability.
  - Secondary goal: pool capacity
- **Google, Akamai, CDNs**
  - Pool reliability of servers, datacenters, ISPs.
  - Pool bandwidth.
  - Pool latency??
- **Bittorrent**
  - Overall: Pool upstream capacity (over space and time)
  - Per-chunk: pool for reliability from unreliable servers.

Summary:
Motivations for Resource Pooling

- Robustness
- Increased capacity or utilization

Currently two main resource pooling mechanisms:

- **Routing-based traffic engineering.**
  - Inter-domain routing is too slow and doesn't scale well (especially if a human is in the control loop)
  - Intra-domain routing is better, but not fast enough with a human in the loop, not stable if automatic.
  - There are many examples where no network-based flow-based mechanism can achieve pooling.
- **Application-based load-balancing between multiple servers.**
  - Pretty effective, but strong tussle with what the network operators are doing.

The requirements have changed

- Need a more robust Internet than we can get from simply making better components.
    - Traditional routing can't solve this in a scalable way.

- Applications are becoming more demanding:
    - VoIP, TV, Games.

- Most of the end-systems will be mobile, with multiple radios that can be used simultaneously.

So what might work?

- Multipath.
  - Only real way to get robustness is redundancy.
- Multihoming, via multiple addresses.
  - Can aggregate routing information.
- Mobility, via adding and removing addresses.
  - No need to involve the routing system, or use non-aggregatable addresses.

So what might work?

- **Multipath-capable transport layers**.
  - Use multiple subflows within transport connections.
  - Congestion control subflows independently
  - Traffic moves to the less congested paths.

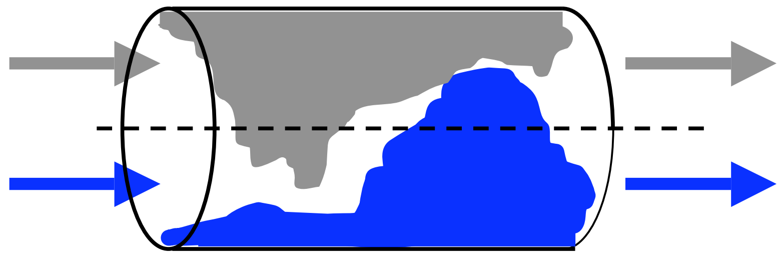
- Note the involvement of congestion control is crucial.
  - Link the backoff parameters for stability and fairness (Kelly+Voice).
  - You can't solve this problem at the IP layer.
- Moves some of the stresses out of the routing system.
  - Might be able to converge slowly and no-one cares?

Multipath transport

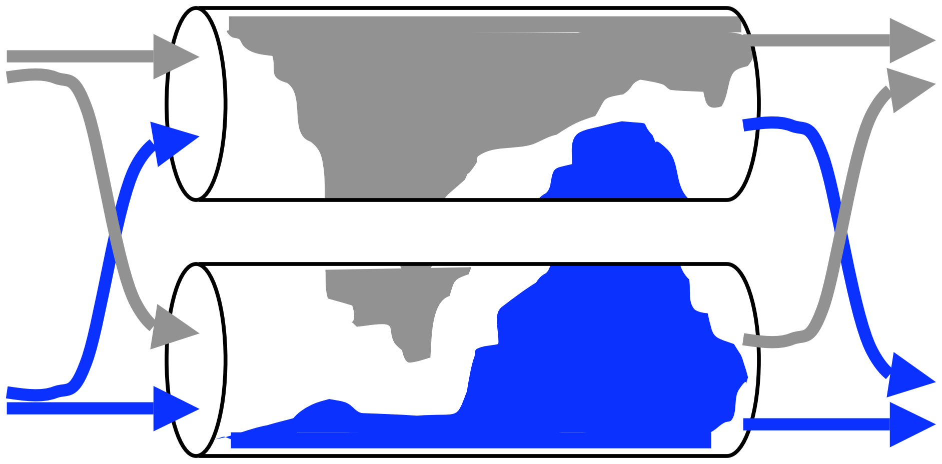
- Multipath transport allows multiple links to be treated as a single pooled resource.

- Traffic moves away from congested links.
- Larger bursts can be accommodated.

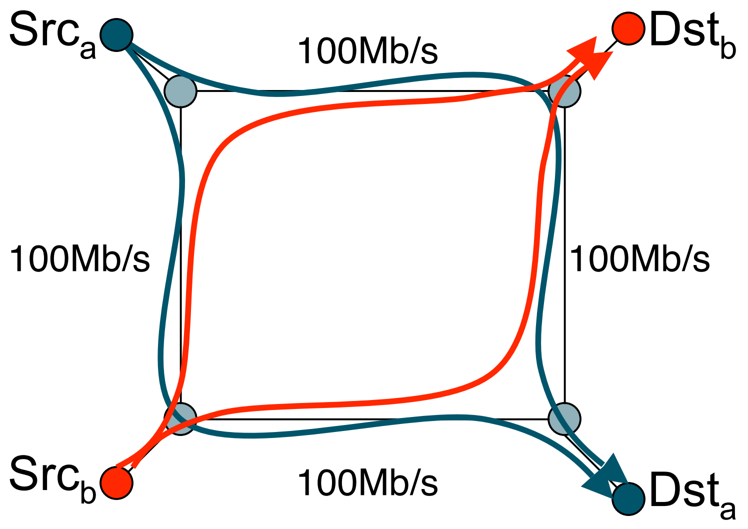
ARPAnet resource pooling:

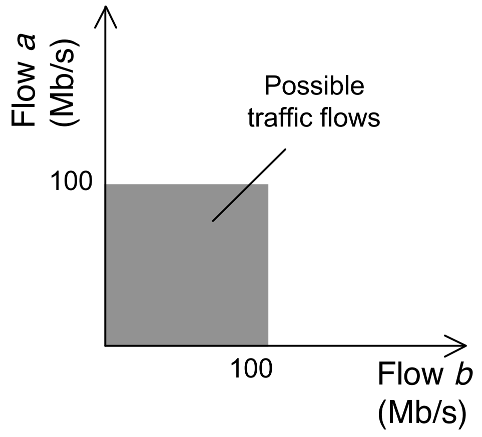


Multipath resource pooling:

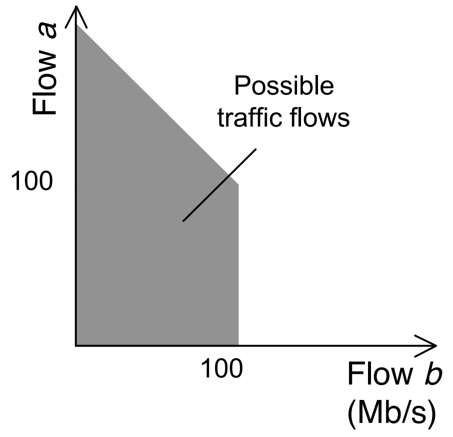


Resource pooling allows a wider range of traffic matrices

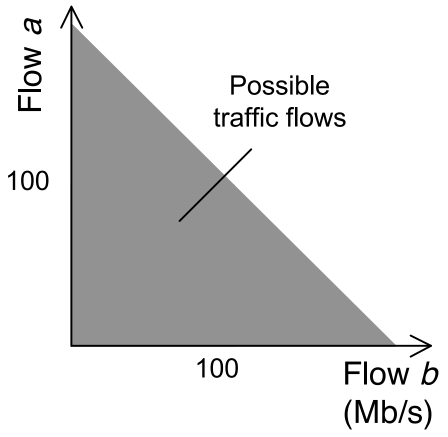




No multi-path flows

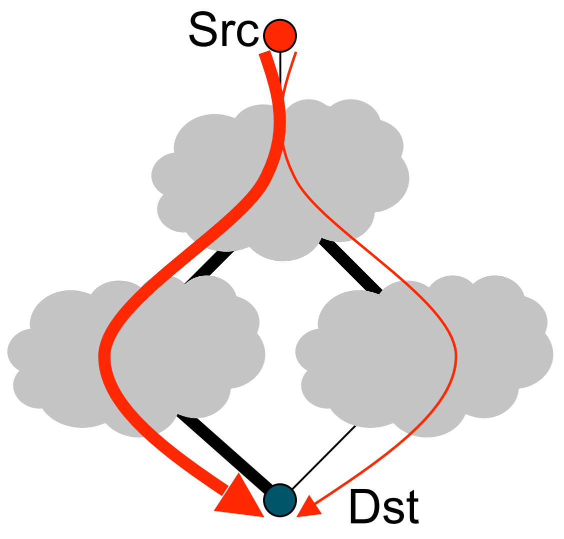


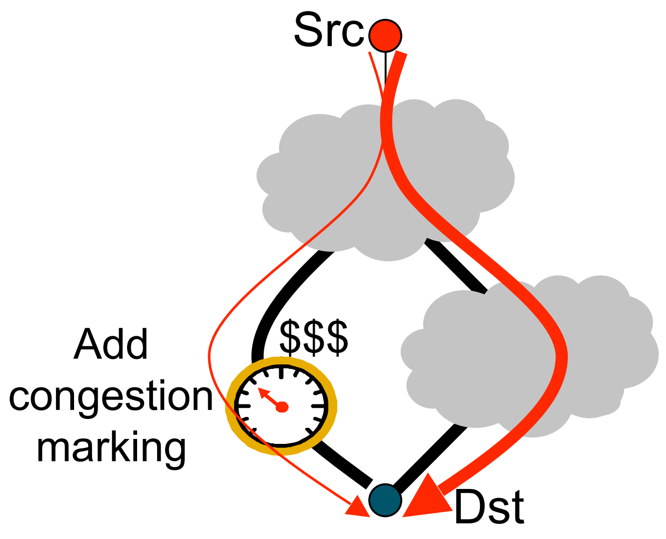
Only flow *a* is multi-path.



Both flows are multi-path

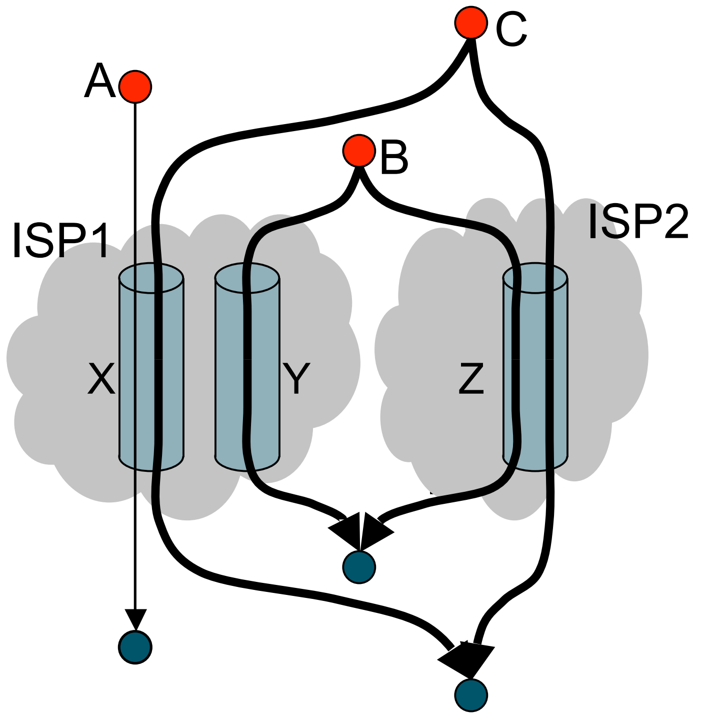
Multipath Traffic Engineering

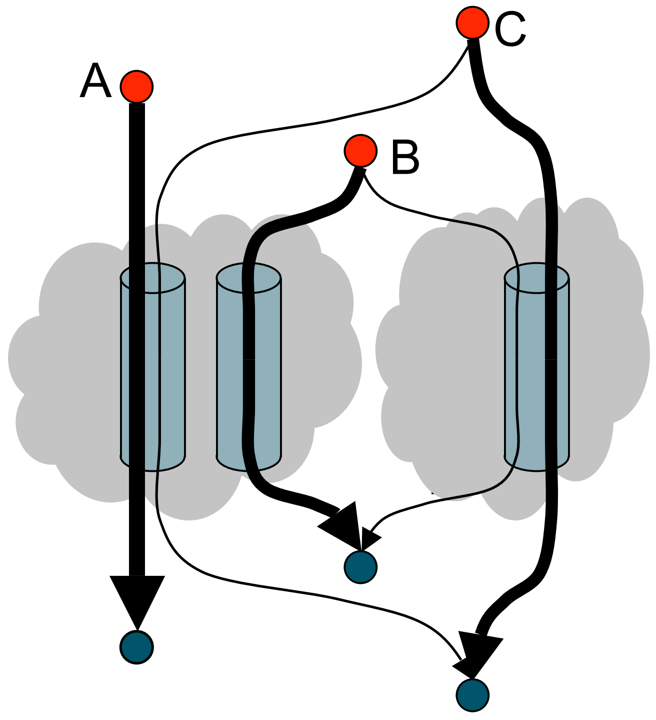




- Balancing across dissimilar speed links
- balancing across dissimilar cost links

End-systems can optimize globally (often ISPs cannot)





Existing Multipath Transport

We already have it: BitTorrent.

Providing traffic engineering for free to ISPs who don't want that sort of traffic engineering :-)

If flows were accountable for congestion, BitTorrent would be optimizing for cost.

The problem for ISPs is that it reveals their pricing model is somewhat suboptimal.

Robustness at an Affordable Price

- What if all flows looked a bit like BitTorrent?
  - Can we build an extremely robust and cost effective network for billions of mobile hosts based on multipath transport and multi-server services?

- I think we can.
- I think we must.

The "Resource Pooling Principle"

- **Observation 1**: Resource pooling is often the only practical way to achieve resilience at acceptable cost.
- **Observation 2**: Resource pooling is also a cost-effective way to achieve flexibility and high utilization.
- **Consequence:** At every place in a network architecture where sufficient diversity of resources is available, designers will attempt to design their own resource pooling mechanisms.
- **Principle:** A network architecture is effective overall, only if the resource pooling mechanisms used by its components are both *effective* and *do not conflict with each other*.

Corollary of the "Resource Pooling Principle"

- **Principle:** A network architecture is effective overall, only if the resource pooling mechanisms used by its components are both *effective* and *do not conflict with each other*.

- **Corollary:** The most effective way to do resource pooling in the Internet is to harness the responsiveness of the end systems in the most generic way possible, as this maximizes the benefits while minimizing the conflicts.

Multipath Transport Design Space

Multipath TCP:

- So obvious it's been proposed at least four times (originally by Huitema?).
  - SCTP is already going there.
- We now understand that multipath TCP, if done appropriately, can go a long way towards solving network-wide traffic engineering problems.
- We're starting to understand the consequences of not solving the issue in a general way.

Multi-server HTTP:

- Request chunks of file, each from a different server.
- Better pooling, but less general.

P2P interactions with ISPs.

Where are we today?

- Good theoretical understanding of the issues:
  - Kelly and Voice
  - Key, Massoulié and Towsley
- Working on the details for TCP:
  - How well does it work in practice?
  - When to start additional flows?
  - Are there cases where multipath does worse?
- How much of the traffic engineering problems does this solve?
  - How much remains to be done in routing?
  - How to manage such dynamic networks?